Rectilinear-Upward Planarity Testing of Digraphs

Walter Didimo
Department of Engineering, University of Perugia, Italy

Michael Kaufmann
Department of Computer Science, University of Tübingen, Germany

Giuseppe Liotta
Department of Engineering, University of Perugia, Italy

Giacomo Ortali
Department of Engineering, University of Perugia, Italy

Maurizio Patrignani
Department of Civil, Computer and Aeronautical Engineering, Roma Tre University, Italy

Abstract

A *rectilinear-upward planar drawing* of a digraph G is a crossing-free drawing of G where each edge is either a horizontal or a vertical segment, and such that no directed edge points downward. RECTILINEAR-UPWARD PLANARITY TESTING is the problem of deciding whether a digraph G admits a rectilinear-upward planar drawing. We show that: (*i*) RECTILINEAR-UPWARD PLANARITY TESTING is NP-complete, even if G is biconnected; (*ii*) it can be solved in linear time when an upward planar embedding of G is fixed; (*iii*) the problem is polynomial-time solvable for biconnected digraphs of treewidth at most two, i.e., for digraphs whose underlying undirected graph is a series-parallel graph; (*iv*) for any biconnected digraph the problem is fixed-parameter tractable when parameterized by the number of sources and sinks in the digraph.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Dynamic programming; Theory of computation → Graph algorithms analysis

Keywords and phrases Graph drawing, orthogonal drawings, upward drawings, rectilinear planarity, upward planarity

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2023.26

Funding *Walter Didimo*: Project AIDMIX – Artificial Intelligence for Decision Making: Methods for Interpretability and eXplainability – Ricerca di Base 2021.
Giuseppe Liotta: MUR of Italy, PRIN Project n. 2022TS4Y3N – EXPAND: scalable algorithms for EXPloratory Analyses of heterogeneous and dynamic Networked Data.

Acknowledgements We thank Ignaz Rutter for conversations about the problem 1-2-SWITCH-FLOW.

1 Introduction

A *rectilinear planar drawing* of a graph G is a crossing-free drawing of G where vertices are placed at distinct points in the plane (possibly at grid points) and edges are drawn as either horizontal segments or vertical segments. RECTILINEAR PLANARITY TESTING is the problem of deciding whether a planar graph admits a rectilinear planar drawing. Besides the theoretical beauty of the problem, which belongs to the vast literature about graph planarity testing (see, e.g. [10, 38, 39] for books and surveys), the question is at the heart of those technologies that display networked data by means of orthogonal layouts, which find applications in a variety of fields, from software engineering to bioinformatics, from data bases to computer networks (see, e.g., [21, 36]).

34th International Symposium on Algorithms and Computation (ISAAC 2023).
Editors: Satoru Iwata and Naonori Kakimura; Article No. 26; pp. 26:1–26:20
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

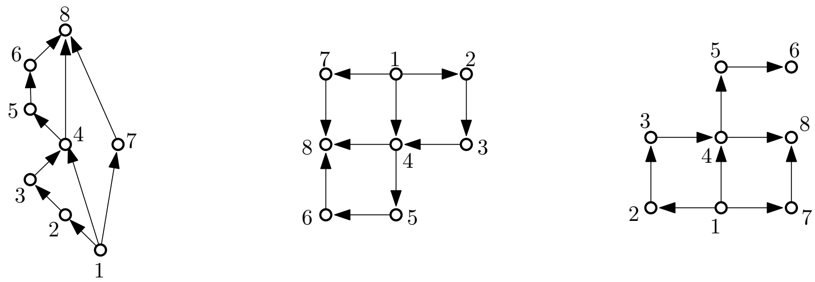
Rectilinear Planarity Testing has been proved to be NP-complete [29]. However, polynomial-time solutions are known both for constrained versions of the problem and for restricted families of graphs. Namely, Rectilinear Planarity Testing can be solved in polynomial time in the so-called *fixed-embedding setting*, that is when the input is given with a planar embedding and the testing algorithm is not allowed to change the embedding (see, e.g. [42]). Also, polynomial-time solutions are known for graphs of bounded treewidth and for sub-cubic graphs (see, e.g., [11, 12, 13, 22, 27, 31, 40, 41]).

In this paper we study rectilinear planar drawings of directed graphs (digraphs). We want to test whether a digraph G admits a rectilinear planar drawing with the additional constraint that no directed edge points downward. We call such a drawing a *rectilinear-upward planar drawing* and the testing problem Rectilinear-Upward Planarity Testing. It may be worth recalling that the problem of testing whether a digraph admits an *upward planar drawing*, i.e., a planar drawing where each edge is monotonically increasing in the upward direction according to its orientation, is NP-complete [29]. See also [1, 3, 4, 5, 7, 15, 32, 34] for polynomial-time solutions and parameterized approaches on restricted graph families or scenarios. Figure 1 shows an example of a digraph that admits both an upward planar drawing and a rectilinear planar drawing, but that does not admit a rectilinear-upward planar drawing. Our contributions can be summarized as follows.

- We prove that Rectilinear-Upward Planarity Testing is NP-complete, even if the input digraph is biconnected (Section 3).
- We show that Rectilinear-Upward Planarity Testing can be solved in linear-time when an upward planar embedding of G is fixed as part of the input (Section 4). We remark that both the problem of testing rectilinear planarity and of testing upward planarity in linear time in the fixed-embedding setting are among of the most famous and long-standing open problems in graph drawing (see, e.g., [6, 43]).
- We consider the variable-embedding setting, where the algorithm is free to chose the planar embedding of the input graph, and we focus on families of biconnected digraphs (Section 5). We show that Rectilinear-Upward Planarity Testing can be solved in polynomial time for biconnected digraphs with treewidth at most two, i.e., when the underlying undirected graph is series-parallel. We recall that polynomial-time testing algorithms for series-parallel graphs are known in the literature both in the context of upward planarity testing only and in the context of rectilinear planarity testing only (see, e.g., [8, 19, 16]). We also show that Rectilinear-Upward Planarity Testing is FPT when parameterized by the number k of sources and sinks of the digraph. Namely, for any n-vertex digraph our FPT algorithm is single-exponential in k and has a quadratic factor in n. We remark that parameterized complexities of upward and rectilinear planarity testing are topics that have been receiving increasing attention (see, e.g., [7, 14, 35]).

From a technical point of view, our linear-time algorithm in the fixed-embedding setting exploits a 2-SAT formulation, instead of using network-flow models as done in the standard approaches for testing both rectilinear and upward planarity (see, e.g., [1, 3, 9, 28, 42]). In the variable-embedding setting, we rely on the concept of *rectilinear-upward spirality*. It combines the notion of spirality introduced in [11] to measure how much a triconnected component of a rectilinear drawing can be "rolled up", with additional information about the orientation of the edges incident to the poles of the triconnected components.

For space restrictions some proofs are sketched or omitted. Full proofs will appear in an extended journal version of the paper.



(a) **(b)** **(c)**

Figure 1 A graph G that is upward planar, rectilinear planar, but not rectilinear-upward planar. (a) An upward planar drawing of G. (b) A rectilinear planar drawing of G. (c) A rectilinear-upward planar drawing of G without edge $(6, 8)$.

2 Basic Definitions and Properties

For basic definitions on graph drawing and planarity refer to [10]. We assume to work with connected graphs, as otherwise we can treat each connected component of the graph independently. A 4-*graph* is a graph with vertex-degree at most four.

Upward planar drawings. In an *upward drawing* of a digraph G each edge is represented as a Jordan arc monotonically increasing in the upward direction, according to its orientation; see Figure 1a. A digraph G is *upward planar* if it admits an upward planar drawing. Clearly, a necessary (but not sufficient) condition for G to be upward planar is that G is acyclic.

Orthogonal drawings and representations. Let G be a planar (undirected or directed) 4-graph, and let Γ be a planar drawing of G. We say that Γ is an *orthogonal planar drawing* of G if each edge is drawn as a sequence of horizontal and vertical segments. A *bend* on an edge is the contact point between a horizontal and a vertical segment of the edge. An *orthogonal planar representation* H of a planar graph G is a class of shape equivalent orthogonal planar drawings; namely, H describes the planar embedding of G, the sequence of left/right bends along the edges, and the angles at every vertex of G, each angle formed by two (possibly coincident) consecutive edges around the vertex and expressed as a value in the set $\{90°, 180°, 270°, 360°\}$. If H is the orthogonal representation of an orthogonal planar drawing Γ, we also say that Γ *preserves* H and that Γ is a *drawing of* H. A drawing of H can be computed in linear time [42], thus we can concentrate on computing orthogonal representations rather than drawings.

Orthogonal planar drawings (resp. representations) without bends are called *rectilinear planar drawings* (resp. *rectilinear planar representations*); see, e.g., Figure 1b. A graph G is *rectilinear planar* if it admits a rectilinear planar drawing (or representation). We say that a rectilinear planar representation H is *oriented* if it also specifies for each edge (u, v) of G the relative position of u with respect to v, i.e., whether u must be to the left, to the right, above, or below v in every rectilinear drawing of H; in particular, this information establishes for each edge e of G if e is horizontal or vertical in H (it is actually enough to specify the relative position of the end-vertices of one edge of H to establish the relative position of the end-vertices for every other edge). Note that the definition of oriented rectilinear representation H of G has nothing to do with the orientation of the edges when G is a digraph. For a given rectilinear representation of G there are always four different oriented versions of it, obtained by rotating one of them by an angle of $k \cdot 90°$, for $k = 0, 1, 2, 3$.

Rectilinear-upward planar representations. In this paper we deal with drawings of digraphs that are at the same time rectilinear and upward. More precisely, we do not require that an edge is strictly upward (which would prevent us from drawing it as a horizontal segment), but rather we exclude that it is drawn downward. Formally, let G be an acyclic planar 4-digraph and let Γ be a planar drawing of G. We say that Γ is a *rectilinear-upward planar drawing* of G if Γ is a rectilinear planar drawing of G with no directed edge that points downward; see, e.g., Figure 1c. This corresponds to saying that a rectilinear upward planar drawing Γ induces an oriented rectilinear planar representation H of G with the property that for each directed edge (u, v) of G, vertex u is never above vertex v. We say that H is a *rectilinear-upward planar representation* of G. As for rectilinear representations, H describes a class of shape equivalent rectilinear-upward planar drawings. A digraph G is *rectilinear-upward planar* if it admits a rectilinear-upward planar drawing, or equivalently, a rectilinear-upward planar representation. Clearly, rectilinear planarity is necessary for rectilinear-upward planarity. The next property implies that also upward planarity is necessary for rectilinear-upward planarity. As already observed, both rectilinear planarity and upward planarity are not sufficient conditions when considered independently (see Figure 1).

▶ **Property 1.** *If Γ is a rectilinear-upward planar drawing of a digraph G, then Γ can be transformed into an upward planar drawing of G with the same planar embedding as Γ.*

In the remainder we only consider planar 4-digraphs, thus we often omit the term "planar" and we avoid to specify that the vertex-degree is at most four. Also, we often use the abbreviation "RU" in place of "rectilinear-upward". Finally, we implicitly assume that the input digraphs are acyclic, as otherwise an upward drawing, and hence an RU drawing, cannot exist. Acyclicity can be tested in linear time, by a classical depth first search.

3 NP-Completeness of Rectilinear-Upward Planarity Testing

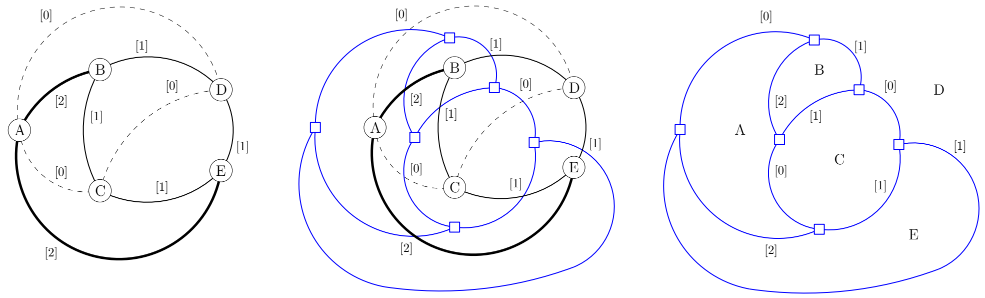
To prove the hardness of Rectilinear-Upward Planarity Testing, we use a reduction from the following 1-2-Switch-Flow problem, introduced in this paper and that may be considered of independent interest. The hardness of 1-2-Switch-Flow can be proved with a reduction from the problem Flow Orientation, which is shown to be NP-complete even if edge capacities are $O(\sqrt{n})$, where n is the number of vertices of the graph [23].

Problem: 1-2-Switch-Flow (12SW)

Instance: A planar undirected graph $G = (V, E)$ where each edge $e \in E$ is labeled with a value $f_e \in \{1, 2\}$.

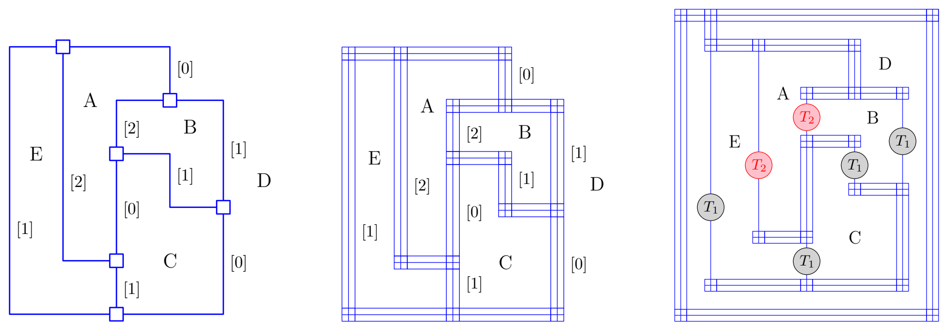
Question: Does there exist an orientation for the edges in E such that for each vertex the sum of the values of the incoming edges equals the sum of values of the outgoing edges?

We now sketch the reduction from 1-2-Switch-Flow (12SW) to Rectilinear-Upward Planarity Testing. Let $G = (V, E)$ be an instance of 12SW. We first compute a planar embedding of G and planarly add extra edges with label [0] (called [0]-*edges*) in such a way to obtain a maximal plane graph G^+ (see Figure 2a). Second, we compute the dual plane graph G^* of G^+ and label the edges of G^* with the values of the corresponding edges of G^+ (see Figures 2b and 2c). Third, we compute an orthogonal drawing Γ_{G^*} of G^* such that each edge has at least one vertical segment (see Figure 3a). Fourth, we transform Γ_{G^*} into an auxiliary positive instance F of Rectilinear-Upward Planarity Testing by replacing orthogonal and vertical segments with rectangular boxes. In particular, each edge



(a) Graph G^+. **(b)** Graph G^+ and its dual G^*. **(c)** Graph G^*.

Figure 2 The first two steps of the reduction from 12SW to Rectilinear-Upward Planarity Testing.



(a) Drawing Γ_{G^*} of G^*. **(b)** Instance F. **(c)** Instance I_{RUPT}.

Figure 3 The last steps of the reduction from 12SW to Rectilinear-Upward Planarity Testing.

segment of Γ_{G^*} is replaced with three parallel edges and each vertex or bend is replaced with a 3×3 grid (see Figure 3b). Edges of the instance F are oriented so that F admits a unique rectilinear-upward planar representation $\mathcal{H}_F$ up to a horizontal flip. Fifth, for each edge e of G^* labeled [1] (labeled [2], respectively), we identify the three parallel edges of F corresponding to a vertical segment of Γ_{G^*} belonging to e and replace them with the subgraph T_1 (the subgraph T_2, respectively). Subgraphs T_1 and T_2, called *tendrils*, are depicted in Figure 7a and Figure 7b, respectively. Finally, we add a framework all around the graph and attach it to a vertex or bend in the upper part of Γ_{G^*} (as in Figure 3c) to obtain the desired instance I_{RUPT} of Rectilinear-Upward Planarity Testing.

▶ **Theorem 1.** *Rectilinear-Upward Planarity Testing is NP-complete.*

Sketch of Proof. The problem is in NP since the recognition of an RU representation is polynomial-time solvable. As for the hardness, we show that the above described reduction from 1-2-Switch-Flow constructs an instance I_{RUPT} of Rectilinear-Upward Planarity Testing that admits an RU representation if and only if its tendrils are embedded in such a way that, for each face of Γ_{G^*}, the number of extra 270° angles provided by some tendrils equals the number of extra 90° angles provided by the remaining tendrils of the same face and this is equivalent to finding a feasible flow for the original 1-2-Switch-Flow instance. ◀

4 Testing Upward Plane Digraphs in Linear Time

If we have in input a rectilinear planar representation H of a digraph G, testing whether H is also an RU representation for one of the four possible orientations of H is a trivial problem. On the contrary, given a plane graph G with a prescribed "upward planar embedding", testing whether it admits an RU representation is a relevant problem. In this section we address this problem and present a linear-time algorithm to test whether an *upward plane* digraph G admits an RU representation. We recall that an early paper in the graph drawing literature [26] claims the result of this section. Unfortunately, that paper only gives a sketch about the algorithm to test RU planarity without giving sufficient details and simultaneously referring to a much more restrictive model [25].

Before describing our algorithm, we formalize the concept of upward plane digraph. In any RU representation H of a digraph G each vertex v is *bimodal*, i.e., all the incoming edges of v (as well as all the outgoing edges of v) are consecutive around v. More specifically, H induces: (a) a planar embedding of G and, (b) for each vertex v of G, a linear left-to-right (possibly empty) list of the incoming edges of v and a linear left-to-right (possibly empty) list of the outgoing edges of v. The information (a) and (b) together are called an *upward planar embedding* of G. A digraph G is upward plane if it comes with a given upward planar embedding. An RU representation of an upward plane digraph G (if any) is an RU representation of G that preserves its upward planar embedding. Note that, given information (a) and (b) for a planar digraph G, it can be easily checked in linear time whether this pair correctly defines an upward plane embedding, i.e., if there exists an upward planar drawing of G whose upward plane embedding coincides with (a) and (b) (see e.g. [3]).

The main ingredient of our approach is a 2-SAT formulation of the testing problem. It consists of three phases, summarized hereunder and then described in more detail.

- Phase 1: For each vertex w and for each edge e outgoing w, we assign a set $\lambda_{\text{out}}(e)$ of labels to e, encoding the sides by which e can leave w. Each of these labels is chosen in the set $\{E, W, N\}$ (East, West, or North). Similarly, for each edge e incoming w, we assign to e a set $\lambda_{\text{in}}(e)$ of $\{E, W, S\}$ (East, West, or South), encoding the sides by which e can enter w.
- Phase 2: Based on $\lambda_{\text{out}}(e)$ and $\lambda_{\text{in}}(e)$ for each directed edge $e = (u, v)$, we compute a set $\lambda(e)$ of labels, each label taken in the set $\{L, U, R\}$, such that $|\lambda(e)| \leq 2$. The set $\lambda(e)$ encodes the possible directions (L=leftward, U=upward, R=rightward, respectively) that an edge can have in an RU representation. If $\lambda(e)$ is an empty set then the input graph does not have an RU representation. The function λ is a *candidate set of labels* for the edges of G.
- Phase 3: By exploiting the labels associated with the edges, RU planarity is modeled as a 2-SAT formula ϕ, which is then solved in linear time [37].

Details for Phase 1. We describe how to define the sets $\lambda_{\text{out}}(\cdot)$ and $\lambda_{\text{in}}(\cdot)$ for every edge outgoing or incoming a vertex w of G. (i) If w has three outgoing (resp. incoming) edges e_1, e_2, e_3, in this left-to-right order in the upward planar embedding of G, then the sides from which these edges are incident to w can be uniquely fixed. Namely, $\lambda_{\text{out}}(e_1) = \{W\}$, $\lambda_{\text{out}}(e_2) = \{N\}$, $\lambda_{\text{out}}(e_3) = \{E\}$ (resp. $\lambda_{\text{in}}(e_1) = \{W\}$, $\lambda_{\text{in}}(e_2) = \{S\}$, $\lambda_{\text{in}}(e_3) = \{E\}$). (ii) If w has two outgoing (resp. incoming) edges e_1 and e_2, in this left-to-right order, we set $\lambda_{\text{out}}(e_1) = \{W, N\}$, $\lambda_{\text{out}}(e_2) = \{N, E\}$ (resp. $\lambda_{\text{in}}(e_1) = \{W, S\}$, $\lambda_{\text{in}}(e_2) = \{S, E\}$). (iii) If

w has one outgoing edge e_1 we set $\lambda_{\text{out}}(e_1) = \{W, N, E\}$ in all cases except when w has three incoming edges, in which case $\lambda_{\text{out}}(e_1) = \{N\}$. *(iv)* If w has one incoming edge e_1 we set $\lambda_{\text{in}}(e_1) = \{W, S, E\}$ in all cases except when w has three outgoing edges, in which case $\lambda_{\text{in}}(e_1) = \{S\}$.

Details for Phase 2. For each edge e, given the label sets $\lambda_{\text{out}}(e)$ and $\lambda_{\text{in}}(e)$ for e, we first initialize $\lambda(e)$ as the empty set. If $S \in \lambda_{\text{in}}(e)$ and $N \in \lambda_{\text{out}}(e)$, we add label U to $\lambda(e)$. If $E \in \lambda_{\text{out}}(e)$ and $W \in \lambda_{\text{in}}(e)$, we add label R to $\lambda(e)$. If $W \in \lambda_{\text{out}}(e)$ and $E \in \lambda_{\text{in}}(e)$, we add label L to $\lambda(e)$. We say that λ is a *good labeling* if it exists an RU representation H of G such that each edge $e \in H$ has a direction that corresponds to one of the labels of $\lambda(e)$; if so, H is said to be *compatible* with λ. Note that the labeling λ constructed as described above is such that, for each edge $e = (u, v)$, $|\lambda(e)| \leq 3$. Also, if $|\lambda(e)| = 3$ then $\lambda(e) = \{L, U, R\}$, and e is the only outgoing edge of u and the only incoming edge of v. Consider another labeling λ', derived from λ as follows: If $|\lambda(e)| \leq 2$, let $\lambda'(e) = \lambda(e)$; if $|\lambda(e)| = 3$, let $\lambda'(e) = \{U\}$. Clearly, λ' is constructed in linear time from λ and $|\lambda'(e)| \leq 2$, for every edge e of the graph. We call λ' the *reduction of* λ. The following lemma is crucial for our 2-SAT model.

▶ **Lemma 2.** *λ is a good labeling if and only if its reduction λ' is a good labeling.*

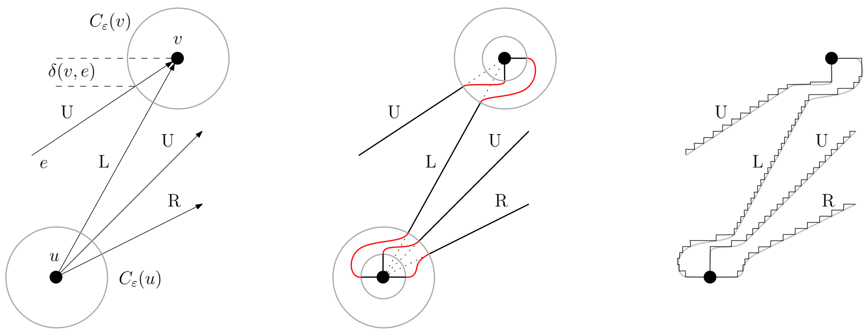
Proof. Clearly, if λ' is a good labeling then λ is, because $\lambda(e)$ is a superset of $\lambda'(e)$. Suppose, vice versa, that λ is a good labeling. We prove that λ' is a good labeling by induction on the number k of edges e for which $|\lambda(e)| = 3$. If $k = 0$, λ and λ' coincides, and the statement is obvious. Suppose that the statement is true for any $k \geq 1$, and let e be any edge for which $|\lambda(e)| = 3$. Let λ'' be the labeling obtained from λ' by setting $\lambda''(e) = \lambda(e) = \{L, U, R\}$. By the inductive hypothesis λ'' is a good labeling. Consider an RU representation H of G compatible with λ and let d be the direction of e in H. If d is the upward direction, then H is also compatible with λ', because $\lambda'(e) = \{U\}$. Otherwise (i.e., d is either the rightward or the leftward direction) there is neither an edge of H that leaves u from North nor an edge of H that enters v from South (because e is the only outgoing edge of u and the only incoming edge of v). Hence, we can derive from H another RU representation H' such that e points upward while all other edges of H' have the same direction as in H. The representation H' is now compatible with λ', which implies that λ' is a good labeling. ◀

Details for Phase 3. By Lemma 2, we can always assume that the labeling λ determined in the previous phase is such that $\lambda(e)$ contains either one or two labels, for each edge e of G. Indeed, if this is not the case, we can restrict to consider its reduction λ', obtained from λ in linear time. Let w be a vertex of G and let e_1 and e_2 be two edges of G that are either both outgoing w or both incoming w. Two labels $X \in \lambda(e_1)$ and $Y \in \lambda(e_2)$ are *conflicting* if $X = Y$. This is true, because there cannot exist an RU representation of G such that the directions of e_1 and e_2 coincide. Let e_1 be an edge outgoing w and let e_2 be an edge incoming w. Two labels $X \in \lambda(e_1)$ and $Y \in \lambda(e_2)$ are *conflicting* if X and Y represent *opposite directions* (i.e., $X = L$ and $Y = R$ or $X = R$ and $Y = L$). This phase aims to assign a single label to each edge, in such a way that there is no conflicting labels. Such an assignment (if any) is a *non-conflicting label assignment within* λ. We use the notation $\mathcal{L}(\lambda)$ to denote any non-conflicting assignment within λ. The next lemma establishes an equivalence between non-conflicting label assignments and RU representations of G compatible with λ.

▶ **Lemma 3.** *Let λ be a candidate set of labels for the edges of G. There exists an RU representation H that is compatible with λ if and only if there exists a non-conflicting label assignment within λ. The edge directions defined by H correspond to those defined by the label assignment, and H preserves the planar embedding of G.*

Proof. If there exists an RU representation H that is compatible with λ, then choosing for each edge e the label of $\lambda(e)$ that corresponds to the direction of e in H immediately yields a non-conflicting label assignment within λ.

Suppose vice versa that there exists a non-conflicting label assignment $\mathcal{L}(\lambda)$. We show that from $\mathcal{L}(\lambda)$ we can derive an RU representation H of G that is compatible with λ. Since by hypothesis G is upward planar and comes with an upward-planar embedding, there exists a straight-line upward planar drawing Γ' of G that preserves its upward planar embedding [2]. We can construct from Γ' an orthogonal-upward drawing Γ'' of G such that for each edge $e = (u, v)$: (i) the directions of the segments of e that are incident to u and v are coherent with the label of e in $\mathcal{L}(\lambda)$; and (ii) moving from u to v along e, the number of right bends equals the number of left bends.



(a) **(b)** **(c)**

Figure 4 An illustration for the proof of Lemma 3.

To construct Γ'', proceed as follows. Let $\varepsilon > 0$ be a length such that the circular area of radius ε around each vertex v does not intersect any other vertex and any other edge that is not incident to v in Γ'. For each vertex v of Γ', draw a circle $C_\varepsilon(v)$, centered at v, of radius ε (see Figure 4a). Let e be an edge incident to v. Denote by $\delta(v, e)$ the smallest vertical distance between v and the intersection of e with $C_\epsilon(v)$ (see Figure 4a). Let δ be the minimum of all $\delta(v, e)$. Draw a circle $C_\delta(v)$ of radius δ around each vertex v. Now construct a drawing of G such that each edge $e = (u, v)$ is non-decreasing with respect to the y-coordinate, leaves the u vertex and enters vertex v with a straight segment of length δ directed as prescribed by λ. This is possible because $\delta = \min_{v,e}\{\delta(v, e)\}$ (see Figure 4b). To finally obtain Γ'', we replace each edge e by a sequence of horizontal and vertical segments that follows the drawing of e at a distance that is small enough to guarantee that it does not intersect any other edge or vertex of the drawing (see Figure 4a). Since the sequence of horizontal and vertical segments of each edge e starts and ends with a segment that goes in the same direction and is non-decreasing, the numbers of right and left turns are the same.

To construct the final RU representation H, consider the orthogonal representation H'' of Γ''. Since each edge of H'' has the same number of left and right turns, by [42] there exists an orthogonal representation H of G without bends (i.e., a rectilinear representation of G) such that H has the same embedding as H'' and such that each edge in H is incident to its end-vertices from the same side as in H''. ◀

Given a non-conflicting label assignment $\mathcal{L}(\lambda)$ of G, an RU representation H of G compatible with λ and whose edge directions correspond to the edge labels of $\mathcal{L}(\lambda)$, can be easily constructed in linear time. Namely, since H preserves the planar embedding of G, for each vertex w of H the angles at w can be easily determined by the label assignment $\mathcal{L}(\lambda)$ for the edges incident to w. Also, H is oriented in such a way that the directions of the edges are coherent with λ. We now give the main result of this section.

▶ **Theorem 4.** *Let G be an n-vertex upward plane digraph. There exists an $O(n)$-time algorithm that tests whether G admits an RU representation, and that computes one in the positive case.*

Proof. Let λ be a candidate set of labels for the edges of G, computed as described in Phase 1 and Phase 2. By Lemma 2, we also assume that, for each edge e of G, $\lambda(e)$ contains at most two labels. Based on Lemma 3, deciding whether G admits an embedding-preserving RU representation is equivalent to deciding whether G admits a non-conflicting labeling $\mathcal{L}(\lambda)$. We model this problem as a 2-SAT problem, which is defined as follows.

For each edge e and for each label $X \in \lambda(e)$, define a Boolean variable b_e^X; this variable will be set to `True` if we select label X for edge e, and it will be set to `False` otherwise. We define a formula $cl(e)$ for every edge e and a formula $cl(v)$ for every vertex v that has at least one incident edge e with $|\lambda(e)| = 2$. Our 2-SAT formula Φ is the conjunction of all the formulas defined for the edges and for the vertices of G.

For each edge e of G we define $cl(e)$ as either the conjunction of two clauses or as a single clause in Φ, depending on whether $|\lambda(e)| = 2$ or $|\lambda(e)| = 1$. More precisely, if $\lambda(e) = \{X, Y\}$ we have $cl(e) = (b_e^X \vee b_e^Y) \wedge (\neg b_e^X \vee \neg b_e^Y)$. This ensures that in order to satisfy Φ we have to select exactly one of the two labels X and Y. If $\lambda(e) = \{X\}$, we have $cl(e) = (b_e^X \vee b_e^X)$. For $cl(v)$ we have two cases: (i) v is a source or a sink; (ii) v is neither a source nor a sink.

Case (i). If v is a source (resp. a sink), let e_1 and e_2 be the two outgoing (resp. incoming) edges from v, respectively. We set:

$$cl(v) = \neg b_{e_1}^U \vee \neg b_{e_2}^U$$

Case (ii). We have four subcases: (a) $\deg(v) = 2$; (b) $\deg(v) = 3$ and v has two incoming edges; (c) $\deg(v) = 3$ and v has two outgoing edges; (d) $\deg(v) = 4$ and v has two incoming and two outgoing edges.

(a) Let e_1 and e_2 be the incoming edge and the outgoing edge of v, respectively. Suppose $\lambda(e_1) = \{U, X\}$ and $\lambda(e_2) = \{U, Y\}$, where $X, Y \in \{R, L\}$. If $X = Y$, we do not add any clause associated with v, because any two labels for e_1 and e_2 in $\lambda(e_1)$ and in $\lambda(e_2)$ are non-conflicting. If $X \neq Y$, we set:

$$cl(v) = \neg b_{e_1}^X \vee \neg b_{e_2}^Y.$$

(b) Let e_1 and e_2 be the two incoming edges of v, in this left-to-right order, and let e_3 be the outgoing edge of v. We have $\lambda(e_1) = \{U, R\}$, $\lambda(e_2) = \{L, U\}$, and either (1) $\lambda(e_3) = \{L, U\}$ or (2) $\lambda(e_3) = \{U, R\}$. We define $cl(v)$ as follows, depending on the two sub-cases:

$$(1)\ cl(v) = (\neg b_{e_1}^U \vee \neg b_{e_2}^U) \wedge (\neg b_{e_1}^R \vee \neg b_{e_3}^L)$$

$$(2)\ cl(v) = (\neg b_{e_1}^U \vee \neg b_{e_2}^U) \wedge (\neg b_{e_2}^L \vee \neg b_{e_3}^R)$$

(c) Symmetric to (b).

(d) Let e_1 and e_2 be the two outgoing edges of v, and let e_3 and e_4 be the two incoming edges of v, in this left-to-right order. We have: $\lambda(e_1) = \{L, U\}$, $\lambda(e_2) = \{U, R\}$; $\lambda(e_3) = \{L, U\}$, $\lambda(e_4) = \{U, R\}$. We define $cl(v)$ as follows:

$$cl(v) = (\neg b_{e_1}^U \vee \neg b_{e_2}^U) \wedge (\neg b_{e_2}^R \vee \neg b_{e_3}^L) \wedge (\neg b_{e_3}^U \vee \neg b_{e_4}^U) \wedge (\neg b_{e_1}^L \vee \neg b_{e_4}^R)$$

Observe that, if a vertex is incident to an edge e with $|\lambda(e)| = 1$, we use the clauses defined above, but in these cases they are simplified since the values b_e^X are fixed (either to True or to False) for any possible value of X. ◀

5 Testing in the Variable Embedding Setting

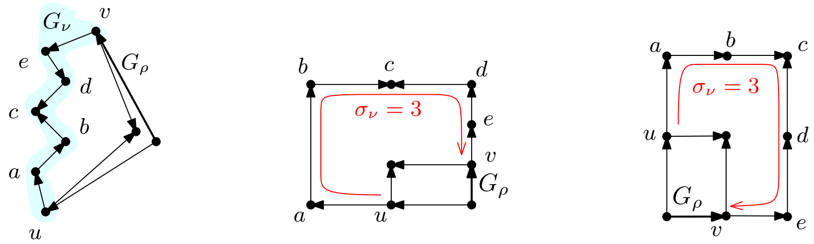
In this section we deal with biconnected planar digraphs whose embedding is not fixed. In Section 5.1 we define the notion of rectilinear-upward spirality. In Section 5.2 we describe a polynomial-time testing algorithm for digraphs whose underlying undiracted graph is series-parallel. In Section 5.3 we consider the general case, and give a FPT algorithm parameterized by the the number of sources and sinks in the digraph.

5.1 Rectilinear-Upward Spirality

We introduce the new concept of rectilinear-upward spirality, which specializes the notion of orthogonal spirality defined in [11]. While the orthogonal spirality is a measure of how much a given subgraph of an undirected graph G is rolled-up in an orthogonal representation of G, our notion of spirality is for directed graphs and incorporates additional information about the sides to which edges are incident to the poles of the triconnected components.

SPQR-trees. As in [11], our definition of spirality exploits the popular SPQR-tree data structure introduced by Di Battista and Tamassia [10]. The *SPQR-tree* T of a biconnected (di)graph G represents the decomposition of G into its triconnected components [33], and it can be computed in linear time [10, 30]. Refer to Figure 8. Each triconnected component corresponds to a non-leaf node ν of T; the triconnected component itself is the *skeleton* of ν and is denoted as $\text{skel}(\nu)$. Node ν can be: *(i)* an *S-node* (series composition), if $\text{skel}(\nu)$ is a simple cycle of length at least three; *(ii)* a *P-node* (parallel composition), if $\text{skel}(\nu)$ is a bundle of at least three parallel edges; *(iii)* an *R-node* (rigid composition), if $\text{skel}(\nu)$ is a triconnected graph. A degree-1 node of T is a *Q-node* and represents a single edge of G. A *real edge* (resp. *virtual edge*) in $\text{skel}(\nu)$ corresponds to a Q-node (resp., to an S-, P-, or R-node) adjacent to ν in T. Let e be a designated edge of G, called the *reference edge* of G, let ρ be the Q-node of T corresponding to e, and let T be rooted at ρ. For any P-, S-, or R-node ν of T distinct from the root child, $\text{skel}(\nu)$ has a virtual edge, called *reference edge* of $\text{skel}(\nu)$ and of ν, associated with a virtual edge in the skeleton of its parent. The reference edge of the root child of T is the edge corresponding to ρ. For every node $\nu \neq \rho$, the *pertinent graph G_ν of ν* is the subgraph of G whose edges correspond to the Q-nodes in the subtree of T rooted at ν. We also say that G_ν is a *component of* G. The pertinent graph G_ρ of the root ρ coincides with the reference edge of G. If H is a rectilinear representation or an RU rectilinear representation of G, its restriction H_ν to G_ν is a *rectilinear component* or an *RU rectilinear component* of H. As in [7, 11, 15, 16, 18, 20, 23], we implicitly assume to work with a *normalized* SPQR-tree, in which every S-node has exactly two children. Every SPQR-tree can be normalized in $O(n)$ time by recursively splitting an S-node with more than two children into multiple S-nodes with two children. If G has n vertices, a normalized SPQR-tree of G still has $O(n)$ nodes.

RU-Spirality. Let G be a biconnected planar digraph and consider an SPQR-tree T of G rooted at a Q-node ρ, corresponding to a reference edge (s, t). Assume for convenience that the vertices of G are labeled with an *st*-numbering [24] of G (see, e.g., Figure 8a). Let H be an orthogonal representation of G with the reference edge $G_\rho = (s, t)$ in the external



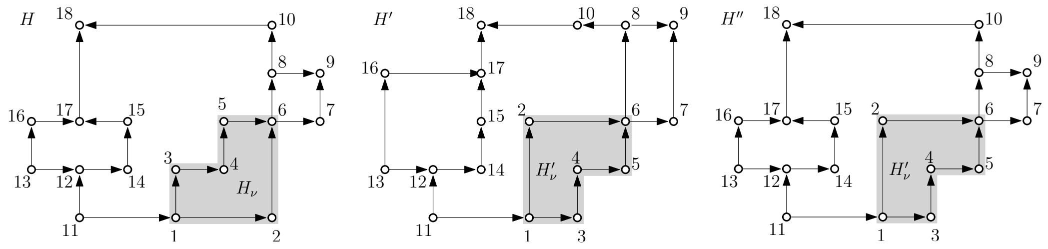
(a) G **(b)** H **(c)** H'

Figure 5 Illustration of the concept of RU-spirality. The two representations in (b) and (c) have the same value of σ_ν but different RU spiralities.

face, let H_ν be a component of H (i.e., the restriction of H to G_ν), and let $\{u, v\}$ be the poles of ν, where u precedes v in the st-numbering. We say that u and v are the *first-pole* and the *second-pole* of ν, respectively. Note that we are not assuming any relationship between the st-numbering and the orientation of the edges of G. For each pole $w \in \{u, v\}$, let $\text{intdeg}_\nu(w)$ and $\text{extdeg}_\nu(w)$ be the degree of w inside and outside H_ν, respectively. We define two (possibly coincident) *alias vertices* of w, denoted by w' and w'', as follows: (i) if $\text{intdeg}_\nu(w) = 1$, then $w' = w'' = w$; (ii) if $\text{intdeg}_\nu(w) = \text{extdeg}_\nu(w) = 2$, then w' and w'' are dummy vertices, each splitting one of the two distinct edge segments incident to w outside H_ν; (iii) if $\text{intdeg}_\nu(w) > 1$ and $\text{extdeg}_\nu(w) = 1$, then $w' = w''$ is a dummy vertex that splits the edge segment incident to w outside H_ν.

Let A^w be the set of distinct alias vertices of a pole w. Let P^{uv} be any simple undirected path from u to v inside H_ν and let $u' \in A^u$ and $v' \in A^v$ be two alias vertices of u and of v, respectively. The path $S^{u'v'}$ obtained concatenating (u', u), P^{uv}, and (v, v') is a *spine* of H_ν. Denote by $n(S^{u'v'})$ the number of right turns minus the number of left turns encountered along $S^{u'v'}$ moving from u' to v'. The *rectilinear spirality* $\sigma(H_\nu)$ of H_ν is either an integer or a semi-integer number, defined based on the following cases: (i) If $A^u = \{u'\}$ and $A^v = \{v'\}$ then $\sigma(H_\nu) = n(S^{u'v'})$. (ii) If $A^u = \{u'\}$ and $A^v = \{v', v''\}$ then $\sigma(H_\nu) = \frac{n(S^{u'v'}) + n(S^{u'v''})}{2}$. (iii) If $A^u = \{u', u''\}$ and $A^v = \{v'\}$ then $\sigma(H_\nu) = \frac{n(S^{u'v'}) + n(S^{u''v'})}{2}$. (iv) If $A^u = \{u', u''\}$ and $A^v = \{v', v''\}$ assume, without loss of generality, that (u, u') succeeds (u, u'') clockwise around u and that (v, v') precedes (v, v'') clockwise around v; then $\sigma(H_\nu) = \frac{n(S^{u'v'}) + n(S^{u''v''})}{2}$.

For brevity, in the following we often denote by σ_ν the rectilinear spirality of an RU representation of G_ν. Let $\{S, N, W, E\}$ denote the set of the four possible sides (North, South, East, West) by which an edge can be incident to a vertex in an RU representation. The *rectilinear-upward spirality* (*RU-spirality* for short) of H_ν, denoted by $\tau(H_\nu)$ (or simply by τ_ν), is a tuple $\langle \sigma_\nu, \varphi_u, \varphi_v \rangle$, where σ_ν is the rectilinear spirality of H_ν and where $\varphi_w = (S_w, N_w, W_w, E_w)$ specifies the arrangement of the internal and external edges of H_ν incident to a pole $w \in \{u, v\}$, with respect to the four sides S (South), N (North), W (West), and E (East). Precisely, for each $D \in \{S, N, W, E\}$ and $w \in \{u, v\}$, we have $D_w \in \{\mathsf{free}, \mathsf{int}, \mathsf{ext}\}$ in such a way that: $D_w = \mathsf{free}$ if no edge is incident to w from side D; $D_w = \mathsf{int}$ if there is an edge of H_ν (i.e., an edge internal to H_ν) incident to w from side D; $D_w = \mathsf{ext}$ if there is an edge of H not in H_ν (i.e., an edge external to H_ν) incident to w from side D. We call σ_ν the *rectilinear spirality* of τ_ν; φ_u and φ_v the *pole side specifications* of τ_ν. Figure 5 shows an illustration of the concept of RU-spirality. In Figure 5a there is a digraph G with a highlighted S-component G_ν with first-pole u and second-pole v.



(a) G **(b)** H **(c)** H'

Figure 6 Illustration of the concept of substitution. The RU representation H'' in (c) is obtained by substituting H_ν with H'_ν in H. The first-pole of ν is vertex 1 and the second-pole of ν is vertex 6. We have $\tau(H_\nu) = \tau(H'_\nu) = \langle -\frac{1}{2}, (\mathsf{free}, \mathsf{int}, \mathsf{ext}, \mathsf{int}), (\mathsf{int}, \mathsf{ext}, \mathsf{int}, \mathsf{ext})\rangle$.

Figure 5b and Figure 5c show two different RU representations H and H' of G. In H we have $\tau_\nu = \langle 3, \varphi_u = (\mathsf{free}, \mathsf{ext}, \mathsf{int}, \mathsf{ext}), \varphi_v = (\mathsf{ext}, \mathsf{int}, \mathsf{ext}, \mathsf{free})\rangle$ and in H' we have $\tau_\nu = \langle 3, \varphi_u = (\mathsf{ext}, \mathsf{int}, \mathsf{free}, \mathsf{ext}), \varphi_v = (\mathsf{free}, \mathsf{ext}, \mathsf{ext}, \mathsf{int})\rangle$.

Note that, denoted by G' the subgraph of G consisting of G_ν plus the external edges incident to the poles of ν, the RU-spirality for an RU representation H_ν of G_ν can also be defined referring to an RU representation of G' that contains H_ν, rather than to an RU representation of G. If we are able to construct an RU representation H' of G' such that its restriction H_ν to G_ν has RU-spirality τ_ν, then we say that G_ν (or simply ν) *admits* RU-spirality τ_ν. Observe that, even if H_ν admits RU-spirality τ_ν, it might not exist an RU representation of G whose restriction to G_ν has spirality τ_ν.

Substituting components with the same RU-spirality. We extend the results in [11, 18] to show that components with the same RU spirality are "interchangeable". Let H and H' be two different RU representations of G with the same reference edge G_ρ on the external face. Also let H_ν and H'_ν be the restrictions of H and H' to the same component G_ν. If $\tau(H_\nu) = \tau(H'_\nu)$, the operation $\mathrm{Sub}(H_\nu, H'_\nu)$ of *substituting* H_ν with H'_ν in H defines a new plane digraph H'' with an angle labeling such that the restriction of H'' to G_ν coincides with H'_ν, while the restriction of H'' to $G \setminus G_\nu$ stays as in H. More formally, let u and v be the first-pole and second-pole of ν, respectively. The external boundary of H_ν contains a left path p_l and a right path p_r, such that p_l (resp. p_r) goes from u to v traversing the external boundary of H_ν clockwise (resp. counterclockwise). Let f_l and f_r be the faces of H outside H_ν and incident to p_l and p_r, respectively. With respect to H'_ν and H', define p'_l, p'_r, f'_l, f'_r analogously. Since $\tau(H_\nu) = \tau(H'_\nu)$, the circular sequence of angles at each pole $w \in \{u, v\}$ is the same in H and in H', namely the angles at w internal and external to G_ν are the same in H and H'. The digraph H'' is defined as follows:

- H'' has the same set of vertices and edges as G.
- The planar embedding of H'' is such that: all the faces of H outside H_ν and distinct from f_l and f_r, as well as all faces of H'_ν, are also faces of H''. Further, H'' has two faces f''_l and f''_r obtained by replacing p_l with p'_l and p_r with p'_r in the boundary of f_l and of f_r, respectively.
- The angle labeling of H'' is such that: (i) all the angles at the vertices of G not belonging to G_ν are those in H; (ii) all the angles at the vertices of G_ν distinct from u and v are those in H'_ν; (iii) for each pole $w \in \{u, v\}$, the internal and external angles at w are defined as in H or, equivalently, as in H' (they are the same as $\tau(H_\nu) = \tau(H'_\nu)$).

The following result proves that H'' is an RU representation.

▶ **Theorem 5.** *Let G be a biconnected planar digraph, T be an SPQR-tree of G with respect to a given reference e, and ν be a non-root node of T. Let H and H' be two different RU representations of G with e on the external face, and let H_ν and H'_ν be the restrictions of H and of H' to G_ν, respectively. If $\tau(H_\nu) = \tau(H'_\nu)$ then the graph H'' defined by $\mathrm{Sub}(H_\nu, H'_\nu)$ is an RU representation of G.*

Proof. The fact that the planar embedding and the labeling of H'' describe a rectilinear planar representation of G is proved in [11, 18], as a consequence that H_ν and H'_ν have the same rectilinear spirality. We now orient H'' in such a way that, for an arbitrarily chosen edge $e = (x, y)$ of H'_ν, the vertices x and y have the same relative positions as in H'_ν. This implies that for each edge e' of G_ν, the relative position of the end-vertices of e' in H'' remains the same as in H'. Also, for each side $\{S, N, W, E\}$ of w, either this side is free in both H and H', or it is occupied either by an edge internal to G_ν or by an edge external to G_ν in both H and H'. This implies that, with the chosen orientation, for each edge e'' of $G \setminus G_\nu$ the relative position of the end-vertices of e'' in H'' is the same as in H. It follows that, with the chosen orientation, no edge of H'' is downward. ◀

Based on Theorem 5, in order to test RU planarity of a biconnected digraph G with a given reference edge on the external face, we exploit a dynamic programming technique that visits a rooted SPQR-tree T of G bottom-up. At each visited node ν of T, and for each RU spirality τ_ν admitted by ν, we store at ν a pair $\langle \tau_\nu, H_\nu \rangle$, where H_ν is just one RU representation of G_ν with spirality τ_ν, called a *representative of* τ_ν. The set of all pairs $\langle \tau_\nu, H_\nu \rangle$ is the *feasible set of* ν and is denoted by Σ_ν. Observe that, if G_ν has n_ν vertices and if $\tau_\nu \in \Sigma_\nu$, the rectilinear spirality σ_ν in τ_ν cannot exceed n_ν, as we can make at most n_ν right or n_ν left turns. Also, for each value σ_ν, the number of RU spiralities τ_ν in Σ_ν with rectilinear spirality σ_ν is bounded by a constant. Hence, we have the following.

▶ **Property 2.** *For any component G_ν with n_ν vertices, $|\Sigma_\nu| = O(n_\nu)$. Also, for each $\tau_\nu \in \Sigma_\nu$, the corresponding rectilinear spirality σ_ν belongs to the interval $[-n_\nu, n_\nu]$.*

5.2 Testing Series-Parallel Digraphs in Polynomial Time

When the SPQR-tree T of a biconnected graph G does not have R-nodes, G is a *series-parallel graph*, or simply an *SP-graph*. Also, T is called the SPQ-tree of G. In this section we assume that G is an *SP-digraph*, i.e., a digraph whose underlying undirected graph is an SP-graph. We also assume that T is normalized. We prove the following lemmas.

▶ **Lemma 6.** *Let ν be a Q-node of T. We can compute Σ_ν in $O(1)$ time.*

Proof. G_ν is a directed edge $e = (u, v)$ of G. In any RU representation of G, edge e is either leftward, or rightward, or upward. For each of these three possibilities, we have to consider the $O(1)$ possible arrangements of the edges incident to e on the different sides of u and v, each of them defining a different spirality τ_ν. Thus Σ_ν is constructed in $O(1)$ time. ◀

▶ **Lemma 7.** *Let ν be an S-node of T with children μ_1 and μ_2, and let n_1^ν and n_2^ν be the number of nodes in G_{μ_1} and G_{μ_2}, respectively. If Σ_{μ_1} and Σ_{μ_2} are given, then we can compute Σ_ν in $O(n_1^\nu \cdot n_2^\nu)$ time.*

Sketch of Proof. For each pair $\tau_{\mu_1} \in \Sigma_{\mu_1}$ and $\tau_{\mu_2} \in \Sigma_{\mu_2}$, let H_{μ_1} and H_{μ_2} be the representatives of τ_{μ_1} and τ_{μ_2}, respectively. Let u_i and v_i be the first-and second-pole of μ_i, respectively, with $i = 1, 2$. Clearly $v_1 = u_2$. Suppose that for each side $D \in \{S, N, W, E\}$

one of these three cases holds: (i) $D_{v_1} = \mathsf{free}$ in τ_{μ_1} and $D_{u_2} = \mathsf{free}$ in τ_{μ_2}; (ii) $D_{v_1} = \mathsf{int}$ in τ_{μ_1} and $D_{u_2} = \mathsf{ext}$ in τ_{μ_2}; (iii) $D_{v_1} = \mathsf{ext}$ in τ_{μ_1} and $D_{u_2} = \mathsf{int}$ in τ_{μ_2}. If so the pole side specifications of τ_{μ_1} and τ_{μ_2} are compatible, and we can construct an RU representation H_ν by gluing together H_{μ_1} and H_{μ_2} at the common pole $v_1 = u_2$; the rectilinear spirality σ_ν in τ_ν is computed based on σ_{μ_1}, σ_{μ_2}, and on the pole side specifications in τ_{μ_1} and τ_{μ_2}. ◀

The next structural lemma, which is proven by induction on the depth of a normalized rooted SPQ-tree T, is given in [17]. Corollary 9 follows by combining Lemma 7 and Lemma 8.

▶ **Lemma 8** ([17]). *Let T be a normalized rooted SPQ-tree of an n-vertex SP-digraph G, and let $\mathcal{S}$ be the set of all S-nodes of T. We have $\sum_{\nu \in \mathcal{S}} n_1^\nu \cdot n_2^\nu = O(n^2)$, where n_1^ν and n_2^ν are the number of vertices in the pertinent graphs of the two children of ν.*

▶ **Corollary 9.** *Let T be a normalized rooted SPQ-tree of an n-vertex SP-digraph G. Assume that T is visited bottom-up and that when we visit a node the feasible sets of its children are known. Then, the feasible sets of all S-nodes of T can be computed in overall $O(n^2)$ time.*

The next lemma is about the feasible sets of P-nodes. Theorem 11 summarizes the main result of this subsection.

▶ **Lemma 10.** *Let ν be a P-node of T with children $\mu_1, \mu_2, \dots \mu_h$ $(h = 2, 3)$. If Σ_{μ_1} and Σ_{μ_2} are given, then we can compute Σ_ν in $O(n)$ time.*

Proof. Let u and v be the first-pole and the second-pole of v, respectively. By definition of P-node, u and v are also the first-pole and the second-pole of each child of ν. Denote by n_ν the number of vertices of G_ν. Suppose first that ν is a P-node with three children μ_1, μ_2, and μ_3. Any planar embedding of G defines a planar embedding of skel(ν). If for simplicity we topologically imagine v above u, in any given embedding of skel(ν) the three children of ν (namely, the edges of skel(ν) that correspond to these children) occur from left to right in some order (equivalently, this order coincides with the circular order in which these children are encountered around v moving counterclockwise from the reference edge of skel(ν)). Suppose that for a given embedding ϕ of skel(ν), we rename the three children of ν as μ_l, μ_c, and μ_r, if they occur in this left-to-right order in ϕ. Let H be any rectilinear representation of G that induces for skel(ν) the embedding ϕ. Also denote by σ_ν, σ_{μ_l}, σ_{μ_c}, and σ_{μ_r} the rectilinear spirality values of the restrictions of H to G_ν, G_{μ_l}, G_{μ_c}, and G_{μ_r}, respectively. It is proved in [11] that the following relationship holds: $\sigma_\nu = \sigma_{\mu_l} - 2 = \sigma_{\mu_c} = \sigma_{\mu_r} + 2$. Clearly, since an RU representation is in particular a rectilinear representation, then the same relationship must be verified for any RU representation that induces the embedding ϕ for skel(ν). Hence, as done in [11], for each candidate rectilinear spirality value $\sigma_\nu \in [-n_\nu, n_\nu]$ (see Property 2) and for each possible embedding ϕ of skel(ν), one can check in $O(1)$ time whether there exist three elements $\tau_{\mu_l} \in \Sigma_{\mu_l}$, $\tau_{\mu_c} \in \Sigma_{\mu_c}$, and $\tau_{\mu_r} \in \Sigma_{\mu_r}$ such that the corresponding rectilinear spiralities σ_{μ_l}, σ_{μ_c}, and σ_{μ_r} satisfy the above relationship. If not, then the target rectilinear spirality value σ_ν is not feasible, otherwise suppose that such elements τ_{μ_l}, τ_{μ_c}, and τ_{μ_r} exist. To check whether we can combine them into an RU spirality τ_ν having rectilinear spirality σ_ν, we have to test the compatibility of the pole side specifications for each pole $w \in \{u, v\}$. This compatibility can be checked with the following simple considerations. Since ν has three children, w has degree four in G. Also, each of the three components G_{μ_l}, G_{μ_c}, and G_{μ_r} contains exactly one edge incident to w, while the fourth edge incident to w is external to all the three components. Hence, to have compatibility, we must have that in the pole specification of each τ_{μ_j} $(j = l, c, r)$ there is exactly one side of w with value int and each other side of w with value ext. In particular, call D_w the side of w in the pole specification of τ_{μ_l} for which $D_w = \mathsf{int}$. To fix the ideas, assume that $D_w = W_w$, i.e., the edge of G_{μ_l} incident w occupies the West side of w (the cases $D_w = S_w$, $D_w = N_w$, and $D_w = E_w$ are treated in a similar way). This means that $\varphi_w = (\mathsf{ext}, \mathsf{ext}, \mathsf{int}, \mathsf{ext})$ in τ_{μ_l}. Thus, in order to have compatibility at w it must be $\varphi_w = (\mathsf{int}, \mathsf{ext}, \mathsf{ext}, \mathsf{ext})$ in τ_{μ_c} and $\varphi_w = (\mathsf{ext}, \mathsf{ext}, \mathsf{ext}, \mathsf{int})$ in τ_{μ_r}. If there is compatibility at the pole u and at the pole v, we can glue together the representatives H_{μ_l}, H_{μ_c}, H_{μ_r} into an RU representation H_ν with rectilinear spirality σ_ν, and we insert $\tau_\nu = \langle \sigma_\nu, H_\nu \rangle$ in Σ_ν; otherwise we discard the triplet $\tau_{\mu_l}, \tau_{\mu_c}, \tau_{\mu_r}$.

The described procedure for constructing Σ_ν takes $O(n)$ time because: (i) by Property 2 there are $O(n)$ possible target rectilinear spirality values to consider; (ii) for each target spirality value, skel(ν) has 6 distinct planar embeddings to consider; (iii) for each embedding of skel(ν) we can check in $O(1)$ time which triplets $\tau_{\mu_l}, \tau_{\mu_c}, \tau_{\mu_r}$ that satisfy the relation $\sigma_\nu = \sigma_{\mu_l} - 2 = \sigma_{\mu_c} = \sigma_{\mu_r} + 2$ (see also [11]), and for each of these triplets we can also check in $O(1)$ time the compatibility of the pole side specifications.

If ν is a P-node with two children, the strategy for constructing Σ_ν is exactly the same. However in this case, skel(ν) has only two embeddings to consider for each target value of rectilinear spirality σ_ν. Also, for each of these two embeddings, the relationship between σ_ν and the rectilinear spiralities of the children of ν, as well as the compatibility conditions for the pole side specifications, may require to analyze more cases, whose number is however still bounded by a constant (see [11] for details about the relationships between a rectilinear representation of ν and those of its two children). ◀

▶ **Theorem 11.** *Let G be an n-vertex SP-digraph. There exists an $O(n^3)$-time algorithm that tests whether G admits an RU representation, and that computes one in the affirmative case.*

Sketch of Proof. Let T be the SPQ-tree of G. For each Q-node ρ of T, the algorithm considers T rooted at ρ and performs a post-order visit of T to tests whether G admits an RU representation with the reference edge G_ρ on the external face. It first computes Σ_ν for each leaf ν of T, that is, for each Q-node of T distinct from ρ. Then, for each internal node ν of T distinct from ρ the algorithm computes Σ_ν by using the feasible sets of the children of ν, by means of Lemma 7 or of Lemma 10 depending on whether ν is an S-node or a P-node. If Σ_ν is empty then G does not have an RU representation with G_ρ on the external face, and the algorithm starts visiting T rooted at another Q-node. Suppose vice versa that the algorithm achieves the root child ν and that Σ_ν is not empty. The algorithm checks if there is $\tau_\nu \in \Sigma_\nu$ whose H_ν can be glued together with a straight-line representation of the reference edge G_ρ, which is oriented either upward, or leftward, or rightward.

Regarding the time complexity, the algorithm has to test $O(n)$ rooted SPQ-trees. For each tree, the feasible sets of all Q-nodes can be computed in overall $O(n)$ time by Lemma 6, those of all S-nodes can be computed in $O(n^2)$ time by Corollary 9, and those of all P-nodes can be computed in $O(n^2)$ time by Lemma 10. Finally, the condition at the root can be checked in $O(n)$ time. Hence, the whole algorithm can be executed in $O(n^3)$ time. ◀

5.3 FPT Testing Algorithm by the Number of Sources and Sinks

Let G be a biconnected digraph. A vertex of G that is either a source or a sink of G is called a *switch* [10]. We sketch the description of an FPT algorithm for Rectilinear-Upward Planarity Testing parameterized by the number k of switches of G.

Let T be a rooted SPQR-tree of G and let ν be any node of T. It can be shown that: (i) if G_ν does not contain any switches of G, then it can only admit a constant number of rectilinear spirality values, and hence a constant number of RU spiralities; (ii) otherwise, the possible values of rectilinear spirality admitted by G_ν is a linear function of k.

The FPT algorithm extends the dynamic programming of Section 5.2 so to handle R-nodes. When an R-node ν of a rooted SPQR-tree is visited, we consider the two possible planar embeddings of its skeleton, and for each of these two embeddings we consider every possible upward planar embedding and every possible target RU spirality τ_ν; then, we test whether G_ν admits τ_ν. If so, as for S-nodes and P-nodes, we construct an RU representation H_ν and we insert $\langle \tau_\nu, H_\nu \rangle$ in Σ_ν; otherwise, τ_ν is discarded. To perform the test for each target value τ_ν, we partition the children of ν into two sets A and B. Namely, a child μ of ν is inserted in A if G_μ contains at least one switch of G, otherwise we insert μ in B. Clearly $|A| = O(k)$, while for each element in B the size of the feasible set is constant. Then, for each combination of fixed RU spiralities in the feasible sets of the elements in A, we solve a constrained RU planarity testing problem that: (a) forces G_ν to have the target rectilinear spirality σ_ν (associated with τ_ν); (b) preserves the chosen combination of RU spiralities for the elements of A; and (c) guarantees that the pertinent graphs of the nodes in B have one of the constantly many RU spiralities in their feasible sets. We prove that this test can be executed in $O(n)$ time by using the 2-SAT model of Section 4, enriched with $O(n)$ number of constraints. Since there are $O(k^k) = 2^{O(k \log k)}$ combinations of RU spiralities for the elements in A, and since there are $O(4^k) = O(2^{2k})$ upward planar embeddings for each of the two possible planar embeddings of an R-node, we get the following.

▶ **Lemma 12.** *Let ν be an R-node of T and let $\mu_1, \dots, \mu_h$ be its children. Given the feasible set Σ_{μ_i} for each $i \in \{1, \dots, h\}$, we can compute Σ_ν in $2^{O(k \cdot \log k + 2k)} \cdot O(n)$ time.*

By Lemma 12, for processing all R-nodes of T we spend in total $2^{O(k \cdot \log k + 2k)} \cdot O(n^2)$. For an S-node or a P-node ν, we use exactly the same strategy as in Section 5.2. However, since the sizes of the feasible sets of all children of ν, and of ν itself, are now $O(k)$, Σ_ν can be constructed in $O(k^2)$ time if ν is an S-node and in time $O(k)$ if ν is a P-node; hence we spend $O(nk^2)$ for processing all S- and P-nodes of T. Finally, since every RU representation of G has at least one source and one sink in its external face, it suffices to test $O(k)$ possible rooted SPQR-trees, thus saving the extra $O(n)$ factor of Theorem 11. The following holds.

▶ **Theorem 13.** *Let G be a planar digraph with k switches. There exists an $2^{O(k \cdot \log k + 2k)} \cdot O(n^2)$-time algorithm that tests whether G is rectilinear-upward planar and that computes an RU representation of G in the positive case.*

A byproduct of the previous theorem is the following corollary for the well-known family of *st*-digraphs, i.e., digraphs with a single source and a single sink.

▶ **Corollary 14.** *The Rectilinear-Upward Planarity Testing problem can be solved in $O(n^2)$ time for planar st-digraphs with n vertices.*

6 Open Problems

The NP-hardness of Rectilinear-Upward Planarity Testing holds when the embedding can vary while the linear-time solution holds for upward plane digraphs. Also the testing is trivial if a rectilinear embedding is given. Establishing the complexity of the problem when a planar embedding (neither rectilinear nor upward) is fixed remains an open question. Moreover, our results in the variable embedding setting consider biconnected graphs; extending these results to simply connected instances is a topic for future exploration. Lastly, there is potential for future research in improving the time complexity for series-parallel digraphs.

References

1 Sarmad Abbasi, Patrick Healy, and Aimal Rextin. Improving the running time of embedded upward planarity testing. *Inf. Process. Lett.*, 110(7):274–278, 2010. doi:10.1016/j.ipl.2010.02.004.

2 Giuseppe Di Battista and Roberto Tamassia. Algorithms for plane representations of acyclic digraphs. *Theor. Comput. Sci.*, 61:175–198, 1988. doi:10.1016/0304-3975(88)90123-5.

3 Paola Bertolazzi, Giuseppe Di Battista, Giuseppe Liotta, and Carlo Mannino. Upward drawings of triconnected digraphs. *Algorithmica*, 12(6):476–497, 1994. doi:10.1007/BF01188716.

4 Paola Bertolazzi, Giuseppe Di Battista, Carlo Mannino, and Roberto Tamassia. Optimal upward planarity testing of single-source digraphs. *SIAM J. Comput.*, 27(1):132–169, 1998. doi:10.1137/S0097539794279626.

5 Carla Binucci, Walter Didimo, and Maurizio Patrignani. Upward and quasi-upward planarity testing of embedded mixed graphs. *Theor. Comput. Sci.*, 526:75–89, 2014. doi:10.1016/j.tcs.2014.01.015.

6 Franz-Josef Brandenburg, David Eppstein, Michael T. Goodrich, Stephen G. Kobourov, Giuseppe Liotta, and Petra Mutzel. Selected open problems in graph drawing. In Giuseppe Liotta, editor, *Graph Drawing, 11th International Symposium, GD 2003, Perugia, Italy, September 21-24, 2003, Proceedings*, volume 2912 of *Lecture Notes in Computer Science*, pages 515–539. Springer, 2003. doi:10.1007/978-3-540-24595-7_55.

7 Steven Chaplick, Emilio Di Giacomo, Fabrizio Frati, Robert Ganian, Chrysanthi N. Raftopoulou, and Kirill Simonov. Parameterized algorithms for upward planarity. In Xavier Goaoc and Michael Kerber, editors, *38th International Symposium on Computational Geometry, SoCG 2022, June 7-10, 2022, Berlin, Germany*, volume 224 of *LIPIcs*, pages 26:1–26:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.SoCG.2022.26.

8 Steven Chaplick, Emilio Di Giacomo, Fabrizio Frati, Robert Ganian, Chrysanthi N. Raftopoulou, and Kirill Simonov. Testing upward planarity of partial 2-trees. In Patrizio Angelini and Reinhard von Hanxleden, editors, *Graph Drawing and Network Visualization – 30th International Symposium, GD 2022, Tokyo, Japan, September 13-16, 2022, Proceedings*, volume 13764 of *Lecture Notes in Computer Science*, pages 175–187. Springer, 2022. doi:10.1007/978-3-031-22203-0_13.

9 Sabine Cornelsen and Andreas Karrenbauer. Accelerated bend minimization. *J. Graph Algorithms Appl.*, 16(3):635–650, 2012. doi:10.7155/jgaa.00265.

10 Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs.* Prentice-Hall, 1999.

11 Giuseppe Di Battista, Giuseppe Liotta, and Francesco Vargiu. Spirality and optimal orthogonal drawings. *SIAM J. Comput.*, 27(6):1764–1811, 1998. doi:10.1137/S0097539794262847.

12 Emilio Di Giacomo, Giuseppe Liotta, and Fabrizio Montecchiani. Sketched representations and orthogonal planarity of bounded treewidth graphs. In Daniel Archambault and Csaba D. Tóth, editors, *Graph Drawing and Network Visualization – 27th International Symposium, GD 2019, Prague, Czech Republic, September 17-20, 2019, Proceedings*, volume 11904 of *Lecture Notes in Computer Science*, pages 379–392. Springer, 2019. doi:10.1007/978-3-030-35802-0_29.

13 Emilio Di Giacomo, Giuseppe Liotta, and Fabrizio Montecchiani. Orthogonal planarity testing of bounded treewidth graphs. *J. Comput. Syst. Sci.*, 125:129–148, 2022. doi:10.1016/j.jcss.2021.11.004.

14 Walter Didimo, Emilio Di Giacomo, Giuseppe Liotta, Fabrizio Montecchiani, and Giacomo Ortali. On the parameterized complexity of bend-minimum orthogonal planarity. *CoRR*, abs/2308.13665, 2023. doi:10.48550/arXiv.2308.13665.

15 Walter Didimo, Francesco Giordano, and Giuseppe Liotta. Upward spirality and upward planarity testing. *SIAM J. Discret. Math.*, 23(4):1842–1899, 2009. doi:10.1137/070696854.

16 Walter Didimo, Michael Kaufmann, Giuseppe Liotta, and Giacomo Ortali. Rectilinear planarity of partial 2-trees. In Patrizio Angelini and Reinhard von Hanxleden, editors, *Graph Drawing and Network Visualization – 30th International Symposium, GD 2022, Tokyo, Japan, September 13-16, 2022, Proceedings*, volume 13764 of *Lecture Notes in Computer Science*, pages 157–172. Springer, 2022. doi:10.1007/978-3-031-22203-0_12.

17 Walter Didimo, Michael Kaufmann, Giuseppe Liotta, and Giacomo Ortali. Rectilinear planarity of partial 2-trees. *CoRR*, abs/2208.12558, 2022. doi:10.48550/arXiv.2208.12558.

18 Walter Didimo, Michael Kaufmann, Giuseppe Liotta, and Giacomo Ortali. Computing bend-minimum orthogonal drawings of plane series–parallel graphs in linear time. *Algorithmica*, 2023. doi:10.1007/s00453-023-01110-6.

19 Walter Didimo, Michael Kaufmann, Giuseppe Liotta, and Giacomo Ortali. Rectilinear planarity of partial 2-trees. *J. of Graph Algorithms Appl., special issue on GD 2022.*, 2023. to appear.

20 Walter Didimo and Giuseppe Liotta. Computing orthogonal drawings in a variable embedding setting. In Kyung-Yong Chwa and Oscar H. Ibarra, editors, *Algorithms and Computation, 9th International Symposium, ISAAC '98, Taejon, Korea, December 14-16, 1998, Proceedings*, volume 1533 of *Lecture Notes in Computer Science*, pages 79–88. Springer, 1998. doi:10.1007/3-540-49381-6_10.

21 Walter Didimo and Giuseppe Liotta. *Graph Visualization and Data Mining*, chapter 3, pages 35–63. John Wiley & Sons, Ltd, 2006. doi:10.1002/9780470073049.ch3.

22 Walter Didimo, Giuseppe Liotta, Giacomo Ortali, and Maurizio Patrignani. Optimal orthogonal drawings of planar 3-graphs in linear time. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 806–825. SIAM, 2020. doi:10.1137/1.9781611975994.49.

23 Walter Didimo, Giuseppe Liotta, and Maurizio Patrignani. HV-planarity: Algorithms and complexity. *J. Comput. Syst. Sci.*, 99:72–90, 2019. doi:10.1016/j.jcss.2018.08.003.

24 Shimon Even and Robert Endre Tarjan. Corrigendum: Computing an *st*-numbering. TCS 2(1976):339-344. *Theor. Comput. Sci.*, 4(1):123, 1977.

25 Uli Fossmeier and Michael Kaufmann. An approach to bend-minimal upward drawing. In *Graph Drawing, 1th International Symposium, GD 1993, Paris, France*, pages 27–29, 1993.

26 Ulrich Fößmeier and Michael Kaufmann. On bend-minimum orthogonal upward drawing of directed planar graphs. In Roberto Tamassia and Ioannis G. Tollis, editors, *Graph Drawing, DIMACS International Workshop, GD '94, Princeton, New Jersey, USA, October 10-12, 1994, Proceedings*, volume 894 of *Lecture Notes in Computer Science*, pages 52–63. Springer, 1994. doi:10.1007/3-540-58950-3_356.

27 Fabrizio Frati. Planar rectilinear drawings of outerplanar graphs in linear time. *Comput. Geom.*, 103:101854, 2022. doi:10.1016/j.comgeo.2021.101854.

28 Ashim Garg and Roberto Tamassia. A new minimum cost flow algorithm with applications to graph drawing. In Stephen C. North, editor, *Graph Drawing, Symposium on Graph Drawing, GD '96, Berkeley, California, USA, September 18-20, Proceedings*, volume 1190 of *Lecture Notes in Computer Science*, pages 201–216. Springer, 1996. doi:10.1007/3-540-62495-3_49.

29 Ashim Garg and Roberto Tamassia. On the computational complexity of upward and rectilinear planarity testing. *SIAM J. Comput.*, 31(2):601–625, 2001. doi:10.1137/S0097539794277123.

30 Carsten Gutwenger and Petra Mutzel. A linear time implementation of spqr-trees. In Joe Marks, editor, *Graph Drawing, 8th International Symposium, GD 2000, Colonial Williamsburg, VA, USA, September 20-23, 2000, Proceedings*, volume 1984 of *Lecture Notes in Computer Science*, pages 77–90. Springer, 2000. doi:10.1007/3-540-44541-2_8.

31 Md. Manzurul Hasan and Md. Saidur Rahman. No-bend orthogonal drawings and no-bend orthogonally convex drawings of planar graphs (extended abstract). In Ding-Zhu Du, Zhenhua Duan, and Cong Tian, editors, *Computing and Combinatorics – 25th International Conference, COCOON 2019, Xi'an, China, July 29-31, 2019, Proceedings*, volume 11653 of *Lecture Notes in Computer Science*, pages 254–265. Springer, 2019. doi:10.1007/978-3-030-26176-4_21.

32 Patrick Healy and Karol Lynch. Two fixed-parameter tractable algorithms for testing upward planarity. *Int. J. Found. Comput. Sci.*, 17(5):1095–1114, 2006. doi:10.1142/S0129054106004285.

33 John E. Hopcroft and Robert Endre Tarjan. Dividing a graph into triconnected components. *SIAM J. Comput.*, 2(3):135–158, 1973. doi:10.1137/0202012.

34 Michael D. Hutton and Anna Lubiw. Upward planning of single-source acyclic digraphs. *SIAM J. Comput.*, 25(2):291–311, 1996. `doi:10.1137/S0097539792235906`.

35 Bart M. P. Jansen, Liana Khazaliya, Philipp Kindermann, Giuseppe Liotta, Fabrizio Montecchiani, and Kirill Simonov. Upward and orthogonal planarity are W[1]-hard parameterized by treewidth. *CoRR*, abs/2309.01264, 2023. `doi:10.48550/arXiv.2309.01264`.

36 Michael Jünger and Petra Mutzel, editors. *Graph Drawing Software*. Springer, 2004. `doi:10.1007/978-3-642-18638-7`.

37 M. R. Krom. The decision problem for a class of first-order formulas in which all disjunctions are binary. *Mathematical Logic Quarterly*, 13(1-2):15–20, 1967. `doi:10.1002/malq.19670130104`.

38 Takao Nishizeki and Md. Saidur Rahman. *Planar Graph Drawing*, volume 12 of *Lecture Notes Series on Computing*. World Scientific, 2004. `doi:10.1142/5648`.

39 Maurizio Patrignani. Planarity testing and embedding. In Roberto Tamassia, editor, *Handbook on Graph Drawing and Visualization*, pages 1–42. Chapman and Hall/CRC, 2013.

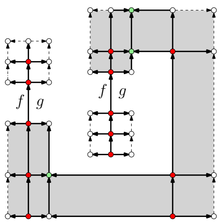
40 Md. Saidur Rahman, Noritsugu Egi, and Takao Nishizeki. No-bend orthogonal drawings of subdivisions of planar triconnected cubic graphs. *IEICE Trans. Inf. Syst.*, 88-D(1):23–30, 2005. URL: `http://search.ieice.org/bin/summary.php?id=e88-d_1_23&category=D&year=2005&lang=E&abst=`.

41 Md. Saidur Rahman, Takao Nishizeki, and Mahmuda Naznin. Orthogonal drawings of plane graphs without bends. *J. Graph Algorithms Appl.*, 7(4):335–362, 2003. `doi:10.7155/jgaa.00074`.

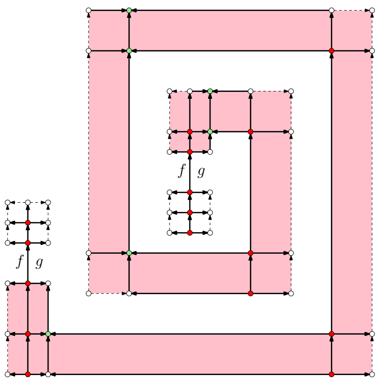
42 Roberto Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM J. Comput.*, 16(3):421–444, 1987. `doi:10.1137/0216030`.

43 Roberto Tamassia and Giuseppe Liotta. Graph drawing. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry, Second Edition*, pages 1163–1185. Chapman and Hall/CRC, 2004. `doi:10.1201/9781420035315.ch52`.

A Appendix



(a) Tentril T_1.



(b) Tendril T_2.

Figure 7 Tendrils T_1 (a) and T_2 (b). Red vertices have three outgoing edges. Green vertices have three incoming edges. Solid edges are incident either to a red or to a green vertex (or both).

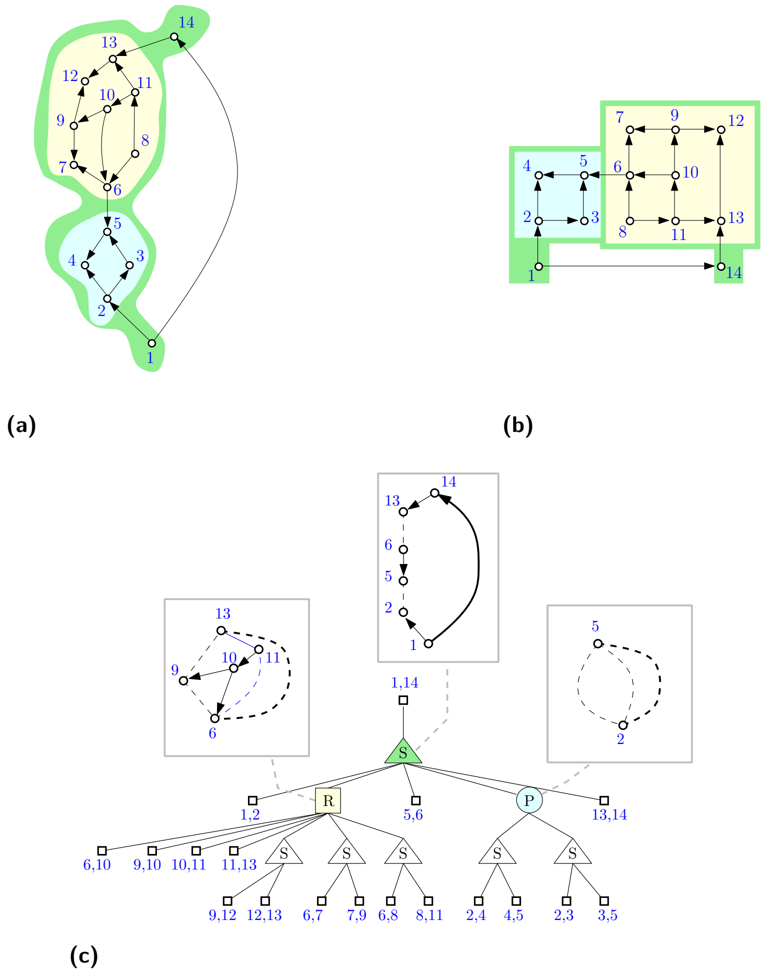


Figure 8 (a) A digraph G with three highlighted components (an S-, an R- and a P-component); (b) an RU representation of G. (c) The SPQR-tree of G with reference edge $(1, 14)$; the skeletons of the highlighted components are shown: dashed edges are virtual and the reference edge is thicker. Q-nodes are labeled with the end-vertices of their corresponding edges.