Rolling Upward Planarity Testing of Strongly Connected Graphs

Christopher Auer, Christian Bachmaier,
Franz J. Brandenburg, and Kathrin Hanauer

University of Passau, Germany
{auerc|bachmaier|brandenb|hanauer}@fim.uni-passau.de

Abstract. A graph is upward planar if it can be drawn without edge crossings such that all edges point upward. Upward planar graphs have been studied on the plane, the standing and rolling cylinders. For all these surfaces, the respective decision problem $\mathcal{NP}$-hard in general. Efficient algorithms exist if the graph contains a single source and a single sink, but only for the plane and standing cylinder
Here we show that there is a linear-time algorithm to test whether a strongly connected graph is upward planar on the rolling cylinder. For our algorithm, we introduce dual and directed SPQR-trees as extensions of SPQR-trees.

1 Introduction

A directed graph is upward planar (**UP**) if it has drawing without edge crossings such that all edge curves monotonically increase in y-direction. Upward planar graphs are of interesting in their own, but also arise in the context of the Sugiyama framework [22], which is the common drawing method for directed graphs. In the Sugiyama framework, the graphs are visualized as hierarchies, where all edges point upward, and it works particularly well for acyclic graphs. If the graph contains cycles, the Sugiyama algorithm is extended to recurrent hierarchies [3], where rolling upward planar (**RUP**) graphs naturally arise in the case of planarity.

The upward direction of the edges induces essential differences between planar and upward planar graphs. For instance, for planar graphs all surfaces of genus 0 are equivalent, such as the plane, the sphere, and the standing and rolling cylinders. The situation is different with upward planarity: There are directed graphs that have no upward drawing in the plane but on the surface of the standing cylinder on which edges may wind around the backside. In fact, there is a strict hierarchy of graph classes upward planar in the plane, on the (truncated) sphere [10, 11, 16, 17] or on the standing cylinder [6, 14, 19, 20, 23], and the rolling cylinder [1, 6]. Moreover, there are several linear-time planarity test algorithms, whereas upward planarity is $\mathcal{NP}$-complete in general [12].

In [2], we provide an overview on different types of upward planarity and investigate their relationships. There, we use the fundamental polygon representation: Let $I = (-1, 1)$ be the open interval from -1 to 1. The fundamental

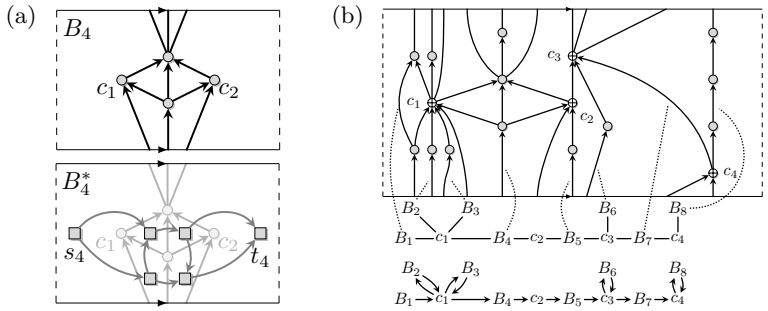

Fig. 1: **RUP** compounds and the (directed) block cut tree

polygon of the plane is $I \times I$, which is (the interior of) a square. By identifying its left and right (top and bottom) sides, we get the *standing* (*rolling*) cylinder. Fig. 1(a) top shows the fundamental polygon of the rolling cylinder, where the arrows at the bottom and top indicate their identification. A graph is *standing upward planar (**SUP**)* (*rolling upward planar (**RUP**)*) if it has a plane upward drawing on the standing (rolling cylinder). In both cases, the edge curves may wind around the cylinder and reappear at the identified sides of the fundamental polygon. An example of a **RUP** graph is shown in Fig. 1(a) at the top. Note that **RUP** graphs may contain cycles whereas **SUP** graphs are acyclic.

Acyclic dipoles are an important tool to study upward planar graphs on the plane and on cylinders. An acyclic dipole has a single source s and a single sink t and no cycles. A graph is **SUP** if and only if it can be augmented to an acyclic dipole by adding edges such that planarity is preserved [14,17,19]. If additionally the edge (s,t) can be added without destroying planarity, we obtain st-graphs, which characterize **UP** [8,18]. In [1], we use acyclic dipoles to characterize **RUP** graphs by means of their duals and investigate strongly connected graphs with at least one edge, which we call *compounds*. A compound is **RUP** if and only if its directed dual is an acyclic dipole (see bottom of Fig. 1(a)).

The **SUP** decision problem is $\mathcal{NP}$-complete [16]. In contrast, an acyclic dipole is **SUP** if and only if it is planar, which can be checked efficiently, and there is an efficient algorithm for triconnected single-source graphs [10]. Hence, the situations for **SUP** and **UP** are similar, as the latter is also efficiently solvable for single-source graphs [5]. The situation for **RUP** is alike: The general decision problem is $\mathcal{NP}$-complete [7] and in this paper we present an efficient algorithm for compounds that utilizes the dipole structure of the compound's dual. We divide the problem into two parts. First, we derive a characterization of **RUP** compounds by means of their block-cut trees which also yields a decision algorithm (Sect. 3). In the second part (Sect. 4), we tackle the blocks, i. e., the biconnected components, by using SPQR-trees to decide if a block is **RUP**. We conclude in Sect. 5.

2 Preliminaries

We consider connected, planar, directed graphs $G = (V, E)$ with vertices V and edges E. A plane *drawing* of G maps the vertices to distinct points and the edges to non-intersecting Jordan curves in the plane. A plane drawing induces an embedding and, equivalently, a rotation system which is the cyclic counterclockwise ordering of the edges at every vertex. An embedding is **RUP** if it is obtained from a **RUP** drawing. An embedding of G specifies faces and defines the (directed) dual graph $G^* = (F, E^*)$ [4]. The vertices of G^* are the faces as defined in the embedding. To avoid confusion, we call the elements of V vertices and the elements F faces. There is a one-to-one correspondence between the edges of the *primal* G and the edges of its dual G^*. For each edge e in G, there is an edge e^* between faces f and f' in G^* if and only if e separates f and f'. If e is directed, edge e^* is oriented such that it points from the face left of e to the face right of e, when traversing the edge curve of e in its direction. Fig. 1(a) top shows a directed graph with its dual at the bottom. A face f is enclosed by edges at its *boundary* and by vertices which are *incident* to f. Note that G^* is always connected and G^* is acyclic if G is strongly connected. Whenever a dual graph G^* is given, then the primal is assumed to be embedded accordingly. Moreover, G^* can be computed from G's embedding in linear time.

In [1], we use directed duals to characterize **RUP** graphs.

Proposition 1 ([1]). *An embedded compound is **RUP** if and only if its dual is an acyclic dipole.*

Hence, we can efficiently decide whether an embeddig is **RUP** by testing if its dual is an acyclic dipole.

Corollary 1. *There is a linear-time algorithm to test whether the embedding of a compound is **RUP**.*

The embedding of a triconnected graph is unique up to *flipping*, i. e., inversion of all cyclic orderings in the rotation system.

Corollary 2. *There is a linear-time algorithm to test if a triconnected compound is **RUP**.*

Hence, deciding whether a compound is **RUP** can be done efficiently if the embedding is fixed. If no embedding is given, we use block-cut and SPQR-trees to find a **RUP** embedding. For a connected graph $G = (V, E)$, a block B is a subgraph induced by a set of edges such that no biconnected subgraph of G properly contains B. Two distinct blocks $B_i = (V_i, E_i)$ and $B_j = (V_j, E_j)$ may share a *cut vertex* $c \in V_i \cap V_j$. The *block-cut tree* $\mathcal{T}_B = (\mathcal{B}, \mathcal{C}, \mathcal{E}_B)$ is a tree, where $\mathcal{B} = (B_1, \ldots, B_k)$ $(k \geq 1)$ are the blocks, $\mathcal{C}$ are the cut vertices and there is an edge $\{B_i, c\} \in \mathcal{E}_B$ if c is in block B_i. Observe that a connected graph is strongly connected if and only if every block is strongly connected.

In the following, we call a directed path *dipath* and an undirected path simply *path*, e. g., of the underlying undirected graph. A dipath (path) with coinciding start and end is a *cycle* (*circle*). Any of these is called *simple* if it contains no vertex/edge twice.

3 Directed Block-Cut Trees of RUP-Compounds

First, we investigate the block-cut trees of **RUP** compounds and define the directed block-cut tree which we us to find a **RUP** embedding of a compound. As an example, consider the compound G in Fig. 1(b) consisting of blocks $B_1, \ldots, B_8$ connected by cut vertices $c_1, \ldots, c_4$, which are displayed by $\oplus$. A block B_i of a compound is strongly connected. Hence, its dual B_i^* is an acyclic dipole with source s_i and sink t_i (see also Fig. 1(a)). Source s_i is a face enclosed by the leftmost cycle C_i^l in the embedding of B_i and the vertices in C_i^l are incident to s_i. Accordingly, t_i corresponds to the rightmost cycle in B_i's embedding. A block contains at least one cycle that winds around the cylinder and, thus, divides it into two halves. Hence, a cut vertex shared by two blocks must be incident to the source or sink of the blocks' duals. For instance in Fig. 1(b), cut vertex c_1 is located at the rightmost cycle of block B_1 and at the leftmost cycle of B_2.

Lemma 1. *Let B_i and B_j be two blocks of a **RUP**-embedded compound sharing cut vertex c. Then, c is either incident to both s_i and t_j or to both s_j and t_i.*

A **RUP** embedding of a block B_i is *feasible* if each of the block's cut vertices is incident to the source or sink in B_i^*. Note that a cut vertex can be incident to both the source and the sink, e. g., c_1 in Fig. 1(b). By Lemma 1, a necessary condition for an embedding to be **RUP** is that each block's embedding is feasible.

Below the fundamental polygon in Fig. 1(b), the block-cut tree $\mathcal{T}_B$ of compound G is displayed. $\mathcal{T}_B$ has a linear structure in the following sense: The cut vertices and all blocks which contain two cut vertices, i. e., B_4, B_5, and B_7, form a path called *spine*. All other blocks contain only one cut vertex and "group" around the cut vertices on the spine. In fact, after removing all blocks with degree one from the directed block-cut tree only the spine remains and, hence, the block-cut tree of a **RUP** compound is a *caterpillar* [15], i. e., a tree where the removal of all leaves yields a path:

Lemma 2. *The block-cut tree of a **RUP** graph is a caterpillar.*

From Lemma 2, we immediately obtain that a block contains at most two cut vertices. We now define the *directed block-cut tree*, which describes the embedding of a compound with respect to its blocks and cut vertices. For the definition, we assume that each block is feasibly **RUP**-embedded and that the block-cut tree is a caterpillar. We obtain a total order $c_1, \ldots, c_l$ on the set of cut vertices by traversing the spine of the caterpillar in either direction. The *directed block-cut tree* $\overrightarrow{\mathcal{T}_B} = (\mathcal{B}, \mathcal{C}, \overrightarrow{E}_B)$ contains all vertices of the original block-cut tree (see the bottom of Fig. 1(b)). Let B_i be a block which contains only cut vertex c_j. If c_j is incident to the source of B_i^*, we add edge (B_i, c_j) to $\overrightarrow{E}_B$ and if c_j is incident to the sink, we get edge (c_j, B_i). Let B_i be a block on the spine containing cut vertices c_j and c_{j+1}. There is an edge $(c_j, B_i) \in \overrightarrow{E}_B$ if c_j is incident to the source of B_i^* and $(B_i, c_{j+1}) \in \overrightarrow{E}_B$ if c_{j+1} is incident to the sink.

In the **RUP** embedding of B_4, c_1 is located at the leftmost cycle whereas c_2 is located at its rightmost cycle (see Fig. 1(a)). Hence, block B_4 has an incoming

edge from c_1 and an outgoing edge to c_2. Note that $\overrightarrow{\mathcal{T}_B}$ is technically not a tree since it may contain antiparallel edge pairs whenever a cut vertex is incident to the source and sink of a block's dual. However, it still has a tree structure. Observe that the in- and outdegree of a block is at most one. Moreover, all "inner" cut vertices, i. e., c_2 and c_3, have the same in- and outdegree. Consider, for instance, c_3 in Fig. 1(b), which is attached to the right side of B_5, to both sides of B_6, and to the left side of B_7. Also, the indegree of c_1 equals its outdegree since B_1 is closing off its left side. Similarly, all "inner" blocks have equal in- and outdegree, i. e., blocks $B_2, \ldots, B_8$. Hence, the directed block cut tree has an Eulerian dipath $B_1, c_1, B_2, c_1, B_4, c_1, B_4, \ldots, c_4$. This observation holds in general and yields a characterization:

Lemma 3. *A compound is **RUP** if and only if every block has a feasible **RUP** embedding such that the directed block cut tree has an Eulerian dipath.*

If the directed block-cut tree has an Eulerian dipath, then this dipath visits each block exactly once and, hence, defines a total order on the blocks. The blocks can then be attached in order to each other at their shared cut vertices to obtain a **RUP** embedding for the whole compound.

Algorithm 1 returns a **RUP** embedding of a compound G or `false` if the compound is not **RUP**. As planarity is a necessary condition for a graph to be **RUP**, we assume that the compound is planar. The algorithm uses the linear-time subroutine TestRUPBiconnected, which is the topic of Sect. 4. TestRUPBiconnected(B_i, V_l, V_r) returns an embedding of block B_i such that all vertices in V_l and V_r are incident to the source and sink of B_i^*, respectively; if no such embedding exists, it returns `false`. First, Algorithm 1 checks whether G is biconnected. If this is the case, it directly calls TestRUPBiconnected. Otherwise, it checks whether the block-cut tree $\mathcal{T}_B$ is a caterpillar. The spine of the caterpillar induces the total order $c_1, \ldots, c_l$ on the set of cut vertices. In the remainder of the algorithm, the directed block-cut tree $\overrightarrow{\mathcal{T}_B}$ is derived by testing the **RUP** embeddability of each block. Simultaneously, it stores the start and the end of the Eulerian dipath of $\overrightarrow{\mathcal{T}_B}$ in $\mathrm{ep}_{\mathrm{Start}}$ and $\mathrm{ep}_{\mathrm{End}}$, respectively, with initial values $\mathrm{ep}_{\mathrm{Start}} = c_1$ and $\mathrm{ep}_{\mathrm{End}} = c_l$. If there is a block B_i containing c_1 that has no embedding such that c_1 is incident to both the source and sink of B_i^*, B_i must have an embedding such that c_1 is incident to the sink of B_i^* (e. g., B_1 in Fig. 1(b)). Hence, B_i is the start of the Eulerian dipath and $\mathrm{ep}_{\mathrm{Start}}$ is set to B_i. Note that no second block with the same properties can exist if the graph is **RUP** since otherwise $\overrightarrow{\mathcal{T}_B}$ has no Eulerian dipath. Likewise, the end of the Eulerian dipath is obtained. Finally, Algorithm 1 traverses the Eulerian dipath from $\mathrm{ep}_{\mathrm{Start}}$ to $\mathrm{ep}_{\mathrm{End}}$, which induces a total order on the set of blocks, and assembles a **RUP** embedding of G. Note that TestRUPBiconnected is called at most twice for each block and that the block-cut tree can be calculated in time $\mathcal{O}(|V| + |E|) = \mathcal{O}(|V|)$. The embeddings of all blocks can be merged in $\mathcal{O}(|V|)$. Also, all other steps have a running time linear in the size of either G or the block-cut tree. Hence, the overall running time of Algorithm 1 is $\mathcal{O}(|V|)$.

Algorithm 1: TestRUPCompound

Input: Planar compound $G = (V, E)$
Output: **RUP** embedding of G or `false` if G is not **RUP**

if G *is biconnected* **then return** TestRUPBiconnected$(G, \emptyset, \emptyset)$
$\mathcal{T}_B = (\mathcal{B}, \mathcal{C}, \mathcal{E}_B) \leftarrow$ BlockCutTree(G)
if $\mathcal{T}_B$ *is no caterpillar* **then return** `false`
Order cut vertices from c_1 to c_l according to the spine of $\mathcal{T}_B$
$\overrightarrow{\mathcal{T}_B} = (\mathcal{B}, \mathcal{C}, \overrightarrow{E}_B)$ with $\overrightarrow{E}_B = \emptyset$
$\text{ep}_{\text{Start}} \leftarrow c_1$; $\text{ep}_{\text{End}} \leftarrow c_l$
foreach $B_i = (V_i, E_i) \in \mathcal{B}$ **do**
  if B_i *contains exactly one cut vertex* c_j **then**
    if TestRUPBiconnected$(B_i, \{c_j\}, \{c_j\}) \neq$ `false` **then**
      $\overrightarrow{E}_B \leftarrow \overrightarrow{E}_B \cup \{(c_j, B_i), (B_i, c_j)\}$
    else B_i must be either the beginning or the end of the Eulerian dipath
      if $c_j = \text{ep}_{\text{Start}} \wedge$ TestRUPBiconnected$(B_i, \emptyset, \{c_j\}) \neq$ `false` **then**
        $\text{ep}_{\text{Start}} \leftarrow B_i$; $\overrightarrow{E}_B \leftarrow \overrightarrow{E}_B \cup \{(B_i, c_j)\}$
      else if $c_j = \text{ep}_{\text{End}} \wedge$ TestRUPBiconnected$(B_i, \{c_j\}, \emptyset) \neq$ `false` **then**
        $\text{ep}_{\text{End}} \leftarrow B_i$; $\overrightarrow{E}_B \leftarrow \overrightarrow{E}_B \cup \{(c_j, B_i)\}$
      else return `false`
  else B_i contains two cut vertices c_j, c_{j+1}
    if TestRUPBiconnected$(B_i, \{c_j\}, \{c_{j+1}\}) \neq$ `false` **then**
      $\overrightarrow{E}_B \leftarrow \overrightarrow{E}_B \cup \{(c_j, B_i), (B_j, c_{j+1})\}$
    else return `false`
Construct **RUP** embedding $\mathcal{E}$ from the **RUP** embeddings of the blocks in order of the Eulerian dipath in $\overrightarrow{\mathcal{T}_B}$ from ep_{Start} to ep_{End}
return $\mathcal{E}$

4 Testing Biconnected Graphs

In the following, we define directed SPQR-trees of the primal and dual of biconnected compounds to store all possible embeddings. The main idea of the second phase of our algorithm is that if each nodes' skeleton of the primal directed SPQR-tree has an embedding whose dual is an acyclic dipole, then the dual of the whole graph is an acyclic dipole and, hence, the primal graph is **RUP**. For the **RUP** testing algorithm of biconnected graphs, we assume that the input graph $G = (V, E)$ contains no anti-parallel pairs of edges $(u, v), (v, u) \in E$, and no self-loops $(u, u) \in E$. Note that both wrap around the cylinder in a **RUP** embedding. All anti-parallel pairs of edges can be replaced by introducing a new vertex w and substituting edge (u, v) by edges (u, w) and (w, v). All self-loops are replaced alike. The so obtained graph is **RUP** if and only if the original graph is **RUP**.

*SPQR-tree*s were introduced by Di Battista and Tamassia [9] and provide a description of how a graph is composed. Let G_u be an undirected biconnected graph, e. g., the underlying undirected graph of G. In the definition we adopt

here, the SPQR-tree $\mathcal{T}$ of G_u is unrooted. The nodes of $\mathcal{T}$ either represent a series composition (S), a parallel composition (P), a single edge (Q), or a triconnected component (R). Associated with each node μ of $\mathcal{T}$ is a graph that is homeomorphic to a subgraph of G_u and called the *skeleton* $\text{skel}(\mu)$ of μ. In its original definition, every edge $e = \{u, v\}$ of a skeleton, except for one of a Q-node, is a *virtual edge*, i. e., an edge that represents the subgraph of G_u which connects u and v. This subgraph is also referred to as the *expansion graph* $\text{expg}(e)$ of e. For every virtual edge e in the skeleton of a node μ, there is another node ν that refines the structure of $\text{expg}(e)$. This link is represented by an edge between μ and ν in $\mathcal{T}$. Therefore, every leaf of $\mathcal{T}$ is a Q-node. For simplification, however, we represent edges of the graph directly in the skeleton of an S- ,P-, or R-node, so that we can neglect Q-nodes. If G_u is planar, its SPQR-tree stores all planar embeddings of G_u. They can be obtained by two basic operations: swapping two (virtual) edges in the skeleton of a P-node and flipping the triconnected component represented by an R-node. The embedding of G_u can be obtained by merging all skeletons at their associated virtual edges.

Let $\mathcal{E}$ be a planar embedding of G_u. Then, $\mathcal{E}$ implies an embedding of the skeleton of each node of $\mathcal{T}$. The *dual skeleton* $\text{skel}(\mu)^*$ of a node μ is the dual graph of $\text{skel}(\mu)$. The *dual SPQR-tree* $\mathcal{T}^*$ of G_u is a tree whose nodes' skeletons are the dual skeletons of $\mathcal{T}$ with types S^*, P^*, and R^*, and $\mathcal{T}^*$ inherits the topology of $\mathcal{T}$. Note that the dual graph of the skeleton of a P-node yields a circle of length at least 3, which corresponds to an S-node, and the dual of the skeleton of an S-node is a P-node in turn. Also, the dual of a triconnected and embedded graph is triconnected [21, Thm. 2.6.7] and, hence, the dual of an R-node's skeleton is triconnected as well. Consequently, for the node types of the dual SPQR-tree holds $S^* = P$, $P^* = S$, and $R^* = R$. Moreover, merging all skeletons of $\mathcal{T}^*$ at the associated virtual edges yields the dual graph G_u^* of G_u.

Lemma 4. *The dual SPQR-tree of a graph G_u with embedding $\mathcal{E}$ is the SPQR-tree of the dual graph G_u^* with $S^* = P$, $P^* = S$, and $R^* = R$.*

In the next step, we extend SPQR-trees to directed graphs. Let G be a directed, biconnected graph and $\mathcal{T}$ be the SPQR-tree of its underlying undirected graph. For every skeleton $\text{skel}(\mu)$ of a node μ in $\mathcal{T}$, we obtain the *directed skeleton* $\overrightarrow{\text{skel}}(\mu)$ by directing all non-virtual edges according to their direction in G. See Figs. 2(b) and (c) for an example. In the following, we call an acyclic dipole with source u and sink v a *uv-graph*. Each virtual edge $e = \{u, v\}$ is treated as follows: If its expansion graph $\text{expg}(e)$ is either a uv-graph or a vu-graph, e is represented as a directed edge (u, v) or (v, u), respectively. Otherwise, e remains undirected and is mapped to a subset of the flags $\{\odot, \blacktriangleleft\blacktriangleright, \blacktriangleright\blacktriangleleft\}$, depending on whether $\text{expg}(e)$ contains a source ($\blacktriangleleft\blacktriangleright$) or a sink ($\blacktriangleright\blacktriangleleft$) other than u and v, or a cycle ($\odot$). The *directed SPQR-tree* $\overrightarrow{\mathcal{T}}$ is obtained from $\mathcal{T}$ by replacing each skeleton by its directed counterpart. Likewise, we obtain the *directed dual SPQR-tree* $\overrightarrow{\mathcal{T}^*}$ for a planar embedding $\mathcal{E}$ from $\mathcal{T}^*$, see Figs. 2(c) and (d).

Observe that with the flags, every skeleton in the directed SPQR-tree stores information whether the graph contains cycles, sources, or sinks, and, to a certain

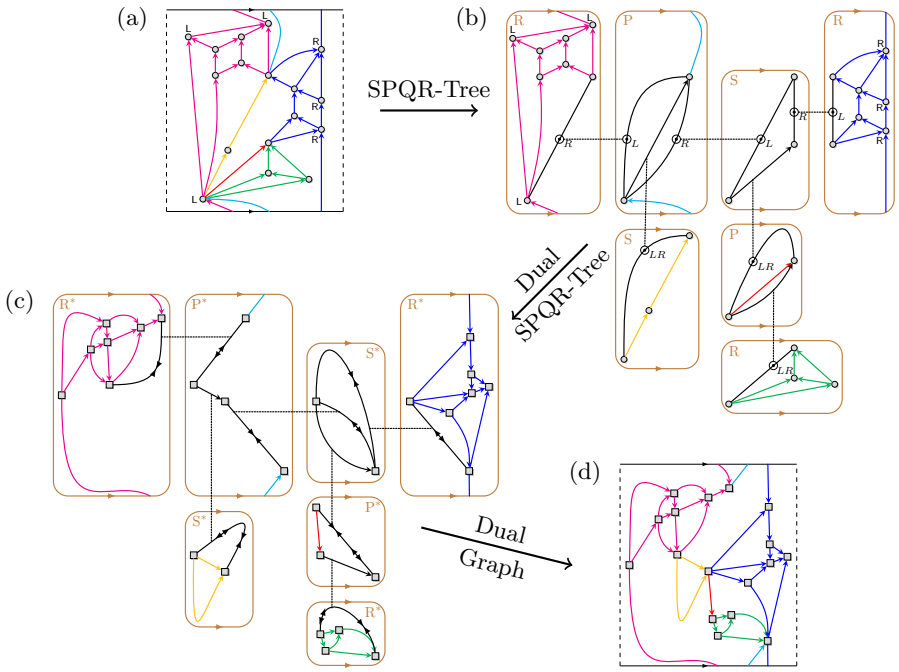

Fig. 2: A compound with triconnected components and its dual together with their SPQR-trees. Black lines are virtual edges. Dotted edges connect the nodes of the SPQR-tree and indicate the associated virtual edges.

extent also how many. This implies transitivity for the flags $\odot$, $\blacktriangleleft\blacktriangleright$, $\blacktriangleright\blacktriangleleft$: Let μ be a node containing a virtual edge e that is refined by node ν and let e' be the virtual edge in ν that is refined by μ. If any virtual edge $e'' \neq e'$ in ν carries flag $X \in \{\odot, \blacktriangleleft\blacktriangleright, \blacktriangleright\blacktriangleleft\}$, then e in μ also has flag X.

The planar embedding of a graph is uniquely defined by the planar embeddings of the skeletons in its SPQR-tree and it can be obtained by merging the nodes at the associated virtual edges. Note that the direction or the flags of a virtual edge are insignificant for the merge operations since the vertices that are matched are always those that represent the same vertex in the graph.

Corollary 3. *The directed dual SPQR-tree of a graph G with embedding $\mathcal{E}$ is the directed SPQR-tree of the dual graph G^* of G.*

From now on, we only consider the biconnected and embedded compound G with acyclic dual G^*. Note that for a strongly connected graph, all virtual edges in the skeletons of its directed SPQR-tree are either directed or carry flag $\odot$, whereas for acyclic graphs, a virtual edge may only have one or both of the flags $\blacktriangleleft\blacktriangleright$ and $\blacktriangleright\blacktriangleleft$.

We introduce optional labels L and R for vertices of G and refine the flag $\odot$ on all virtual edges as follows: If the expansion graph of a virtual edge e with

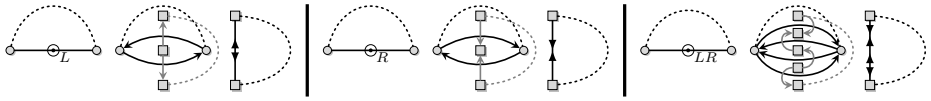

Fig. 3: Auxiliary skeleton

flag $\odot$ contains a vertex with label L, then e's flag is refined to $\odot_L$. Analogously, if $\text{expg}(e)$ contains a vertex with label R, then e's flag is refined to $\odot_R$. In case that $\text{expg}(e)$ contains vertices with both labels, e carries both flags. We say that a directed SPQR-tree is *labeled*, if every $\odot$-flag has been refined. Observe that this implies at least one vertex with label L or R in G. A directed SPQR-tree of an acyclic graph is trivially labeled.

Consider graph G^*, which is acyclic, and contains at least one source and one sink. Every source and every sink in G^* is a face which is bounded by a cycle in G. We assign the label L to every vertex that is incident to a source, and the label R to every vertex that is incident to a sink. Note that a vertex may have both labels.

Lemma 5. *There is a labeled directed SPQR-tree for G such that exactly the vertices incident to a source in G^* are labeled L and those incident to a sink are labeled R.*

Lemma 5 enables us to obtain the directed dual SPQR-tree $\overrightarrow{\mathcal{T}^*}$ of G directly from the directed SPQR-tree $\overrightarrow{\mathcal{T}}$ that is labeled according to its embedding. If a virtual edge in the skeleton of a node in $\overrightarrow{\mathcal{T}}$ has flag $\odot_L$ or $\odot_R$, we can assign to its dual edge the flag $\blacktriangleleft\blacktriangleright$ or $\blacktriangleright\blacktriangleleft$, respectively. The dual of a virtual edge with flag $\blacktriangleleft\blacktriangleright$ or $\blacktriangleright\blacktriangleleft$ then is a virtual edge with flag $\odot_R$ or $\odot_L$, respectively. Observe that for directed graphs, the dual of a dual graph is the *converse* of the primal graph, i. e., G with all edges reversed.

Let $\overrightarrow{\text{skel}}(\mu)$ be the skeleton of a node μ in a labeled directed SPQR-tree $\overrightarrow{\mathcal{T}}$ of a graph G. The *auxiliary skeleton* $\text{aux}(\overrightarrow{\text{skel}}(\mu))$ is derived from $\overrightarrow{\text{skel}}(\mu)$ by substituting each undirected virtual edge $\{u, v\}$ according to its flags by a *proxy* as depicted in Fig. 3. For instance, the proxy of a virtual edge with flag $\odot_L$ is an anti-parallel pair of edges, such that it induces a source in the dual graph, which is in turn the proxy of a virtual edge with flag $\blacktriangleleft\blacktriangleright$. If G has a planar embedding, the new edges inherit the position of $\{u, v\}$ in the rotation systems at u and v. Observe that all constructions preserve planarity.

By Proposition 1, G is **RUP** if and only if its dual G^* is an acyclic dipole. Consider the directed SPQR-tree $\overrightarrow{\mathcal{T}^*}$ of G^*. Then, the skeleton of every node represents the structure of G^* either directly or via flags on its virtual edges.

Lemma 6. *G^* is an acyclic dipole if and only if every auxiliary skeleton of its directed SPQR-tree is an acyclic dipole.*

Algorithm 2: TestRUPBiconnected

Input: Planar biconnected compound $G = (V, E)$, $V_l, V_r \subseteq V$
Output: **RUP** embedding of G such that all $v \in V_l$ (V_r) are incident to the source (sink) of G^*; or `false` if no such **RUP** embedding exists

1 $\mathcal{T} \leftarrow$ ComputeSPQRTree(G) and keep edge directions according to G
2 Obtain $\overrightarrow{\mathcal{T}}$: Root the tree arbitrarily, direct all edges towards root, and let $\mu_1, \ldots, \mu_k$ be a topological sorting of $\mathcal{T}$'s nodes
3 **foreach** $\mu = \mu_1, \ldots, \mu_k$ **do** ignoring virtual edge associated with parent
4 **if** $\overrightarrow{\text{skel}}(\mu)$ *contains virtual edge with flag* $\odot$ *or a cycle* **then**
5 Set flag $\odot$ on virtual edge of parent associated with μ
6 **else** $\overrightarrow{\text{skel}}(\mu)$ is uv-graph
7 Direct virtual edge of parent associated with μ from u to v
8 Propagate completive information from root to children
9 **if** $\exists$ *skeleton with* > 2 $\odot$*-flags* **then return** `false`
10 **foreach** *node* μ *and vertex* v *in* $\overrightarrow{\text{skel}}(\mu)$ **do if** $v \in V_l$ *(*$v \in V_r$*)* **then**
11 label v with L (R) in $\overrightarrow{\text{skel}}(\mu)$
12 $\overrightarrow{\text{skel}}(\mu) \leftarrow$ skeleton with maximum number of virtual edges with flag $\odot$
13 **if** $\overrightarrow{\text{skel}}(\mu)$ *contains* ≥ 1 *virtual edges with flag* $\odot$ **then**
14 **foreach** $f \in \{\odot_L, \odot_R, \odot_L \,\&\, \odot_R\}$ **do**
15 Refine $\odot$-flag on one virtual edge to f
16 Refine $\odot$-flag on other virtual edge, if existent, to complement of f
17 Establish transitivity
18 **if** TestRUPLabeledSPQRTree($\overrightarrow{\mathcal{T}}$) $\neq$ `false` **then goto 22**
19 **return** `false`
20 **else** no virtual edges with $\odot$-flag
21 **if** TestRUPLabeledSPQRTree($\overrightarrow{\mathcal{T}}$) $=$ `false` **then return** `false`
22 Build and return the **RUP** embedding of G by merging all skeletons

Algorithm 2 takes as input a planar, biconnected compound $G = (V, E)$ with two sets $V_l, V_r \subseteq V$. It returns a **RUP** embedding of G such that all vertices in V_l (V_r) are incident to the source (sink) of G^* or `false` if no such embedding exists. First, the SPQR-tree of G is computed which can be done in linear time [13]. From lines 1–8 the directed SPQR-tree $\overrightarrow{\mathcal{T}}$ is obtained, i. e., each virtual edges is either directed or the $\odot$ flag is set. Note that for each skeleton an algorithm is executed whose running time is linear in the number of vertices in the skeleton. Since the number of edges represented in the skeletons is in $\mathcal{O}(|V|)$ [9], we obtain a running time linear in the size of G. By Lemma 6, the dual of every skeleton must be an acyclic dipole. Hence, the primal skeleton can contain at most one virtual edge with $\odot_L$- and one with $\odot_R$-flag and, thus, at most two $\odot$-edges (cf. line 9). Next, in the skeletons, the representatives of the vertices in V_l and V_r are labeled accordingly. In lines 13–22, the algorithm tries all transitive refinements of the $\odot$-flag. Note that if there is a skeleton

with two cylic edges e_1, e_2, either e_1 gets $\odot_L$ and e_2 gets $\odot_R$ or vice versa. In a skeleton with one $\odot$-edge, the three refinements $\odot_L$,$\odot_R$, and $\odot_L \,\&\, \odot_R$ are possible. If all $\odot$-edges in one skeleton have been refined, the refinements of all other $\odot$-edges are determined uniquely by transitivity. Thus, the algorithm has to test at most three different refinements altogether. The obtained labeled SPQR-tree is passed to TestRUPLabeledSPQRTree. This routine verifies that each skeleton has an embedding such that its dual is an acyclic dipole and the vertices with labels L and R are incident to the source and sink, respectively. TestRUPLabeledSPQRTree proceeds for every node μ as follows: If μ is of type R, its auxiliary skeleton is obtained. A planarity test yields the (up to flipping) unique embedding of the auxiliary skeleton. If one dual is an acyclic dipole, then so is the other. If in one of them the vertices with label L (R) are incident to the source (sink), this embedding is retained. Otherwise, the test aborts and returns `false`. If μ is a P-node, we define the rotation system at any of the two vertices in the skeleton as follows: Denote by $e_1^+, \ldots, e_m^+$ and by $e_1^-, \ldots, e_n^-$ all outgoing and incoming edges of the vertex in arbitrary order. The rotation system is then $e_1^+, \ldots, e_m^+, e_{\odot_L}, e_1^-, \ldots, e_n^-, e_{\odot_R}$. Observe that if the $\odot$-edge carries both flags, then either $m = 0$ or $n = 0$. Because of planarity, this rotation system also defines the rotation system at the other vertex. If one or both of the $\odot$-edges are missing, they are skipped. In all cases, the dual is an acyclic dipole. Note that whenever an anti-parallel pair of edges is adjacent in the rotation system, this always results in either a source or sink in the dual, and the duals of the proxies of $e_{\odot_L}$ and $e_{\odot_R}$ are a source and a sink, respectively. From the perspective a P-node provides on the whole graph, the split pair can be incident to both the source and sink of the dual. Hence, no further tests are needed. If μ is an S-node, its skeleton has a unique embedding and, hence, also its auxiliary skeleton. The algorithm computes the dual and verifies that the vertices with L and R are incident to the source and sink, respectively. If all checks succeed, TestRUPBiconnected returns a **RUP** embedding of G. For every skeleton, the test can be performed in time linear in the size of the skeleton. Finally, we obtain:

Theorem 1. *There is a linear-time algorithm to decide whether a strongly connected graph is **RUP**.*

5 Conclusion

We presented a linear time algorithm to check whether a strongly connected graph is drawable upward planar on a rolling cylinder. We were also able to extend the **RUP**-test to graphs without sources and sinks, i. e., graphs consisting of multiple compounds. The proof is left as future work.

References

1. Auer, C., Bachmaier, C., Brandenburg, F.J., Gleißner, A., Hanauer, K.: The duals of upward planar graphs on cylinders. In: Golumbic, M.C., Stern, M., Levy, A., Morgenstern, G. (eds.) WG 2012. LNCS, vol. 7551, pp. 103–113. Springer (2012)

2. Auer, C., Bachmaier, C., Brandenburg, F.J., Gleißner, A.: Classification of planar upward embedding. In: van Kreveld, M., Speckmann, B. (eds.) GD 2011. LNCS, vol. 7034, pp. 415–426. Springer (2012)
3. Bachmaier, C., Brandenburg, F.J., Brunner, W., Fülöp, R.: Drawing recurrent hierarchies. J. Graph Alg. App. 16(2), 151–198 (2012)
4. Bang-Jensen, J., Gutin, G.: Digraphs: Theory, Algorithms and Applications. Springer, 1st edn. (2000)
5. Bertolazzi, P., Battista, G.D., Mannino, C., Tamassia, R.: Optimal upward planarity testing of single-source digraphs. SIAM J. Comput. 27(1), 132–169 (1998)
6. Brandenburg, F.J.: On the curve complexity of upward planar drawings. In: Mestre, J. (ed.) Computing: The Australasian Theory Symposium, CATS 2012. CRPIT, vol. 128, pp. 27–36. Australian Computer Society (2012)
7. Brandenburg, F.J.: Upward planar drawings, on the standing and the rolling cylinders. Comput. Geom. Theory Appl. (2013), to appear
8. Di Battista, G., Tamassia, R.: Algorithms for plane representations of acyclic digraphs. Theor. Comput. Sci. 61(2–3), 175–198 (1988)
9. Di Battista, G., Tamassia, R.: On-line planarity testing. SIAM J. Comput. 25(5), 956–997 (1996)
10. Dolati, A., Hashemi, S.M.: On the sphericity testing of single source digraphs. Discrete Math. 308(11), 2175–2181 (2008)
11. Foldes, S., Rival, I., Urrutia, J.: Light sources, obstructions and spherical orders. Discrete Math. 102(1), 13–23 (1992)
12. Garg, A., Tamassia, R.: On the computational complexity of upward and rectilinear planarity testing. SIAM J. Comput. 31(2), 601–625 (2001)
13. Gutwenger, C., Mutzel, P.: A linear time implementation of SPQR-trees. In: Marks, J. (ed.) GD 2000. LNCS, vol. 1984, pp. 77–90. Springer (2001)
14. Hansen, K.A.: Constant width planar computation characterizes ACC^0. Theor. Comput. Sci. 39(1), 79–92 (2006)
15. Harary, F., Schwenk, A.J.: The number of caterpillars. Discrete Math. 6(4), 359–365 (1973)
16. Hashemi, S.M., Rival, I., Kisielewicz, A.: The complexity of upward drawings on spheres. Order 14, 327–363 (1998)
17. Hashemi, S.M.: Digraph embedding. Discrete Math. 233(1–3), 321–328 (2001)
18. Kelly, D.: Fundamentals of planar ordered sets. Discrete Math. 63, 197–216 (1987)
19. Limaye, N., Mahajan, M., Sarma, J.M.N.: Evaluating monotone circuits on cylinders, planes and tori. In: Durand, B., Thomas, W. (eds.) STACS 2006. LNCS, vol. 3884, pp. 660–671. Springer (2006)
20. Limaye, N., Mahajan, M., Sarma, J.M.N.: Upper bounds for monotone planar circuit value and variants. Comput. Complex. 18(3), 377–412 (2009)
21. Mohar, B., Thomassen, C.: Graphs on Surfaces. Johns Hopkins Series in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD, USA (2001)
22. Sugiyama, K., Tagawa, S., Toda, M.: Methods for visual understanding of hierarchical system structures. IEEE Trans. Syst., Man, Cybern. 11(2), 109–125 (1981)
23. Thomassen, C.: Planar acyclic oriented graphs. Order 5(1), 349–361 (1989)