An SPQR-Tree-Like Embedding Representation for Upward Planarity

Guido Brückner[1(✉)], Markus Himmel[1], and Ignaz Rutter[2]

[1] Karlsruhe Institute of Technology, Karlsruhe, Germany
brueckner@kit.edu, markus.himmel@studentkit.edu
[2] University of Passau, Passau, Germany
rutter@fim.uni-passau.de

Abstract. The SPQR-tree is a data structure that compactly represents all planar embeddings of a biconnected planar graph. It plays a key role in constrained planarity testing.

We develop a similar data structure, called the UP-tree, that compactly represents all upward planar embeddings of a biconnected single-source directed graph. We demonstrate the usefulness of the UP-tree by solving the upward planar embedding extension problem for biconnected single-source directed graphs.

1 Introduction

A natural extension of planarity to directed graphs (digraphs) is to consider planar drawings where each edge is drawn as a y-monotone curve. Such drawings are called *upward planar*, and a graph admitting an upward planar drawing is *upward planar*. A planar (combinatorial) embedding $\mathcal{E}$ of a graph G is an *upward planar embedding* if G has an upward planar drawing whose (combinatorial) embedding is $\mathcal{E}$. Whereas undirected graphes can be tested for planarity in linear time, upward planarity testing is NP-complete in general, though there are efficient algorithms for graphs with a single source [4,24] and graphs with a fixed embedding [3]. In the special case of *st-graphs*, i.e., graphs with a single source s and a single sink t with s and t on the same face, planar embeddings are the same as the upward planar embeddings [28], and hence upward planarity and planarity are equivalent.

A related but different planarity notion for digraphs is *level planarity*, where the vertices of the graph have fixed levels that correspond to horizontal lines in the drawing. The task is to order the vertices on each level so that the drawing is planar. Level planarity can be tested in linear time [26] by a quite involved algorithm, or in quadratic time by several simpler algorithms [11,20,29].

In a constrained embedding problem, one seeks a planar embedding of a given graph that satisfies additional constraints. Typical examples are simultaneous

---

embeddings with fixed edges [5], cluster planarity [19], constraints on the face sizes [13,14], optimizing the depth of the embedding [2] and optimizing the bends in an orthogonal drawing [6,7,18]. One of the most prominent examples of the last years is the *partial drawing extension* problem, which asks whether a given drawing of a subgraph can be extended to a planar drawing of the whole graph. The *partial embedding extension* problem is strongly related, here the input is a planar embedding of a subgraph and the question is whether it can be extended to a planar embedding of the whole graph. For undirected planar graphs the two problems are equivalent and can be solved in linear time [1,25].

One of the key tools for all of these applications is the SPQR-tree, which compactly represents all planar embeddings of a biconnected planar graph G and breaks down the complicated task of choosing a planar embedding of G into simpler independent embedding choices of its triconnected components [15–17,23,27,30]. In fact, these embeddings are either uniquely determined up to reversal, or they consist in arbitrarily choosing a permutation of parallel edges between two pole vertices. The common approach for attacking the above-mentioned constrained embedding problems is to project the constraints on the global embedding to local constraints on the skeleton embeddings that can then be satisfied by consistent local choices. While the implementation details are often highly technical and non-trivial, the approach has proven to be extremely successful.

In comparison, relatively little is known about constrained planarity problems for planarity notions of digraphs. Brückner and Rutter [10] study the problem of extending a given partial drawing of a level graph and Da Lozzo et al. [12] study the same question for upward planarity. In general, extending a given partial upward planar drawing requires to determine an upward planar embedding that (i) extends the embedding of the partial drawing, and (ii) admits a drawing that extends the given drawing. Here step (i) requires solving the embedding extension problem but with additional constraints that ensure that a drawing extension is feasible. It is worth noting that for upward planarity the embedding extension problem and the drawing extension problem are distinct; Da Lozzo et al. show that, generally, even if an upward planar embedding of the whole graph is given, it is NP-complete to decide whether it can be drawn such that it extends a given partial drawing [12, Theorem 2]. On the positive side, they present tractability results for directed paths and cycles with a given upward planar embedding, and for st-graphs. The restriction to st-graphs allows a relatively simple characterization of the upward planar embeddings that extend the given partial drawings [12, Lemma 6], which yields an $O(n \log n)$-time algorithm for step (ii). For step (i), Da Lozzo et al. exploit the fact that for st-graphs, the choice of an upward planar embedding is equivalent to choosing a planar embedding, and hence the SPQR-tree allows to efficiently search for an upward planar embedding satisfying the additional constraints required by condition (ii).

In this paper, we seek to generalize the approach of Da Lozzo et al. to biconnected single-source graphs. The key difficulty in this case is that neither do we have access to all the upward planar embeddings of such graphs, nor is it known

what the necessary and sufficient conditions are for an upward planar embedding to admit a drawing that extends a given subdrawing.

Contribution and Outline. We construct a novel SPQR-tree-like embedding representation, called the UP-tree, that represents exactly the upward planar embeddings of a biconnected single-source graph. As in SPQR-trees, the embedding choices in the UP-tree are broken down into independent embedding choices of skeleton graphs that are either unique up to reversal or allow to arbitrarily permute parallel edges between two poles. As such, UP-trees can take the role of SPQR-trees for constrained embedding problems in upward planarity, making them a powerful tool with a broad range of applications. We demonstrate this by giving an quadratic-time algorithm for the upward planar embedding extension problem for biconnected single-source graphs.

After introducing some preliminaries in Sect. 2, we review the results on decomposing upward planar single-source digraphs due to Hutton and Lubiw [24] in Sect. 3. We proceed to extend this idea from a single decomposition to decomposition trees. Proofs of statements marked with a star ($\star$) can be found in the full version [9]. In Sect. 4, we define the UP-tree and in Sect. 5 we use it to solve the partial upward embedding extension problem.

2   Preliminaries

Let $G = (V, E)$ be a connected simple undirected graph. A *cutvertex* of G is a vertex whose removal disconnects G. We say that G is *biconnected* if it has no cutvertex. We say that $\{u, v\}$ is a *cutpair* if there are connected subgraphs H_1, H_2 of G with $H_1 \cup H_2 = G$ and $H_1 \cap H_2 = \{u, v\}$. If a graph has no cutpair it is *triconnected.*

Decomposition Trees. Assume that G is biconnected. A *decomposition* along a cutpair $\{u, v\}$ of G is defined as follows. Let μ_1, μ_2 be two nodes connected by an undirected arc (μ_1, μ_2). Node μ_i is equipped with a multigraph $H_i \cup \{(u, v)\}$ called its *skeleton* denoted by $\mathrm{skel}(\mu_i)$. The newly added edge (u, v) is a *virtual edge* and corresponds to μ_2 in μ_1 and to μ_1 in μ_2, respectively. This is formalized as functions $\mathrm{corr}_{\mu_1} \colon (u, v) \mapsto \mu_2$ and $\mathrm{corr}_{\mu_2} \colon (u, v) \mapsto \mu_1$. If there exists a cutpair $\{u', v'\}$ in $\mathrm{skel}(\mu_i)$ and we may once again decompose along that cutpair. By repeating this process we obtain an unrooted *decomposition tree* $\mathcal{T}$.

Let $a = (\mu, \nu)$ be an arc of $\mathcal{T}$. Then $\mathrm{skel}(\mu)$ and $\mathrm{skel}(\nu)$ share two vertices u, v and the existence of a can be traced back to a decomposition along u, v. We then say that the *poles* of μ in ν are u and v. When ν is clear from the context we also simply refer to u and v as the poles of μ.

A decomposition can be reverted by contracting an arc (μ, ν) of $\mathcal{T}$ and merging the skeletons of μ and ν. To merge $\mathrm{skel}(\mu)$ and $\mathrm{skel}(\nu)$, remove from $\mathrm{skel}(\mu)$ the virtual edge e with $\mathrm{corr}_\mu(e) = \nu$ and from $\mathrm{skel}(\nu)$ the virtual edge e' with $\mathrm{corr}_\nu(e') = \mu$ and set the union of these two graphs as the skeleton of the node obtained by contracting (μ, ν) in $\mathcal{T}$. This is a *composition* along (μ, ν).

Consider an arc $a = (\mu, \nu)$ of $\mathcal{T}$. Removing a from $\mathcal{T}$ separates $\mathcal{T}$ into two subtrees $\mathcal{T}_\mu$ and $\mathcal{T}_\nu$ containing μ and ν, respectively. Define the *pertinent graph* of μ in ν as the skeleton of the single node obtained by contracting all arcs in $\mathcal{T}_\mu$. Again, when ν is clear from the context we simply refer to this graph as the pertinent graph of μ and denote it by $G(\mu)$.

Rooted Decomposition Trees and Planar Embeddings. Throughout this paper let an *embedding* of a graph denote a rotation system together with an outer face. Decomposition trees can be used to decompose not only a graph, but also an embedding of it. Consider a biconnected graph G together with a planar embedding $\mathcal{E}$. Let $e^\star$ be an edge of G incident to the outer face of $\mathcal{E}$. Further, let $\mathcal{T}$ be a decomposition tree of G rooted at a node whose skeleton contains $e^\star$. Equip the skeleton of each node μ of $\mathcal{T}$ with an embedding as follows. The embedding of skel(μ) is obtained from $\mathcal{E}$ by contracting for each virtual edge (u, v) of skel(μ) the pertinent graph $G(\mathrm{corr}_\mu(u, v))$ into a single edge. These embeddings of the skeletons of the nodes of $\mathcal{T}$ are referred to as a *configuration*. The fact that $e^\star$ is incident to the outer face gives two properties. First, the edge $e^\star$ lies on the outer face of the skeleton of the root node of $\mathcal{T}$. Second, every non-root node ν of $\mathcal{T}$ has some parent node μ and the virtual edge e with $\mathrm{corr}_\nu(e) = \mu$ lies on the outer face of skel(ν). We extend our notion of a *configuration* to any set of embeddings of the skeletons of the nodes of $\mathcal{T}$ that fulfills these two properties.

Recall that decomposition trees allow for (graph) composition along arcs. We can also compose embeddings. When contracting an arc (μ, ν), we merge skel(μ) and skel(ν) as described above. Obtain the embedding of the merged skeleton by replacing the occurrences of the virtual edge in the rotation system around its poles by the appropriate rotation system in the embedding of the other skeleton. This means that G together with a planar embedding can be decomposed into a decomposition tree $\mathcal{T}$ together with a configuration. And symmetrically, $\mathcal{T}$ together with any configuration can be composed into a planar embedding of G.

SPQR-Trees. As described in the previous paragraph, decomposition trees separate independent choices in finding planar embeddings of a graph. We may either choose an embedding of the entire graph, which is generally very complex, or we may decompose the graph into smaller skeletons, independently choose embeddings of these skeletons and compose them into an embedding of the entire graph. In this sense decomposition trees implement a tradeoff between making few complex choices or many simple choices.

The *SPQR-tree* is a decomposition tree that makes this tradeoff in favor of many simple choices. SPQR-trees have four kinds of nodes, all of whose skeletons offer only few and well-structured embedding choices. (i) R-nodes are nodes whose skeleton is triconnected. Such skeletons have a unique planar embedding up to flipping. (ii) S-nodes are nodes whose skeleton is a simple cycle. Such skeletons offer no embedding choice (recall that the outer face is fixed by the rooting). Adjacent S-nodes are contracted into one larger S-node, i.e., an S-node whose skeleton is a larger simple cycle. This means that in SPQR-trees no two S-nodes are adjacent. (iii) P-nodes are nodes whose skeleton is a multigraph that
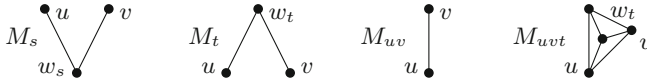
Fig. 1. The four markers used by Hutton and Lubiw. The markers are digraphs; in the figure, all edges are directed upward.

consists of two vertices connected by three or more edges. The order of these edges may be arbitrarily permuted. Again, adjacent P-nodes are contracted into one larger P-node, i.e., no two P-nodes are adjacent. (iv) Q-nodes are nodes whose skeleton consists of two vertices connected by two edges, namely one virtual edge and one non-virtual edge. They offer no embedding choice. Note that only the skeletons of Q-nodes contain non-virtual edges. See Fig. 3(a) and (b) for a graph and its SPQR-tree decomposition.

3 Decomposition Trees and Upward Planar Embeddings

Recall from the previous section that for biconnected graphs we can decompose any planar embedding into planar embeddings of the skeletons of a decomposition tree; and symmetrically, we can compose a planar embedding of the whole graph from planar embeddings of the skeletons. In this section we find a similar relationship between upward planar embeddings of a biconnected single-source digraph G and upward planar embeddings of the skeletons of a suitably-defined decomposition tree of G.

3.1 Decompositions and Upward Planar Embeddings

In this section we review the decomposition result of Hutton and Lubiw and formulate the interface between their result and our results.

Let G be a biconnected single-source digraph together with an upward planar embedding $\mathcal{E}$. Further, let $e^\star$ denote the edge around the source of G that is leftmost in $\mathcal{E}$. Now let H_1, H_2 be two subgraphs of G with (i) $H_1 \cup H_2 = G$, (ii) $H_1 \cap H_2 = \{u, v\}$, (iii) $e^\star \in H_1$ and (iv) $H_1 \setminus \{u, v\}$ or $H_2 \setminus \{u, v\}$ is connected.

Hutton and Lubiw construct two graphs H_1' and H_2' from H_1 and H_2 by including one of the *markers* shown in Fig. 1. Markers are simple digraphs with two vertices u, v that connect the marker to the remaining graph. The marker in H_1' is designed to represent H_2 and the marker in H_2' is designed to represent H_1. If there exists a directed path from u to v we say that u *dominates* v and write $u < v$ for short. Otherwise u and v are *incomparable*. The vertex v is a *source* if it has no incoming edges in G, a *sink* if it has no outgoing edges in G and an *internal* vertex if it has both incoming and outgoing edges in G. Markers are determined based on whether $u < v$ and whether v is a source, sink or internal vertex in H_1 and H_2: If u and v are incomparable in G, set $H_1' = H_1 \cup M_t$ and $H_2' = H_2 \cup M_s$. Otherwise, assume $u < v$. Define H_1' as follows. If v is a

source in H_2 set $H_1' = H_1 \cup M_t$. If v is a sink in H_2 set $H_1' = H_1 \cup M_{uv}$. Otherwise v is an internal vertex in H_2 and we set $H_1' = H_1 \cup M_{uvt}$. Define H_2' as follows. If v is a source in H_1 set $H_2' = H_2 \cup M_t$, otherwise set $H_2' = H_2 \cup M_{uv}$. See Fig. 4 in the full version for example decompositions.

Recall that decomposition trees of planar graphs allow for (de-)composition of planar embeddings. Hutton and Lubiw provide a similar property for the graphs G, H_1' and H_2'.

Theorem 1 ($\star$, implicit in [24]). *Let $\mathcal{E}$ be an upward planar embedding of G with $e^\star$ as the leftmost edge around s. Then $\mathcal{E}$ induces upward planar embeddings $\mathcal{F}_1, \mathcal{F}_2$ of H_1', H_2', respectively with the following properties. In $\mathcal{F}_1$, $e^\star$ is the leftmost edge around the source of H_1'. In $\mathcal{F}_2$, the edges of the marker are leftmost around the source of H_2'. Conversely, if $\mathcal{F}_1$ and $\mathcal{F}_2$ are upward planar embeddings of H_1' and H_2' such that $e^\star$ is the leftmost edge around the source of H_1' and the edges of the marker are leftmost around the source of H_2', then the composition of these embeddings is upward planar.*

Hutton and Lubiw do not explicitly state Theorem 1. Instead, Theorems 6.5, 6.7, 6.8 and 6.9 in [24] discuss the same situation as Theorem 1, but are only concerned with upward planarity, not with the embeddings involved. See the full version for a detailed discussion.

3.2 Decomposition Trees and Upward Planar Embeddings

The approach of Hutton and Lubiw is to decompose a single-source digraph G into two smaller single-source digraphs G_1, G_2 and use Theorem 1 to translate upward-planarity testing of G to upward-planarity testing of two smaller instances H_1', H_2'. Observe that the markers and the replacement rules are defined so that both H_1' and H_2' are single-source digraphs. This means that H_1' and H_2' can be recursively decomposed. Note that in the context of connectivity markers are treated simply as edges, i.e., markers are not decomposed further. When a graph cannot be further decomposed it is triconnected and therefore has a unique planar embedding which can be tested for upward planarity in linear time using the algorithm of Bertolazzi et al. [4]. In the context of upward planarity testing the full marker graph is considered. Upward planar embeddings of H_1' and H_2' can then be composed to an upward planar embedding of G. In the context of embedding composition markers are again treated simply as edges. In particular, it does not matter whether the clockwise order of the edges incident to u in M_{uvt} is $(u, v), (u, x), (u, w_t)$ or $(u, w_t), (u, x), (u, v)$.

We use a different approach. Instead of testing H_1' and H_2' for upward-planarity separately, we manage them as the skeletons of two nodes in a decomposition tree $\mathcal{T}$. Note that Theorem 1 requires $H_1 \setminus \{u, v\}$ or $H_2 \setminus \{u, v\}$ to be connected. We call such a decomposition *maximal*. We then decompose these skeletons further, which grows the decomposition tree. A *maximal-decomposition tree* is a decomposition tree obtained by performing only maximal decompositions. A *configuration* equips the skeleton of each node in the tree with an

upward planar embedding. In this embedding, $e^\star$ or the marker that represents the component that contains $e^\star$ must be incident to the outer face and leftmost around the source of the skeleton. See Fig. 3(c) for an example of a maximal-decomposition tree. Applying Theorem 1 at each decomposition step gives the following.

Theorem 2. *Let G be a biconnected graph with a single source s, let $e^\star$ be an edge of G incident to s and let $\mathcal{T}$ denote a maximal-decomposition tree of G. Then the upward-planar embeddings of G in which $e^\star$ is the leftmost edge around s correspond bijectively to the configurations of $\mathcal{T}$.*

We could use Theorem 2 directly to represent all upward planar embeddings of a graph. But we also show that decomposition trees are uniquely defined by the decompositions that are executed, but not by the order of these decompositions. This means that just like we can talk about *the* SPQR-tree decomposition for a graph we will be able to talk about *the* UP-tree decomposition. The benefit of this is that we can use a UP-tree decomposition to determine that some constrained representation problem has no solution without having to consider other conceivable UP-tree decompositions.

To prove uniqueness, we show that the order of the decompositions is irrelevant. We then apply the decompositions as defined by the SPQR-tree decomposition, which is unique, and obtain the unique UP-tree decomposition. To this end, we prove that the marker replacement rules do not depend on the order of the decompositions. Recall that the marker replacement rules depend on vertex dominance and the local neighborhood of certain vertices. We prove Lemma 1, which states that decompositions preserve vertex dominance and Lemma 2, which states that decompositions preserve the local neighborhood of certain vertices.

Lemma 1 ($\star$). *Let G be a biconnected single-source digraph and let H_1', H_2' denote the result of decomposing along a cutpair $\{u, v\}$ of G. For $i = 1, 2$ and any two vertices x, y in H_i' it is $x < y$ in H_i' if and only if $x < y$ in G.*

Lemma 2 ($\star$). *Let G be a biconnected single-source digraph and let H_1', H_2' denote the result of decomposing along a cutpair $\{u, v\}$ of G. For $i = 1, 2$ let $\{x, y\}$ denote a cutpair of H_i' that separates H_i' into F_1 and F_2 and G into D_1 and D_2. Then y is a source in F_1 if and only if y is a source in D_1. Moreover, y is a source, sink or internal vertex in F_2 if and only if y is a source, sink or internal vertex in D_2, respectively.*

Lemmas 1 and 2 immediately give the following.

Lemma 3. *Let G be a biconnected graph with a single source s, let $e^\star$ be an edge of G incident to s and let $\mathcal{T}$ denote a decomposition tree of G. Then $\mathcal{T}$ relative to $e^\star$ is uniquely defined by the decompositions regardless of their order.*

A configuration of $\mathcal{T}$ can be computed as follows. Recall that all skeletons are single-source digraphs. We may therefore run the algorithm due to

Bertolazzi et al. [4] on each skeleton. Observe that in a configuration of $\mathcal{T}$ relative to $e^\star$ the skeleton of each node μ of $\mathcal{T}$ must be embedded so that $e^\star$ or the marker that corresponds to the component that contains $e^\star$ must appear leftmost around the source of $\mathrm{skel}(\mu)$. We can enforce this by rooting the decomposition tree constructed by the algorithm of Bertolazzi et al. at the Q-node corresponding to $e^\star$ or an edge of the marker that corresponds to the component that contains $e^\star$.

4   UP-Trees

We are ready to construct the UP-tree, a maximal-decomposition tree designed to mimic the SPQR-tree. Let G be a biconnected directed single-source graph. The base of the construction is the decomposition tree obtained by performing the same set of decompositions as in the construction of the SPQR-tree decomposition of the underlying undirected graph of G. We then perform two additional steps. The first step is to split P-nodes into chains of smaller nodes. The second step is to determine whether skeletons of R-nodes can be reversed and to contract some arcs of the decomposition tree. In both steps, we reason about upward planarity of fixed embeddings with the following lemma due to Bertolazzi et al. [4].

Let G be a biconnected single-source graph together with a planar embedding. The *face-sink graph* F of G has the vertices and faces of G as its vertices. It contains an undirected edge $\{f, v\}$ if f is a face of G and v is a vertex of G that is incident to f and both edges incident to v and f are directed towards v. The following lemma implies a linear-time algorithm that tests an embedding for upward planarity and outputs for each face whether it can be the outer face.

Lemma 4 **(**[4, Theorem 1]**).** *Let G be an embedded planar single-source digraph and let h be a face of G. Graph G has an upward planar drawing that preserves the embedding with outer face h if and only if all of the following is true: (i) graph F is a forest (ii) there is exactly one tree T of F with no internal vertices of G, while the remaining trees have exactly one internal vertex; (iii) h is in tree T; and (iv) the source of G is in the boundary of h.*

4.1   P-Node Splits

In SPQR-trees, the edges of P-nodes may be arbitrarily permuted. In decomposition trees for upward planar graphs there are stricter rules for the ordering of the markers in P-nodes. In this section, we determine these rules and find that by breaking up the P-nodes into chains of smaller nodes we obtain a decomposition tree for upward planarity whose P-nodes exhibit the same behavior as in SPQR-trees, i.e., their edges may be arbitrarily permuted. The idea is that certain kinds of markers must appear consecutively.

First, we argue that all M_{uv} markers must appear consecutively. To see this, note that if M_s appears between two M_{uv} markers then the outer face is not
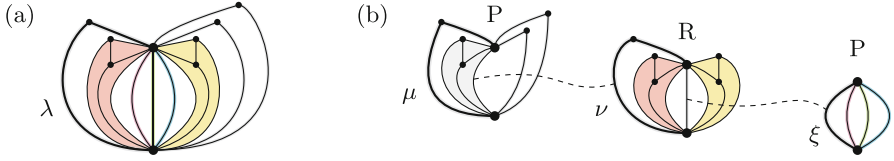
Fig. 2. Splitting a P-node λ obtained from the SPQR-tree (a) into a chain of smaller nodes μ, ν, ξ (b). The bold marker represents the component that contains the edge $e^\star$.

incident to the source of the skeleton, which is vertex w_s of M_s. If a marker M with $M = M_t$ or $M = M_{uvt}$ appears between two M_{uv} markers then the face incident to w_t of M and a marker M_{uv} is not connected to the outer face and not connected to an internal vertex. In all cases the conditions from Lemma 4 are violated.

Moreover, all M_{uv} and M_{uvt} markers must appear consecutively. To see this, note that if M_t appears between two markers M_{uv} or M_{uvt} the vertex w_t of M_t cannot be connected to an internal vertex or the outer face and apply Lemma 4.

These observations motivate the following restructuring of P-nodes. Let λ denote a P-node obtained from the SPQR-tree. The *parent marker* in $\mathrm{skel}(\lambda)$ is the marker that corresponds to the parent node of λ. If the parent marker in $\mathrm{skel}(\lambda)$ is M_s all other markers must be M_t. In this case these markers can already be arbitrarily permuted and nothing further needs to be shown. Otherwise the parent marker is M_t or M_u (recall that by definition of H'_2 the parent marker is not M_{uvt}). See Fig. 2(a) where the parent marker is M_t (the case for M_u is similar). Because all M_{uv} and M_{uvt} markers must appear consecutively, we create a new P-node μ that contains the parent marker of $\mathrm{skel}(\lambda)$, all M_t markers of $\mathrm{skel}(\lambda)$ and a single M_{uvt} marker to represent all M_{uv} and M_{uvt} markers of $\mathrm{skel}(\lambda)$. This marker corresponds to a new node ν that contains all M_{uvt} markers of $\mathrm{skel}(\lambda)$ and—because all M_{uv} markers must appear consecutively— a single M_{uv} marker. This marker corresponds to a new node ξ that contains all M_{uv} markers of $\mathrm{skel}(\lambda)$. If $\mathrm{skel}(\lambda)$ contains no M_{uvt} marker we can include a M_{uv} marker instead of a M_{uvt} marker in $\mathrm{skel}(\mu)$ and connect it directly to ξ, the node ν can then be omitted.

The new node μ has the property that its markers can be arbitrarily permuted, i.e., it is a P-node. Observe that there can be at most two M_{uvt} markers in $\mathrm{skel}(\lambda)$. This means that $\mathrm{skel}(\nu)$ has at most four markers and its embedding is fixed up to reversal, i.e., it is an R-node. Finally, the new node ξ also has the property that its markers can be arbitrarily permuted, i.e., it is also a P-node. See Fig. 2(b) and Fig. 3(c) and (d) for a larger example. We conclude the following.

Lemma 5. *Let G be a biconnected digraph with a single source s and let $e^\star$ denote an edge incident to s. There exists a decomposition tree $\mathcal{T}$ that (i) represents all upward planar embeddings of G in which $e^\star$ is the leftmost edge around s, and (ii) the children of all P-nodes in $\mathcal{T}$ can be arbitrarily permuted.*

4.2   Arc Contractions

Recall that in SPQR-trees the skeletons of R-nodes are triconnected, i.e., their planar embedding is fixed up to reversal. So, every R-node offers one degree of freedom, namely, whether it has some reference embedding or the reversal thereof. In this section we alter our decomposition tree so that it has this same property.

By definition the marker corresponding to the parent node is leftmost in any embedding of a skeleton. Hence, this marker is incident to the outer face. Reversing the embedding of the skeleton is equivalent to choosing the other face incident to the marker as the outer face. Theorem 2 guarantees that any configuration of $\mathcal{T}$ can be composed to an upward planar embedding. This means that a skeleton can be reversed if and only if both faces incident to the parent marker can be chosen as the outer face. This can be checked with the upward planarity test for embedded single-source graphs due to Bertolazzi et al. [4], which also outputs the set of faces that can be chosen as the outer face. If both incident faces are candidates for the outer face this node does indeed offer a degree of freedom and we leave it unchanged. Otherwise, if only one incident face is a candidate for the outer face this node does not offer a degree of freedom. We then merge it with its parent node and contract the corresponding arc in the decomposition tree. This leads to an R-node with a larger skeleton.

See Fig. 5 (a) in the full version for an upward planar graph G and (b) a decomposition tree thereof. Parts of the face sink graphs of $\mathrm{skel}(\mu)$ and $\mathrm{skel}(\nu)$ are shown in red, namely the two quadratic vertices dual to the faces incident to the parent marker and the edges incident to those vertices. One criterion for a face to be a candidate for becoming the outer face due to Bertolazzi et al. is that there has to be a path from this face to the outer face in the face sink graph. This holds true for both faces incident to the parent marker in $\mathrm{skel}(\nu)$, but not in $\mathrm{skel}(\mu)$. Therefore the arc (μ, ν) is not contracted but the arc (λ, μ) is contracted. This leads to the decomposition tree shown in (c). See also Fig. 3(c) and (d) for a larger example.

Lemma 6. *Let G be a biconnected digraph with a single source s and let $e^\star$ denote an edge incident to s. There exists a decomposition tree $\mathcal{T}$ that (i) represents all upward planar embeddings of G in which $e^\star$ is the leftmost edge around s, and (ii) the children of all P-nodes in $\mathcal{T}$ can be arbitrarily permuted. (iii) the skeletons of all R-nodes in $\mathcal{T}$ can be reversed.*

We call the decomposition tree $\mathcal{T}$ the *UP-tree of G relative to $e^\star$*.

4.3   Computation in Linear Time

Let G be a biconnected digraph with a single source s and let $e^\star$ denote an edge incident to s. Recall that the construction of the UP-tree $\mathcal{T}$ of G relative to $e^\star$ consists of the following seven steps. 1. Construct the SPQR-tree $\mathcal{T}$ of G in linear time [22,23]. 2. For each pair of vertices u, v that are the poles of a marker in some skeleton of $\mathcal{T}$, we have to determine whether $u < v$ in G.
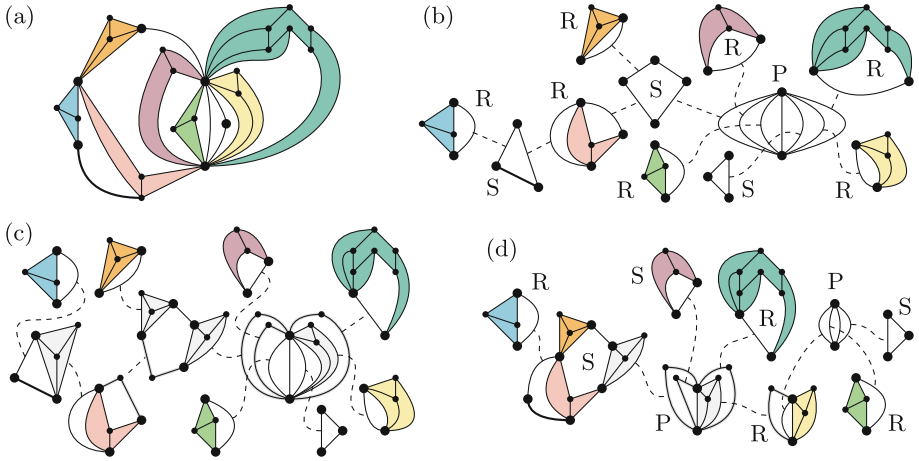
Fig. 3. Construction of the UP-tree. An upward planar biconnected single-source graph (a), the SPQR-tree of its underlying undirected graph (b) with the Q-nodes omitted, the result of replacing virtual edges with markers (c) and the UP-tree after splitting P-nodes and contracting arcs (d).

To compute this information for all pairs in linear time, we use a union-find-based technique described by Bläsius et al. [8]. Process all skeletons of $\mathcal{T}$ and for every pair of poles u, v that is encountered register v as a *candidate* at u and register u as a candidate at v. Next, initialize every vertex of G in its own singleton set. Then, process each vertex u in some reverse topological order of G. Unify the singleton set of u with the sets of its direct descendants in G. Now for any candidate v stored at u we can query in whether u and v belong to the same set, which is equivalent to $u < v$. Note that the operands to all unify operations are completely determined by the structure of G. We exploit this fact to run the linear-time union-find algorithm due to Gabow and Tarjan [21]. 3. For each arc $a = (\mu, \nu)$ of $\mathcal{T}$, decide whether the poles of a are sources, sinks or internal vertices in $G(\mu)$ and $G(\nu)$. This information can be found using a simple bottom-up technique. We first compute the indegree and outdegree of every node of G. We then perform a depth-first traversal of $\mathcal{T}$. We maintain a list of the number of incoming and outgoing edges for each node seen so far, which is updated when a Q-node is visited. Upon entering a subtree, we store these numbers for the poles of the arc leaving the subtree at the root of the subtree. Upon leaving a subtree, we can now calculate the differences between the current numbers and the stored numbers, which gives the in- and outdegree of the poles in the graph $G(\mu)$. Using the in- and outdegree of the poles in G computed earlier, we can also compute the in- and outdegree of the poles in $G(\nu)$. This step clearly takes linear time. 4. In each skeleton, replace all virtual edges with their respective markers. With the information that was computed in the previous step and the fact that all markers have constant size this step is feasible in linear time. 5. Construct a configuration of $\mathcal{T}$ by running

the linear-time upward planar embedding algorithm of Bertolazzi et al. [4] on every skeleton. Because the size of all skeletons is linear in the size of G this step takes linear time. 6. Perform P-node splits. The running time spent on one P-node is clearly linear in the size of its skeleton. This gives linear running time overall. 7. Perform arc contractions. The upward planarity test for fixed embeddings due to Bertolazzi et al. runs in linear time. Contracting an arc is feasible in constant time. This gives linear running time overall.

Theorem 3. *Let G be a biconnected digraph with a single source s and let $e^\star$ denote an edge incident to s. The UP-tree $\mathcal{T}$ of G relative to $e^\star$ is a decomposition tree whose internal nodes are (i) S-nodes whose skeletons have a fixed embedding, (ii) R-nodes whose skeletons have a fixed embedding up to reversal, or (iii) P-nodes where the markers can be arbitrarily permuted in the skeleton and whose leaves are Q-nodes that offer no embedding choice. The configurations of $\mathcal{T}$ correspond bijectively to the upward planar embeddings of G where $e^\star$ appears leftmost around s. Moreover, $\mathcal{T}$ can be computed in linear time.*

5   Partial Upward Embedding

In this section we apply the UP-tree to solve the partial upward embedding problem in quadratic time. A *partially embedded graph* is a tuple $(G, H, \mathcal{H})$, where G is a planar graph, H is a subgraph of G and $\mathcal{H}$ is a planar embedding of H. An embedding $\mathcal{G}$ of G *extends* the partial embedding $\mathcal{H}$ if all edges e, f, g in H that share a common endpoint v appear in the same cyclic order around v in $\mathcal{G}$ and $\mathcal{H}$. The *partial embedding problem* asks whether there exists an embedding $\mathcal{G}$ of G that extends $\mathcal{H}$. Angelini et al. solve the partial embedding problem in linear time [1]. The algorithm considers every triple of edges (e, f, g) in H that share a common endpoint v and enforces the constraints imposed by these edges in the SPQR-tree $\mathcal{T}$. Note that e, f, g each correspond to a Q node in $\mathcal{T}$. Because $\mathcal{T}$ is a tree there is exactly one node μ in $\mathcal{T}$ so that the paths from μ to these Q nodes are disjoint. The relative order of e, f, g in the embedding represented by $\mathcal{T}$ is determined by the embedding of $\mathrm{skel}(\mu)$. If $\mathrm{skel}(\mu)$ offers no embedding choice (as in S nodes) determine whether the ordering of e, f, g given by $\mathcal{H}$ is the same as the one given by the unique embedding of $\mathrm{skel}(\mu)$. If not, reject the instance. If $\mathrm{skel}(\mu)$ has two possible embeddings (as in R nodes) the ordering of e, f, g given by $\mathcal{H}$ fixes one of the two embeddings of $\mathrm{skel}(\mu)$ as the only candidate. Finally, if μ is a P node the ordering of e, f, g given by $\mathcal{H}$ restricts the set of admissible permutations of the virtual edges in $\mathrm{skel}(\mu)$. The algorithm collects all these constraints and checks whether they can be fulfilled at the same time.

A *partially embedded upward graph* is defined as a tuple $(G, H, \mathcal{H})$, where G is an upward planar graph, H is a subgraph of H and $\mathcal{H}$ is an upward planar embedding of H. Note that UP-trees have all properties of SPQR-trees that are needed in the algorithm described above. In particular, the markers in P-nodes may be arbitrarily permuted, R-nodes may be reversed and all other nodes offer no embedding choice. Hence, we use the UP-tree as a drop-in replacement for

the SPQR-tree in the algorithm of Angelini et al. to obtain an algorithm that solves the partial upward embedding problem. Note that the UP-tree is rooted at some edge that must be embedded as the leftmost edge around the source of the graph. We may have to try a linear number of candidate edges in the worst case. This gives the following.

Theorem 4. *The partial upward embedding problem can be solved in quadratic running time for biconnected single-source digraphs.*

6   Conclusion

We have developed the UP-tree, which is an SPQR-tree-like embedding representation for upward planarity. We expect that the UP-tree is a valuable tool that makes it possible to translate existing constrained planar embedding algorithms that use SPQR-trees to the upward planar setting. As an example, we have demonstrated how to use the UP-tree as a drop-in replacement for the SPQR-tree in the partial embedding extension problem, solving the previously open partial upward embedding extension problem for the biconnected single-source case.

References

1. Angelini, P., Di Battista, G., Frati, F., Jelínek, V., Kratochvíl, J., Patrignani, M., Rutter, I.: Testing planarity of partially embedded graphs. ACM Trans. Algorithms **11**(4), 32:1–32:42 (2015). https://doi.org/10.1145/2629341
2. Angelini, P., Di Battista, G., Patrignani, M.: Finding a minimum-depth embedding of a planar graph in $o(n^4)$ time. Algorithmica **60**(4), 890–937 (2011). https://doi.org/10.1007/s00453-009-9380-6
3. Bertolazzi, P., Di Battista, G., Liotta, G., Mannino, C.: Upward drawings of triconnected digraphs. Algorithmica **12**(6), 476–497 (1994)
4. Bertolazzi, P., Di Battista, G., Mannino, C., Tamassia, R.: Optimal upward planarity testing of single-source digraphs. SIAM J. Comput. **27**(1), 132–169 (1998)
5. Bläsius, T., Kobourov, S.G., Rutter, I.: Simultaneous embedding of planar graphs. In: Tamassia, R. (ed.) Handbook of Graph Drawing and Visualization, Discrete Mathematics and its Applications, pp. 349–373. CRC Press (2014)
6. Bläsius, T., Lehmann, S., Rutter, I.: Orthogonal graph drawing with inflexible edges. Comput. Geom. **55**, 26–40 (2016). https://doi.org/10.1016/j.comgeo.2016.03.001
7. Bläsius, T., Rutter, I., Wagner, D.: Optimal orthogonal graph drawing with convex bend costs. ACM Trans. Algorithms **12**(3), 33:1–33:32 (2016). https://doi.org/10.1145/2838736
8. Bläsius, T., Karrer, A., Rutter, I.: Simultaneous embedding: edge orderings, relative positions, cutvertices. Algorithmica **80**(4), 1214–1277 (2018)
9. Brückner, G., Himmel, M., Rutter, I.: An SPQR-tree-like embedding representation for upward planarity. CoRR abs/1908.00352v1 (2019). https://arxiv.org/abs/1908.00352v1

10. Brückner, G., Rutter, I.: Partial and constrained level planarity. In: Klein, P.N. (ed.) Proceedings of 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017), pp. 2000–2011. SIAM (2017)

11. Brückner, G., Rutter, I., Stumpf, P.: Level planarity: transitivity vs. even crossings. In: Biedl, T., Kerren, A. (eds.) GD 2018. LNCS, vol. 11282, pp. 39–52. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-04414-5_3

12. Da Lozzo, G., Di Battista, G., Frati, F.: Extending upward planar graph drawings. CoRR abs/1902.06575 (2019)

13. Da Lozzo, G., Jelínek, V., Kratochvíl, J., Rutter, I.: Planar embeddings with small and uniform faces. In: Ahn, H.-K., Shin, C.-S. (eds.) ISAAC 2014. LNCS, vol. 8889, pp. 633–645. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-13075-0_50

14. Da Lozzo, G., Rutter, I.: Approximation algorithms for facial cycles in planar embeddings. In: Hsu, W.L., Lee, D.T., Liao, C.S. (eds.) Proceedings of the 29th International Symposium on Algorithms and Computation (ISAAC 2018). LIPIcs, vol. 123, pp. 41:1–41:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2018). https://doi.org/10.4230/LIPIcs.ISAAC.2018.41

15. Di Battista, G., Tamassia, R.: Incremental planarity testing. In: Proceedings of the 30th Annual Symposium on Foundations of Computer Science, pp. 436–441, October 1989. https://doi.org/10.1109/SFCS.1989.63515

16. Di Battista, G., Tamassia, R.: On-line graph algorithms with SPQR-trees. In: Paterson, M.S. (ed.) ICALP 1990. LNCS, vol. 443, pp. 598–611. Springer, Heidelberg (1990). https://doi.org/10.1007/BFb0032061

17. Di Battista, G., Tamassia, R.: On-line maintenance of triconnected components with SPQR-trees. Algorithmica **15**(4), 302–318 (1996). https://doi.org/10.1007/BF01961541

18. Didimo, W., Liotta, G., Patrignani, M.: Bend-minimum orthogonal drawings in quadratic time. In: Biedl, T., Kerren, A. (eds.) GD 2018. LNCS, vol. 11282, pp. 481–494. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-04414-5_34

19. Feng, Q.-W., Cohen, R.F., Eades, P.: Planarity for clustered graphs. In: Spirakis, P. (ed.) ESA 1995. LNCS, vol. 979, pp. 213–226. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-60313-1_145

20. Fulek, R., Pelsmajer, M.J., Schaefer, M., Štefankovič, D.: Hanani-Tutte, monotone drawings, and level-planarity. In: Pach, J. (ed.) Thirty Essays on Geometric Graph Theory, pp. 263–287. Springer, New York (2013). https://doi.org/10.1007/978-1-4614-0110-0_14

21. Gabow, H.N., Tarjan, R.E.: A linear-time algorithm for a special case of disjoint set union. J. Comput. Syst. Sci. **30**(2), 209–221 (1985)

22. Gutwenger, C., Mutzel, P.: A linear time implementation of SPQR-trees. In: Marks, J. (ed.) GD 2000. LNCS, vol. 1984, pp. 77–90. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44541-2_8

23. Hopcroft, J.E., Tarjan, R.E.: Dividing a graph into triconnected components. SIAM J. Comput. **2**(3), 135–158 (1973)

24. Hutton, M.D., Lubiw, A.: Upward planar drawing of single-source acyclic digraphs. SIAM J. Comput. **25**(2), 291–311 (1996)

25. Jelínek, V., Kratochvíl, J., Rutter, I.: A Kuratowski-type theorem for planarity of partially embedded graphs. Comput. Geom.: Theory Appl. **46**(4), 466–492 (2013)

26. Jünger, M., Leipert, S.: Level planar embedding in linear time. In: Kratochvíyl, J. (ed.) GD 1999. LNCS, vol. 1731, pp. 72–81. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-46648-7_7

27. Mac Lane, S.: A structural characterization of planar combinatorial graphs. Duke Math. J. **3**(3), 460–472 (1937). https://doi.org/10.1215/S0012-7094-37-00336-3
28. Platt, C.R.: Planar lattices and planar graphs. J. Comb. Theory Ser. B **21**(1), 30–39 (1976)
29. Randerath, B., Speckenmeyer, E., Boros, E., Hammer, P., Kogan, A., Makino, K., Simeone, B., Cepek, O.: A satisfiability formulation of problems on level graphs. Electron. Notes Discret. Math. **9**, 269–277 (2001). lICS 2001 Workshop on Theory and Applications of Satisfiability Testing (SAT 2001)
30. Tutte, W.T.: Connectivity in Graphs. University of Toronto Press, Toronto (1966)