An SPQR-Tree-Like Embedding Representation for Level Planarity

Guido Brückner
Karlsruhe Institute of Technology (KIT), Germany
brueckner@kit.edu

Ignaz Rutter
Universität Passau, Germany
ignaz.rutter@uni-passau.de

Abstract

An SPQR-tree is a data structure that efficiently represents all planar embeddings of a biconnected planar graph. It is a key tool in a number of constrained planarity testing algorithms, which seek a planar embedding of a graph subject to some given set of constraints.

We develop an SPQR-tree-like data structure that represents all level-planar embeddings of a biconnected level graph with a single source, called the LP-tree, and give a simple algorithm to compute it in linear time. Moreover, we show that LP-trees can be used to adapt three constrained planarity algorithms to the level-planar case by using them as a drop-in replacement for SPQR-trees.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Data structures design and analysis; Theory of computation → Graph algorithms analysis

Keywords and phrases SPQR-tree, Level planarity, Partial drawings, Simultaneous drawings

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2020.8

Related Version A full version of the paper is available at [10], `https://arxiv.org/abs/2009.12309`.

Funding This work was partially supported by DFG grant Ru 1903/3-1.

1 Introduction

Testing planarity of a graph and finding a planar embedding, if one exists, are classical algorithmic problems. For visualization purposes, it is often desirable to draw a graph subject to certain additional constraints, e.g., finding orthogonal drawings [28] or symmetric drawings [21], or inserting an edge into an embedding so that few edge crossings are caused [20]. Historically, these problems have been considered for embedded graphs. More recent research has attempted to optimize not only one fixed embedding, but instead to optimize across all possible planar embeddings of a graph. This includes (i) orthogonal drawings [9], (ii) simultaneous embeddings, where one seeks to embed two planar graphs that share a common subgraph such that they induce the same embedding on the shared subgraph (see [8] for a survey), (iii) simultaneous orthogonal drawings [3], (iv) embeddings where some edge intersections are allowed [1], (v) inserting an edge [20], a vertex [13], or multiple edges [14] into an embedding, (vi) partial embeddings, where one insists that the embedding extends a given embedding of a subgraph [4], and (vii) finding minimum-depth embeddings [6, 7].

The common tool in all of these recent algorithms is the SPQR-tree data structure, which efficiently represents all planar embeddings of a biconnected planar graph G by breaking down the complicated task of choosing a planar embedding of G into the task of independently choosing a planar embedding for each triconnected component of G [16, 17, 18, 22, 26, 29]. This is a much simpler task since the triconnected components have a very restricted structure, and so the components offer only basic, well-structured choices.

An *upward planar drawing* is a planar drawing where each edge is represented by a y-monotone curve. For a level graph $G = (V, E)$, which is a directed graph where each vertex $v \in V$ is assigned to a level $\ell(v)$ such that for each edge $(u, v) \in E$ it is $\ell(u) < \ell(v)$, a *level-planar drawing* is an upward planar drawing where each vertex v is mapped to a point on the horizontal line $y = \ell(v)$. Level planarity can be tested in linear time [19, 24, 25, 27]. Recently, the problem of extending partial embeddings for level-planar drawings has been studied [12]. While the problem is NP-hard in general, it can be solved in polynomial time for single-source graphs. Very recently, an SPQR-tree-like embedding representation for upward planarity has been used to extend partial upward embeddings [11]. The construction crucially relies on an existing decomposition result for upward planar graphs [23]. No such result exists for level-planar graphs. Moreover, the level assignment leads to components of different "heights", which makes our decompositions significantly more involved.

Contribution. We develop the LP-tree, an analogue of SPQR-trees for level-planar embeddings of level graphs with a single source whose underlying undirected graph is biconnected. It represents the choice of a level-planar embedding of a level-planar graph by individual embedding choices for certain components of the graph, for each of which the embedding is either unique up to reflection, or allows to arbitrarily permute certain subgraphs around two pole vertices. Its construction is based on suitably modifying the SPQR-tree of G, which represents all planar embeddings of G, not just the level-planar ones, such that, eventually, the modified tree represents exactly the level-planar drawings of G. See Figure 1 (a, b) for examples of how level planarity is more restrictive than planarity. The size of the LP-tree is linear in the size of G and it can be computed in linear time. The LP-tree is a useful tool that unlocks the large amount of SPQR-tree-based algorithmic knowledge for easy translation to the level-planar setting. In particular, we obtain linear-time algorithms for partial and constrained level planarity for biconnected single-source level graphs, which improves upon the $O(n^2)$-time algorithm known to date [12]. Further, we describe the first efficient algorithm for the simultaneous level planarity problem when the shared graph is a biconnected single-source level graph. Proofs of marked statements ($\star$) can be found in the full version [10].

2 Preliminaries

Let $G = (V, E)$ be a connected level graph. For each vertex $v \in V$ let $d(v) \geq \ell(v)$ denote the *demand* of v. Demands provide an interface to model the restrictions imposed on the embeddings of one biconnected component by other biconnected components; see Figure 1 (c). An *apex* of some vertex set $V' \subseteq V$ is a vertex $v \in V'$ whose level is maximum. The *demand* of V', denoted by $d(V')$, is the maximum demand of a vertex in V'. An apex of a face f is an apex of the vertices incident to f. A *planar drawing* of G is a topological planar drawing of the underlying undirected graph of G. Planar drawings are *equivalent* if they can be continuously transformed into each other without creating intermediate intersections. A *planar embedding* is an equivalence class of equivalent planar drawings. A *path* is a sequence $(v_1, v_2, \ldots, v_j)$ of vertices so that for $1 \leq i < j$ either (v_i, v_{i+1}) or (v_{i+1}, v_i) is an edge in E. A *directed path* is a sequence $(v_1, v_2, \ldots, v_j)$ of vertices so that for $1 \leq i < j$ it is $(v_i, v_{i+1}) \in E$. A vertex u *dominates* a vertex v if there exists a directed path from u to v. A vertex is a *sink* if it dominates no vertex except for itself. A vertex is a *source* if it is dominated by no vertex except for itself. An *st-graph* is a graph with a single source and a single sink, usually denoted by s and t, respectively. Throughout this paper all graphs are assumed to

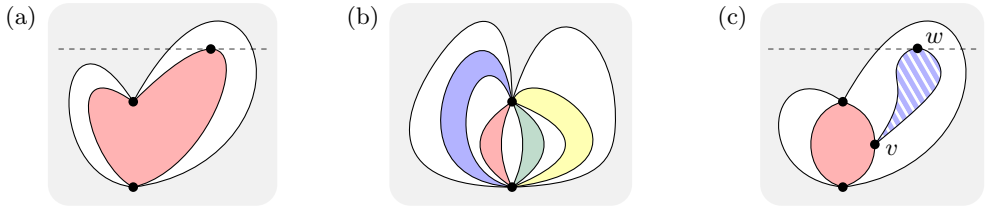

Figure 1 In (a), the height of the red component makes it impossible to flip it. In (b), note that the red and green components can be exchanged, as can the blue and yellow components, but neither the blue nor the yellow component can be embedded between the red and green component. In (c), set the demand of v as $d(v) = \ell(w)$ in the LP-tree that represents the graph that consists of the red and gray part (but not the striped blue part). This models the restriction imposed on the embedding of the red subgraph by the striped blue biconnected component.

have a single source s. For the remainder of this paper we restrict our considerations to level-planar drawings of G where each vertex $v \in V$ that is not incident to the outer face is incident to some inner face f so whose apex a of the set of vertices on the boundary of f satisfies $d(v) < \ell(a)$. We will use demands in Section 4 to restrict the admissible embeddings of biconnected components in the presence of cutvertices. Note that setting $d(v) = \ell(v)$ for each $v \in V$ gives the conventional definition of level-planar drawings. A planar embedding Γ of G is *level planar* if there exists a level-planar drawing of G with planar embedding Γ. We then call Γ a *level-planar embedding*. For single-source level graphs, level-planar embeddings are equivalence classes of topologically equivalent level-planar drawings.

▶ **Lemma 1** (⋆). *The level-planar drawings of a single-source level graph correspond bijectively to its level-planar combinatorial embeddings with s on the outer face.*

To make some of the subsequent arguments easier to follow, we preprocess our input level graph G on k levels to a level graph G' on $d(V) + 1$ levels as follows. We obtain G' from G by adding a new vertex t on level $d(V) + 1$ with demand $d(t) = d(V) + 1$, connecting it to all vertices on level k and adding the edge (s, t). Note that G' is generally not an st-graph. The embeddings of G' where the edge (s, t) is incident to the outer face and the embeddings of G are, in a sense, equivalent.

▶ **Lemma 2** (⋆). *An embedding Γ of G is level-planar if and only if there exists a level-planar embedding Γ' of G' that extends Γ where (s, t) is incident to the outer face.*

To represent all level-planar embeddings of G, it is sufficient to represent all level-planar embeddings of G' and to remove t and its incident edges from all embeddings. It is easily observed that if G is a biconnected single-source graph, then so is G'. We assume from now on that the vertex set of our input graph G has a unique apex t and that G contains the edge (s, t). We still refer to the highest level as level k, i.e., the apex t lies on level k. To prove that embeddings are level planar we present some further tools, including a novel characterization of level planarity, in the full version.

Our description of decomposition trees follows Angelini et al. [2]. Let G be a biconnected graph. A *separation pair* is a subset $\{u, v\} \subseteq V$ whose removal from G disconnects G. Let $\{u, v\}$ be a separation pair and let H_1, H_2 be two subgraphs of G with $H_1 \cup H_2 = G$ and $H_1 \cap H_2 = \{u, v\}$. Define the tree $\mathcal{T}$ that consists of two *nodes* μ_1 and μ_2 connected by an undirected *arc* as follows. For $i = 1, 2$ node μ_i is equipped with a multigraph $\text{skel}(\mu_i) = H_i + e_i$, called its *skeleton*, where $e_i = (u, v)$ is called a *virtual* edge. The arc (μ_1, μ_2) links the two virtual edges e_i in $\text{skel}(\mu_i)$ with each other. We also say that the virtual edge e_1 *corresponds*

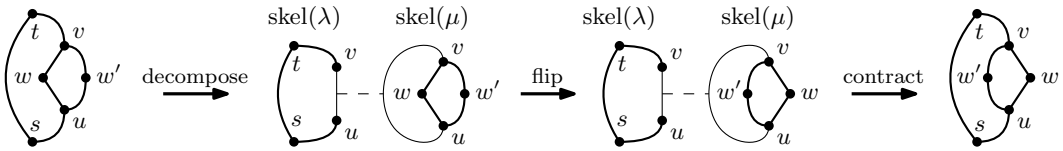

Figure 2 Decompose the embedded graph G on the left at the separation pair u, v. This gives the center-left decomposition tree whose skeletons are embedded as well. Reflecting the embedding of skel(μ) or, equivalently, flipping (λ, μ), yields the same decomposition tree with a different embedding of skel(μ). Contract (λ, μ) to obtain the embedding on the right.

to μ_2 and likewise that e_2 corresponds to μ_1. The idea is that skel(μ_1) provides a more abstract view of G where e_1 serves as a placeholder for H_2. More generally, there is a bijection $\text{corr}_\mu \colon E(\text{skel}(\mu)) \to N(\mu)$ that maps every virtual edge of skel(μ) to a neighbor of μ in $\mathcal{T}$, and vice versa. If it is $\text{corr}_\mu((u, v)) = \nu$, then ν is said to have *poles* u and v in μ. If μ is clear from the context we simply say that ν has poles u, v. When the underlying graph is a level graph, we assume $\ell(u) \leq \ell(v)$ without loss of generality. For an arc (ν, μ) of $\mathcal{T}$, the virtual edges e_1, e_2 with $\text{corr}_\mu(e_1) = \nu$ and $\text{corr}_\nu(e_2) = \mu$ are called *twins*, and e_1 is called the *twin* of e_2 and vice versa. This procedure is called a *decomposition*, see Figure 2 on the left. It can be re-applied to skeletons of the nodes of $\mathcal{T}$, which leads to larger trees with smaller skeletons. A tree obtained in this way is a *decomposition tree* of G. A decomposition can be undone by *contracting* an arc (μ_1, μ_2) of $\mathcal{T}$, forming a new node μ with a larger skeleton as follows. Let e_1, e_2 be twin edges in skel(μ_1), skel(μ_2). The skeleton of μ is the union of skel(μ_1) and skel(μ_2) without the two twin edges e_1, e_2. Contracting all arcs of a decomposition tree of G results in a decomposition tree consisting of a single node whose skeleton is G. See Figure 2 on the right. Let μ be a node of a decomposition tree with a virtual edge e with $\text{corr}_\mu(e) = \nu$. The *expansion graph* of e and ν in μ, denoted by $G(e)$ and $G(\mu, \nu)$, respectively, is the graph obtained by removing the twin of e from skel(ν) and contracting all arcs in the subtree that contains ν.

Each skeleton of a decomposition tree of G is a minor of G. So if G is planar, each skeleton of a decomposition tree $\mathcal{T}$ of G is planar as well. If (μ_1, μ_2) is an arc of $\mathcal{T}$, and skel(μ_1) and skel(μ_2) have fixed planar embeddings Γ_1 and Γ_2, respectively, then the skeleton of the node μ obtained from contracting (μ_1, μ_2) can be equipped with an embedding Γ by merging these embeddings along the twin edges corresponding to (μ_1, μ_2); see Figure 2 on the right. This requires at least one of the virtual edges e_1 in skel(μ_1) with $\text{corr}_{\mu_1}(e_1) = \mu_2$ or e_2 in skel(μ_2) with $\text{corr}_{\mu_2}(e_2) = \mu_1$ to be incident to the outer face. If we equip every skeleton with a planar embedding and contract all arcs, we obtain a planar embedding of G. This embedding is independent of the order of the edge contractions. Thus, every decomposition tree $\mathcal{T}$ of G represents (not necessarily all) planar embeddings of G by choosing a planar embedding of each skeleton and contracting all arcs. Let e_{ref} be an edge of G. Rooting $\mathcal{T}$ at the unique node μ_{ref} whose skeleton contains the real edge e_{ref} identifies a unique parent virtual edge in each of the remaining nodes; all other virtual edges are called *child virtual edges*. The arcs of $\mathcal{T}$ become directed from the parent node to the child node. Restricting the embeddings of the skeletons so that the parent virtual edge (the edge e_{ref} in case of μ_{ref}) is incident to the outer face, we obtain a representation of (not necessarily all) planar embeddings of G where e_{ref} is incident to the outer face. Let μ be a node of $\mathcal{T}$ and let e be a child virtual edge in skel(μ) with $\text{corr}_\mu(e) = \nu$. Then the expansion graph $G(\mu, \nu)$ is simply referred to as $G(\nu)$.

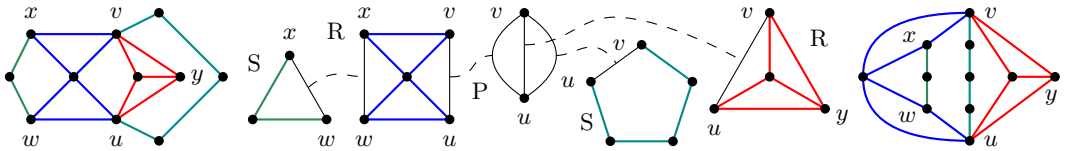

Figure 3 A planar graph on the left and its SPQR-tree in the middle. The five nodes of the SPQR-tree are represented by their respective skeleton graphs. Dashed edges connect twin virtual edges and colored edges correspond to Q-nodes. The embedding of the graph on the right is obtained by flipping the embedding of the blue R-node and swapping the middle and right edge of the P-node.

The *SPQR-tree* is a special decomposition tree whose skeletons are precisely the triconnected components of G. It has four types of nodes: S-nodes, whose skeletons are cycles, P-nodes, whose skeletons consist of three or more parallel edges between two vertices, and R-nodes, whose skeletons are simple triconnected graphs. Finally, a Q-node has a skeleton consisting of two vertices connected by one real and by one virtual edge. This means that in the skeletons of all other node types all edges are virtual. In an SPQR-tree the embedding choices are of a particularly simple form. The skeletons of Q- and S-nodes have a unique planar embedding (not taking into account the choice of the outer face). The child virtual edges of P-node skeletons may be permuted arbitrarily, and the skeletons of R-nodes are 3-connected, and thus have a unique planar embedding up to reflection. We call this the *skeleton-based* embedding representation. There is also an *arc-based* embedding representation. Here the embedding choices are (i) the linear order of the children in each P-node, and (ii) for each arc (λ, μ) whose target μ is an R-node whether the embedding of the expansion graph $G(\mu)$ should be *flipped*. To obtain the embedding of G, we contract the edges of $\mathcal{T}$ bottom-up. Consider the contraction of an arc (λ, μ) whose child μ used to be an R-node in $\mathcal{T}$. At this point, $\text{skel}(\mu)$ is equipped with a planar embedding Γ_μ. If the embedding should be flipped, we reflect the embedding Γ_μ before contracting (λ, μ), otherwise we simply contract (λ, μ). The arc-based and the skeleton-based embedding representations are equivalent. See Figure 3 and Figure 6 (a,b) for examples of a planar graph and its SPQR-tree.

3 A Decomposition Tree for Level Planarity

We construct a decomposition tree of a given single-source level graph G whose underlying undirected graph is biconnected that represents all level-planar embeddings of G, called the *LP-tree*. As noted in the Preliminaries, we assume that G has a unique apex t, for which $\ell(t) = d(t)$ holds true. The LP-tree for G is constructed based on the SPQR-tree for G. We keep the notion of S-, P-, Q- and R-nodes and construct the LP-tree so that the nodes behave similarly to their namesakes in the SPQR-tree. The skeleton of a P-node consists of two vertices that are connected by at least three parallel virtual edges that can be arbitrarily permuted. The skeleton of an R-node μ is equipped with a *reference embedding* Γ_μ, and the choice of embeddings for such a node is limited to either Γ_μ or its reflection. Unlike in SPQR-trees, the skeleton of μ need not be triconnected, instead it can be an arbitrary biconnected planar graph. The embedding of R-node skeletons being fixed up to reflection allows us to again use the equivalence of the arc-based and the skeleton-based embedding representations.

The construction of the LP-tree starts out with an SPQR-tree $\mathcal{T}$ of G. Explicitly label each node of $\mathcal{T}$ as an S-, P-, Q- or R-node. This way, we can continue to talk about S-, P-, Q- and R-nodes of our decomposition tree even when they no longer have their

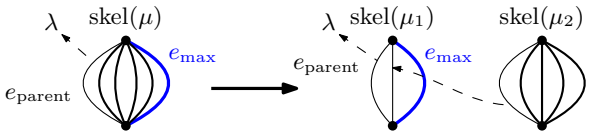

Figure 4 Result of a P-node μ split with parent λ and child with maximum height ν. Note that after the split, μ_1 is an R-node and μ_2 has one less child than μ had.

defining properties in the sense of SPQR-trees. Assume the edge (s,t) to be incident to the outer face of every level-planar drawing of G (Lemma 2), i.e., consider $\mathcal{T}$ rooted at the Q-node corresponding to (s,t). The construction of our decomposition tree works in two steps. First, decompose the graph further by decomposing P-nodes in order to disallow permutations that lead to embeddings that are not level planar. Second, contract arcs of the decomposition tree, each time fixing a reference embedding for the resulting node, so that we can consider it as an R-node, such that the resulting decomposition tree represents exactly the level-planar embeddings of G. The remainder of this section is structured as follows. The details and correctness of the first step are given in Section 3.1. Section 3.2 gives the algorithm for constructing the final decomposition tree $\mathcal{T}$. It follows from the construction that all embeddings it represents are level-planar, and Section 3.3 shows that, conversely, it also represents every level-planar embedding. In the full version, we present a linear-time implementation of the construction algorithm.

3.1 P-Node Splits

In SPQR-trees, the children of P-nodes can be arbitrarily permuted. We would like P-nodes of the LP-tree to have the same property. Hence, we decompose skeletons of P-nodes to disallow orders that lead to embeddings that are not level planar. The decomposition is based on the height of the child virtual edges, which we define as follows. Let μ be a node of a rooted decomposition tree and let u and v be the poles of μ. Define $V(\mu) = V(G(\mu)) \setminus \{u, v\}$. The *height* of μ and of the child virtual edge e with $\mathrm{corr}(e) = \mu$ is $d(\mu) = d(e) = d(V(\mu))$. If μ is a leaf Q-node it is $V(\mu) = \emptyset$ and we define the height of μ as $\ell(u)$.

Now let μ be a P-node, and let Γ be a level-planar embedding of G. The embedding Γ induces a linear order of the child virtual edges of μ. This order can be obtained by splitting the combinatorial embedding of $\mathrm{skel}(\mu)$ around u at the parent edge. Then the following is true.

▶ **Lemma 3** (⋆). *Let $\mathcal{T}$ be a decomposition tree of G, let μ be a P-node of $\mathcal{T}$ with poles u, v, and let $e_{\max}$ be a child virtual edge of μ with maximal height. Further, let Γ be a level-planar embedding of G that is represented by $\mathcal{T}$. If the height of $e_{\max}$ is at least $\ell(v)$, then $e_{\max}$ is either the first or the last edge in the linear ordering of the child virtual edges induced by Γ.*

Lemma 3 motivates the following modification of a decomposition tree $\mathcal{T}$. Take a P-node μ with poles u, v that has a child edge whose height is at least $\ell(v)$. Denote by λ the parent of μ. Further, let $e_{\max}$ be a child virtual edge with maximum height and let e_{parent} denote the parent edge of $\mathrm{skel}(\mu)$. Obtain a new decomposition tree $\mathcal{T}'$ by splitting μ into two nodes μ_1 and μ_2 representing the subgraph H_1 consisting of the edges $e_{\max}$ and e_{parent}, and the subgraph H_2 consisting of the remaining child virtual edges, respectively; see Figure 4. Note that the skeleton of μ_1, which corresponds to H_1, has only two child virtual edges. We therefore define it to be an R-node. Moreover, observe that in any embedding of $\mathrm{skel}(\mu)$

that is obtained from choosing embeddings for skel(μ_1) and skel(μ_2) and contracting the arc (μ_1, μ_2), the edge $e_{\max}$ is the first or last child edge. Conversely, because μ_2 is a P-node, all embeddings where $e_{\max}$ is the first or last child edge are still represented by $\mathcal{T}'$. Apply this decomposition iteratively, creating new R-nodes on the way, until each P-node μ with poles u and v has only child virtual edges e that have height at most $\ell(v) - 1$. We say that a node ν with poles x, y has *I shape* when the height of $G(\nu)$ is less than $\ell(y)$. The following theorem sets the stage to prove that after this decomposition, the children of P-nodes can be arbitrarily permuted.

▶ **Theorem 4.** *Let G be a biconnected single-source graph with unique apex t. There exists a decomposition tree $\mathcal{T}$ that represents all level-planar embeddings of G such that all children of P-nodes in $\mathcal{T}$ have I shape.*

This ensures that P-nodes in our decomposition of level-planar graphs work analogously to those of SPQR-trees for planar graphs. Namely, if we have a level-planar embedding Γ of G and consider a new embedding Γ' that is obtained from Γ by reordering the children of P-nodes, then also Γ' is level-planar. Hence, in our decomposition the children of P-nodes can be arbitrarily permuted. See Figure 6 (b,c) for an example and the full version for a complete proof.

▶ **Theorem 5.** *Let G be a biconnected single-source graph with a unique apex. There exists a decomposition tree $\mathcal{T}$ that (i) represents all level-planar embeddings of G (plus some planar, non-level-planar ones), and (ii) if all skeletons of the nodes of $\mathcal{T}$ are embedded so that contracting all arcs of $\mathcal{T}$ yields a level-planar embedding, then the children of all P-nodes in $\mathcal{T}$ can be arbitrarily permuted and then contracting all arcs of $\mathcal{T}$ still yields a level-planar embedding of G.*

3.2 Arc Processing

In this section, we finish the construction of the LP-tree. The basis of our construction is the decomposition tree $\mathcal{T}$ from Theorem 4, which represents a subset of the planar embeddings of G that contains all level-planar embeddings, and moreover all children of P-nodes have I shape. We now restrict $\mathcal{T}$ even further until it represents exactly the level-planar embeddings of G. As of now, all R-node skeletons have a planar embedding that is unique up to reflection, as they are either triconnected or consist of only three parallel edges. By assumption, G is level-planar, and there exists a level-planar embedding Γ of G. Recall that our definition of level-planar embeddings involves demands. Computing a level-planar embedding Γ of G with demands reduces to computing a level-planar embedding of the supergraph G' of G obtained from G by attaching to each vertex v of G with $d(v) > \ell(v)$ an edge to a vertex v' with $\ell(v') = d(v)$ without demands. Because G' is a single-source graph whose size is linear in the size of G this can be done in linear time [15]. We equip the skeleton of each node μ with the reference embedding Γ_μ such that contracting all arcs yields the embedding Γ. For the remainder of this section we will work with the arc-based embedding representation. As a first step, we contract any arc (λ, μ) of $\mathcal{T}$ where λ is an R-node and μ is an S-node and label the resulting node as an R-node. Note that, since S-nodes do not offer any embedding choices, this does not change the embeddings that are represented by $\mathcal{T}$. This step makes the correctness proof easier. Any remaining arc (λ, μ) of $\mathcal{T}$ is contracted based upon two properties of μ, namely the height of $G(\mu)$ and the space around μ in the level-planar embedding Γ, which we define next. The resulting node is again labeled as an

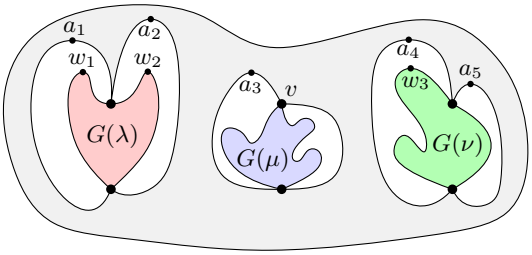

Figure 5 The height of $G(\lambda)$ is at least $\ell(w_1) = \ell(w_2)$, the height of $G(\mu)$ is at most $\ell(v) - 1$ and the height of $G(\nu)$ is at least $\ell(w_3)$. The space around λ is $\ell(a_1)$, the space around μ is $\ell(v)$ and the space around ν is $\ell(a_5)$.

R-node. Let μ be a node of $\mathcal{T}$ with poles u and v. We denote by $\Gamma \circ \mu$ the embedding obtained from Γ by contracting $G(\mu)$ to the single edge $e = (u, v)$. We call the faces f_1, f_2 of Γ that induce the incident faces of e in $\Gamma \circ \mu$ the *μ-incident faces*. The *space around μ in Γ* is $\min\{\ell(\text{apex}(f_1)), \ell(\text{apex}(f_2))\}$; see Figure 5. For the time being we will consider the embeddings of P-node skeletons as fixed. Then all the remaining embedding choices are done by choosing whether or not to flip the embedding for the incoming arc of each R-node. Let A denote the set of arcs in $\mathcal{T}$. For each arc $a = (\lambda, \mu) \in A$ let $\text{space}(\mu)$ denote the space around μ in Γ. We label a as *rigid* if $d(\mu) \geq \text{space}(\mu)$ and as *flexible* otherwise.

Let $\mathcal{T}'$ be the decomposition tree obtained by contracting all rigid arcs and equipping each R-node skeleton with the reference embedding obtained from the contractions. We now release the fixed embedding of the P-nodes, allowing to permute their children arbitrarily. The resulting decomposition tree is called the *LP-tree* of the input graph G. See Figure 6 (d) for an example. Our main result is the following theorem.

▶ **Theorem 6** ($\star$). *Let G be a biconnected, single-source, level-planar graph. The LP-tree of G represents exactly the level-planar embeddings of G and can be computed in linear time.*

The next subsection is dedicated to proving the correctness of Theorem 6. The above algorithm considers every arc of $\mathcal{T}$ once. The height of μ and the space around μ in Γ can be computed in polynomial time. Thus, the algorithm has overall polynomial running time. In the full version, we present a linear-time implementation of this algorithm.

3.3 Correctness

Process the arcs in top-down order $\alpha_1, \ldots, \alpha_m$. For $i = 0, \ldots, m$ let $A_i = \{\alpha_1, \ldots, \alpha_i\}$ contain the first i processed arcs for $i = 0, \ldots, m$. Note that $A_0 = \emptyset$ and $A_m = A$. Denote by R_i and F_i the arcs in A_i that are labeled rigid and flexible, respectively. We now introduce a refinement of the embeddings represented by a decomposition tree. Namely, a *restricted decomposition tree* $\mathcal{T}$ is a decomposition tree together with a subset of its arcs that are labeled as flexible, and, in the arc-based view, the embeddings represented by $\mathcal{T}$ are only those that can be created by flipping only at flexible arcs. We denote by $\mathcal{T}_i$ the restricted decomposition tree obtained from $\mathcal{T}$ by marking only the edges in F_i as flexible.

Initially, $F_0 = \emptyset$, and therefore $\mathcal{T}$ represents exactly the reference embedding Γ_{ref} and its reflection. Since all children of P-nodes have I shape and each P-node has I shape, no arc incident to a P-node is labeled *rigid*. Therefore, if such an edge is contained in A_i, it is flexible. In particular, only arcs between adjacent R-nodes are labeled rigid. As we proceed and label more edges as *flexible*, more and more embeddings are represented. Each time, we justify the level planarity of these embeddings. As a first step, we extend the definition of space from

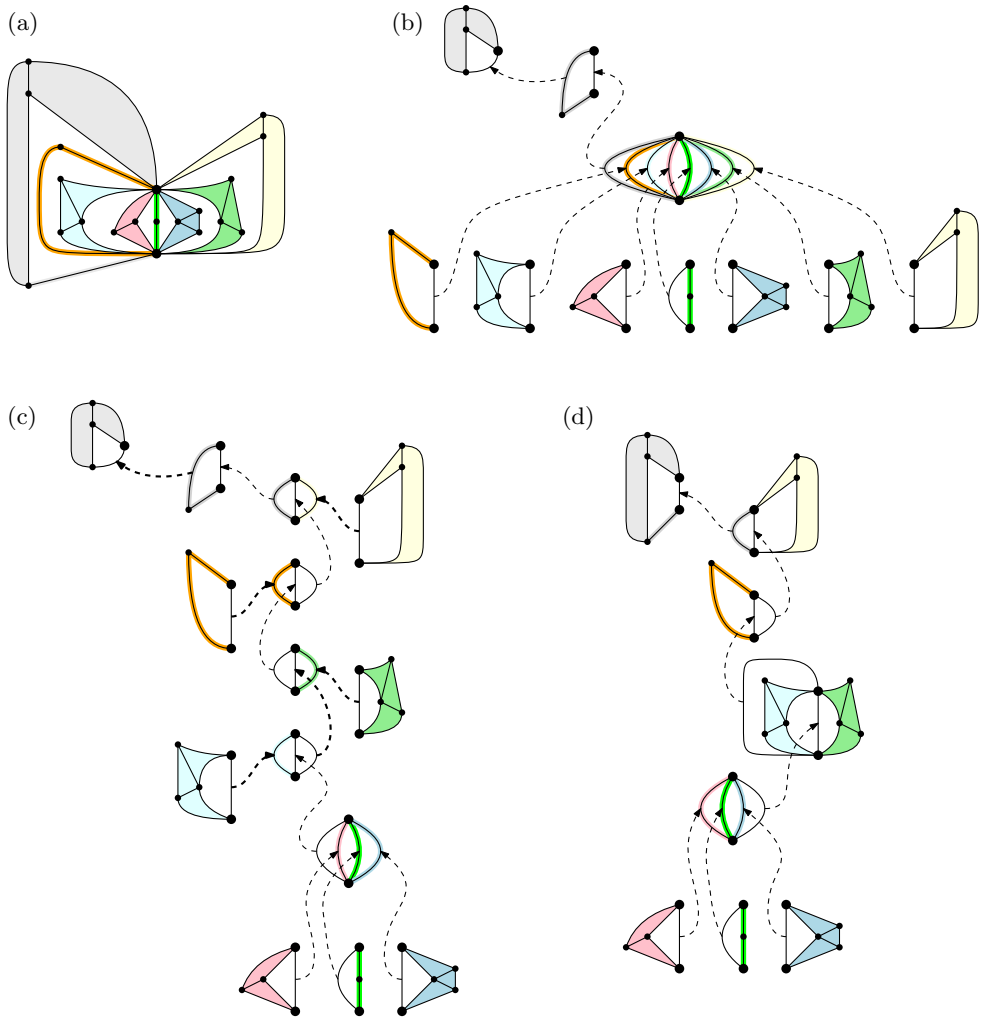

Figure 6 Example construction of the LP-tree for the graph G (a). We start with the SPQR-tree of G (b). Arcs are oriented towards the root. Next, we split the P-node, obtaining the tree shown in (c). Finally, we contract arcs that connect R-nodes with S-nodes and arcs that are found to be rigid (thick dashed lines). This gives the final LP-tree $\mathcal{T}$ for G (d).

the previous subsection, which strongly depends on the initial level-planar embedding Γ, in terms of all level-planar embeddings represented by the restricted decomposition tree $\mathcal{T}_i$. Let μ be a node of $\mathcal{T}_i$ with poles u, v. The *space around* μ is the minimum space around μ in any level-planar embedding represented by the restricted decomposition tree $\mathcal{T}_i$. Now let Γ be a planar embedding of G and let Π be a planar embedding of $G(\mu)$ where u and v lie on the outer face. Because u and v is a separation pair that disconnects $G(\mu)$ from the rest of G and $G(\mu)$ is connected, the embedding of $G(\mu)$ in Γ can be replaced by Π. Let $\Gamma + \Pi$ refer to the resulting embedding. Now let Γ be a planar embedding of G and let μ be a node of $\mathcal{T}$. Let Π denote the restriction of Γ to $G(\mu)$ and let $\bar{\Pi}$ be the reflection of Π. *Reflecting* μ in T corresponds to replacing Π by $\bar{\Pi}$ in Γ, obtaining the embedding $\Gamma + \bar{\Pi}$ of G.

The idea is to show that if there is (is not) enough space around a node μ to reflect it, it can (cannot) be reflected regardless of which level-planar embedding is chosen for $G(\mu)$. So, the algorithm always labels arcs correctly. We use the following invariant.

▶ **Lemma 7** (⋆). *The restricted decomposition tree $\mathcal{T}_i$ satisfies the following five conditions.*

1. *All embeddings represented by $\mathcal{T}_i$ are level planar.*
2. *Let (λ, μ) be an arc that is labeled as flexible. Let Γ be an embedding represented by $\mathcal{T}_{i-1}$ and let Π be any level-planar embedding of $G(\mu)$. Then $\Gamma + \Pi$ and $\Gamma + \bar{\Pi}$ are level planar.*
3. *Let (λ, μ) be an arc that is labeled as rigid. Let Γ be an embedding represented by $\mathcal{T}_{i-1}$ and let Π be a level-planar embedding of $G(\mu)$ so that $\Gamma + \Pi$ is level planar. Let all skeletons of $\mathcal{T}_i$ be embedded according to $\Gamma + \Pi$. Then $\mathrm{skel}(\mu)$ has the reference embedding and $\Gamma + \bar{\Pi}$ is not level planar.*
4. *The space around each node μ of $\mathcal{T}_i$ is the same across all embeddings represented by $\mathcal{T}_i$.*
5. *Let Γ be a level-planar embedding of G so that there exists a level-planar embedding Γ_p of G that (i) is obtained from Γ by reordering the children of P-nodes, and (ii) satisfies $\Gamma_p = \Gamma_{\mathrm{ref}}(\pi_1, \pi_2, \ldots, \pi_m)$ where π_j indicates whether arc $\alpha_j = (\lambda_j, \mu_j)$ should be flipped ($\pi_j = \bar{\alpha}_j$) or not ($\pi_j = \alpha_j$), and it is $\pi_j = \alpha_j$ for $j > i$. Then Γ is represented by $\mathcal{T}_i$.*

The restricted decomposition tree $\mathcal{T}_m$ represents only level-planar embeddings by Property 1 of Lemma 7. Because no arc of $\mathcal{T}_m$ is unlabeled, it also follows that all level-planar embeddings of G are represented by $\mathcal{T}_m$. Contracting all arcs labeled as rigid in $\mathcal{T}_m$ gives the LP-tree for G, which concludes our proof of Theorem 6.

4 Applications

We use the LP-tree to translate efficient algorithms for constrained planarity problems to the level-planar setting. First, we extend the partial planarity algorithm by Angelini et al. [4] to solve partial level planarity for biconnected single-source level graphs. Second, we adapt this algorithm to solve constrained level planarity. In both cases we obtain a linear-time algorithm, improving upon the best previously known running time of $O(n^2)$, though that algorithm also works in the non-biconnected case [12]. Third, we translate the simultaneous planarity algorithm due to Angelini et al. [5] to the simultaneous level planarity problem when the shared graph is a biconnected single-source level graph. Previously, no polynomial-time algorithm was known for this problem.

Partial Level Planarity. Angelini et al. define partial planarity in terms of the cyclic orders of edges around vertices (the "edge-order definition") as follows. A partially embedded graph (PEG) is a triple $(G, H, \mathcal{H})$ that consists of a graph G and a subgraph H of G together with a planar embedding $\mathcal{H}$ of H. The task is to find an embedding $\mathcal{G}$ of G that extends $\mathcal{H}$ in the sense that any three edges e, f, g of H that are incident to a shared vertex v appear in the same order around v in $\mathcal{G}$ as in $\mathcal{H}$. The algorithm works by representing all planar embeddings of G as an SPQR-tree $\mathcal{T}$ and then determining whether there exists a planar embedding of G that extends the given partial embedding $\mathcal{H}$ as follows. Recall that e, f, g correspond to distinct Q-nodes μ_e, μ_f and μ_g in $\mathcal{T}$. There is exactly one node ν of $\mathcal{T}$ that lies on all paths connecting two of these Q-nodes. Furthermore, e, f, g belong to the expansion graphs of three distinct virtual edges $\hat{e}, \hat{f}, \hat{g}$ of $\mathrm{skel}(\nu)$. The order of e, f and g in the planar embedding represented by $\mathcal{T}$ is determined by the order of $\hat{e}, \hat{f}, \hat{g}$ in $\mathrm{skel}(\nu)$, i.e., by the embedding of $\mathrm{skel}(\nu)$. Fixing the relative order of e, f, g therefore imposes certain constraints on the embedding of $\mathrm{skel}(\mu)$. Namely, an R-node can be constrained to have exactly one of its two possible embeddings and the admissible permutations of the neighbors of a P-node can be constrained as a partial ordering. To model the embedding $\mathcal{H}$ consider for each

vertex v of H each triple e, f, g of consecutive edges around v and fix their order as in $\mathcal{H}$. The algorithm collects these linearly many constraints and then checks whether they can be satisfied simultaneously.

Define partial level planarity analogously, i.e., a *partially embedded level graph* is a triple $(G, H, \mathcal{H})$ of a level graph G, a subgraph H of G and a level-planar embedding $\mathcal{H}$ of H. Again the task is to find an embedding $\mathcal{G}$ of G that extends $\mathcal{H}$ in the sense that any three edges e, f, g of H that are incident to a shared vertex v appear in the same order around v in $\mathcal{G}$ as in $\mathcal{H}$. This definition of partial level planarity is distinct from but (due to Lemma 1 ($\star$)) equivalent to the one given in [12], which is a special case of constrained level planarity as presented in the next section. LP-trees exhibit all relevant properties of SPQR-trees used by the partial planarity algorithm. Ordered edges e, f, g of G again correspond to distinct Q-nodes of the LP-tree $\mathcal{T}'$ for G. Again, there is a unique node ν of $\mathcal{T}'$ that has three virtual edges $\hat{e}, \hat{f}, \hat{g}$ that determine the order of e, f, g in the level-planar drawing represented by $\mathcal{T}'$. Finally, in LP-trees just like in SPQR-trees, R-nodes have exactly two possible embeddings and the virtual edges of P-nodes can be arbitrarily permuted. Using the LP-tree as a drop-in replacement for the SPQR-tree in the partial planarity algorithm due to Angelini et al. gives the following, improving upon the previously known best algorithm with $O(n^2)$ running time.

▶ **Theorem 8.** *Partial level planarity can be solved in linear running time for biconnected single-source level graphs.*

Angelini et al. extend their algorithm to the connected case [4]. This requires significant additional effort and the use of another data structure, called the enriched block-cut tree, that manages the biconnected components of a graph in a tree. Some of the techniques described in this paper, in particular our notion of demands, may be helpful in extending our algorithm to the connected single-source case. Consider a connected single-source graph G. All biconnected components of G have a single source and the LP-tree can be used to represent their level-planar embeddings. However, a vertex v of some biconnected component H of G may be a cutvertex in G and can dominate vertices that do not belong to H. Depending on the space around v and the levels on which these vertices lie this may restrict the admissible level-planar embeddings of H. Let $X(v)$ denote the set of vertices dominated by v that do not belong to H. Set the demand of v to $d(v) = d(X(v))$. Computing the LP-tree with these demands ensures that there is enough space around each cutvertex v to embed all components connected at v. The remaining choices are into which faces of H incident to v such components can be embedded and possibly nesting biconnected components. These choices are largely independent for different components and only depend on the available space in each incident face. This information is known from the LP-tree computation. In this way it may be possible to extend the steps for handling non-biconnected graphs due to Angelini et al. to the level planar setting.

Constrained Level Planarity. A *constrained level graph* (CLG) $(G, \{\prec'_1, \prec'_2, \ldots, \prec'_k\})$ consists of a k-level graph G and partial orders $\prec'_i$ of V_i for $i = 1, 2, \ldots, k$ (the "vertex-order definition") [12]. The task is to find a drawing of G, i.e., total orders $\prec_i$ of V_i that extend $\prec'_i$ in the sense that for any two vertices $u, v \in V_i$ with $u \prec'_i v$ it is $u \prec_i v$.

▶ **Theorem 9** ($\star$). *Constrained level planarity can be solved in linear running time for biconnected single-source level graphs.*

Proof Sketch. Consider a depth-first-search tree $\mathcal{D}$ of G. Translate each vertex-order constraint $u \prec'_i v$ to an edge-order constraint around the lowest common ancestor of u and v in $\mathcal{D}$ and use a similar approach as for partial level planarity. ◀

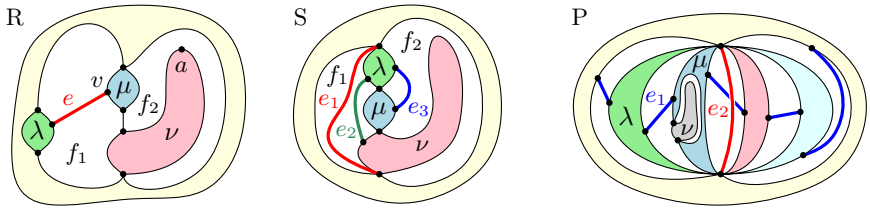

Figure 7 In the R-node, e fixes the relative embeddings of $G(\lambda)$ and $G(\mu)$. In the level-planar setting, e also fixes the embedding of $G(\nu)$. In the S-node, e_2 and e_3 fix the relative embeddings of $G(\lambda), G(\nu)$ and $G(\lambda), G(\mu)$, respectively. In the level-planar setting, e_1 also fixes the embedding of $G(\nu)$. In the P-node, e_1 fixes the relative embeddings of $G(\lambda)$ and $G(\mu)$. In the level-planar setting, e_1 also fixes the embedding of $G(\nu)$.

Simultaneous Level Planarity. We translate the simultaneous planarity algorithm of Angelini et al. [5] to solve simultaneous level planarity for biconnected single-source graphs. Let $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ be two graphs with the same vertices. The *inclusive* edges $E_1 \cap E_2$ together with V make up the intersection graph $G_{1\cap 2}$, or simply G for short. All other edges are *exclusive*. The graphs G_1 and G_2 admit *simultaneous* embeddings $\mathcal{E}_1, \mathcal{E}_2$ if the relative order of any three distinct inclusive edges e, f and g with a shared endpoint is identical in $\mathcal{E}_1$ and $\mathcal{E}_2$. The algorithm of Angelini et al. works by building the SPQR-tree for the shared graph G and then expressing the constraints imposed on G by the exclusive edges as a 2-Sat instance S that is satisfiable iff G_1 and G_2 admit a simultaneous embedding. We give a very brief overview of the 2-Sat constraints in the planar setting. In an R-node, an exclusive edge e has to be embedded into a unique face. This potentially restricts the embedding of the expansion graphs $G(\lambda), G(\mu)$ that contain the endpoints of e, i.e., the embedding of $G(\lambda)$ and $G(\mu)$ is fixed with respect to the embedding of the R-node. Add a variable x_μ to S for every node of $\mathcal{T}$ with the semantics that x_μ is true if skel(μ) has its reference embedding Γ_μ, and false if the embedding of skel(μ) is the reflection of Γ_μ. The restriction imposed by e on $G(\lambda)$ and $G(\mu)$ can then be modeled as a 2-Sat constraint on the variables x_λ and x_μ. For example, in the R-node shown in Figure 7 on the left, the internal edge e must be embedded into face f_1, which fixes the relative embeddings of $G(\lambda)$ and $G(\mu)$. In an S-node, an exclusive edge e may be embedded into one of the two candidate faces f_1, f_2 around the node. The edge e can conflict with another exclusive edge e' of the S-node, meaning that e and e' cannot be embedded in the same face. This is modeled by introducing for every exclusive edge e and candidate face f the variable x_e^f with the semantics that x_e^f is true iff e is embedded into f. The previously mentioned conflict can then be resolved by adding the constraints $x_e^{f_1} \vee x_e^{f_2}$, $x_{e'}^{f_1} \vee x_{e'}^{f_2}$ and $x_e^{f_1} \neq x_{e'}^{f_1}$ to S. Additionally, an exclusive edge e whose endpoints lie in different expansion graphs can restrict their respective embeddings. For example, in the S-node shown in Figure 7 in the middle, the edges e_2 and e_3 may not be embedded into the same face. And e_2 and e_3 fix the embeddings of $G(\lambda)$ and $G(\nu)$ and of $G(\lambda)$ and $G(\mu)$, respectively. This would be modeled as $x_\lambda = x_\nu$ and $x_\lambda = x_\mu$ in S. In a P-node, an exclusive edge can restrict the embeddings of expansion graphs just like in R-nodes. Additionally, exclusive edges between the poles of a P-node can always be embedded unless all virtual edges are forced to be adjacent by internal edges. For example, in the P-node shown in Figure 7 on the right, e_1 fixes the relative embeddings of $G(\lambda)$ and $G(\mu)$. And e_2 can be embedded iff one of the blue edges does not exist.

Adapt the algorithm to the level-planar setting. First, replace the SPQR-tree with the LP-tree $\mathcal{T}$. The satisfying truth assignments of S then correspond to simultaneous planar embeddings $\mathcal{E}_1, \mathcal{E}_2$ of G_1, G_2, so that their shared embedding $\mathcal{E}$ of G is level planar. However,

due to the presence of exclusive edges, $\mathcal{E}_1$ and $\mathcal{E}_2$ are not necessarily level planar. To make sure that $\mathcal{E}_1$ and $\mathcal{E}_2$ are level planar, we add more constraints to S. Consider adding an exclusive edge e into a face f. This splits f into two faces f', f''. The apex of at least one face, say f'', remains unchanged. As a consequence, the space around any virtual edge incident to f'' remains unchanged as well. But the apex of f' can change, namely, the apex of f' is an endpoint of e. Then the space around the virtual edges incident to f' can decrease. This reduces the space around the virtual edge associated with ν. In the same way as described in Section 3.2, this restricts some arcs in $\mathcal{T}$. This can be described as an implication on the variables x_e^f and x_ν. For an example, see Figure 7. In the R-node, adding the edge e with endpoint v into f_1 creates a new face f_1' with apex v. This forces $G(\nu)$ to be embedded so that its apex a is embedded into face f_2. Similarly, in the S-node and in the P-node, adding the edge e_1 restricts $G(\nu)$. We collect all these additional implications of embedding e into f and add them to the 2-Sat instance S. Each exclusive edge leads to a constant number of 2-Sat implications. To find each such implication $O(n)$ time is needed in the worst case. Because there are at most $O(n)$ exclusive edges this gives quadratic running time overall. Clearly, all implications must be satisfied for $\mathcal{E}_1$ and $\mathcal{E}_2$ to be level planar. On the other hand, suppose that one of $\mathcal{E}_1$ or $\mathcal{E}_2$, say $\mathcal{E}_1$, is not level planar. Because the restriction of $\mathcal{E}_1$ to G is level planar due to the LP-tree and planar due to the algorithm by Angelini et al., there must be a crossing involving an exclusive edge e of G_1. This contradicts the fact that we have respected all necessary implications of embedding e. We obtain Theorem 10.

▶ **Theorem 10.** *Simultaneous level planarity can be solved in quadratic time for two graphs whose intersection is a biconnected single-source level graph.*

5 Conclusion

The majority of constrained embedding algorithms for planar graphs rely on two features of the SPQR-tree: they are decomposition trees and the embedding choices consist of arbitrarily permuting parallel edges between two poles or choosing the flip of of a skeleton whose embedding is unique up to reflection. We have developed the LP-tree, an SPQR-tree-like embedding representation that has both of these features. SPQR-tree-based algorithms can then usually be executed on LP-trees without any modification. The necessity for mostly minor modifications only stems from the fact that in many cases the level-planar version of a problem imposes additional restrictions on the embedding compared to the original planar version. Our LP-tree thus allows to leverage a large body of literature on constrained embedding problems and to transfer it to the level-planar setting. In particular, we have used it to obtain linear-time algorithms for partial and constrained level planarity in the biconnected case, which improves upon the previous best known running time of $O(n^2)$. Moreover, we have presented an efficient algorithm for the simultaneous level planarity problem. Previously, no polynomial-time algorithm was known for this problem. Finally, we have argued that an SPQR-tree-like embedding representation for level-planar graphs with multiple sources does not substantially help in solving the partial and constrained level planarity problems, is not efficiently computable, or does not exist.

References

1 Patrizio Angelini and Michael A. Bekos. Hierarchical partial planarity. *Algorithmica*, 81(6):2196–2221, June 2019. doi:10.1007/s00453-018-0530-6.

2 Patrizio Angelini, Thomas Bläsius, and Ignaz Rutter. Testing mutual duality of planar graphs. *International Journal of Computational Geometry & Applications*, 24(4):325–346, 2014. doi:10.1142/S0218195914600103.

3 Patrizio Angelini, Steven Chaplick, Sabine Cornelsen, Giordano Da Lozzo, Giuseppe Di Battista, Peter Eades, Philipp Kindermann, Jan Kratochvíl, Fabian Lipp, and Ignaz Rutter. Simultaneous orthogonal planarity. In Yifan Hu and Martin Nöllenburg, editors, *Graph Drawing and Network Visualization*, pages 532–545. Springer, 2016. doi:10.1007/978-3-319-50106-2_41.

4 Patrizio Angelini, Giuseppe Di Battista, Fabrizio Frati, Vít Jelínek, Jan Kratochvíl, Maurizio Patrignani, and Ignaz Rutter. Testing planarity of partially embedded graphs. *ACM Transactions on Algorithms*, 11(4):32:1–32:42, 2015. doi:10.1145/2629341.

5 Patrizio Angelini, Giuseppe Di Battista, Fabrizio Frati, Maurizio Patrignani, and Ignaz Rutter. Testing the simultaneous embeddability of two graphs whose intersection is a biconnected or a connected graph. *Journal of Discrete Algorithms*, 14:150–172, 2012. doi:10.1016/j.jda.2011.12.015.

6 Patrizio Angelini, Giuseppe Di Battista, and Maurizio Patrignani. Finding a minimum-depth embedding of a planar graph in $O(n^4)$ time. *Algorithmica*, 60(4):890–937, August 2011. doi:10.1007/s00453-009-9380-6.

7 Daniel Bienstock and Clyde L. Monma. On the complexity of embedding planar graphs to minimize certain distance measures. *Algorithmica*, 5(1):93–109, June 1990. doi:10.1007/BF01840379.

8 Thomas Bläsius, Stephen G. Kobourov, and Ignaz Rutter. Simultaneous embedding of planar graphs. In *Handbook on Graph Drawing and Visualization*, pages 349–381. Chapman and Hall/CRC, 2013.

9 Thomas Bläsius, Ignaz Rutter, and Dorothea Wagner. Optimal orthogonal graph drawing with convex bend costs. *ACM Transactions on Algorithms*, 12(3), June 2016. doi:10.1145/2838736.

10 Guido Brückner, , and Ignaz Rutter. An SPQR-tree-like embedding representation for level planarity, 2020. arXiv:2009.12309.

11 Guido Brückner, Markus Himmel, and Ignaz Rutter. An SPQR-tree-like embedding representation for upward planarity. In Daniel Archambault and Csaba D. Tóth, editors, *Graph Drawing and Network Visualization*, pages 517–531. Springer, 2019. doi:10.1007/978-3-030-35802-0_39.

12 Guido Brückner and Ignaz Rutter. Partial and constrained level planarity. In Philip N. Klein, editor, *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2000–2011. SIAM, 2017. doi:10.1137/1.9781611974782.130.

13 Markus Chimani, Carsten Gutwenger, Petra Mutzel, and Christian Wolf. Inserting a vertex into a planar graph. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 375–383. SIAM, 2009.

14 Markus Chimani and Petr Hlinený. Inserting multiple edges into a planar graph. In Sándor Fekete and Anna Lubiw, editors, *32nd International Symposium on Computational Geometry*, volume 51, pages 30:1–30:15, 2016. doi:10.4230/LIPIcs.SoCG.2016.30.

15 Giuseppe Di Battista and Enrico Nardelli. Hierarchies and planarity theory. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(6):1035–1046, 1988. doi:10.1109/21.23105.

16 Giuseppe Di Battista and Roberto Tamassia. Incremental planarity testing. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 436–441, October 1989. doi:10.1109/SFCS.1989.63515.

17 Giuseppe Di Battista and Roberto Tamassia. On-line graph algorithms with SPQR-trees. In Michael S. Paterson, editor, *Proceedings of the 17th International Colloquium on Automata, Languages and Programming*, pages 598–611. Springer Berlin Heidelberg, 1990. doi:10.1007/BFb0032061.

18 Giuseppe Di Battista and Roberto Tamassia. On-line maintenance of triconnected components with SPQR-trees. *Algorithmica*, 15(4):302–318, 1996. doi:10.1007/BF01961541.

19 Radoslav Fulek, Michael J. Pelsmajer, Marcus Schaefer, and Daniel Stefankovic. Hanani-Tutte and monotone drawings. In Petr Kolman and Jan Kratochvíl, editors, *Proceedings of the 37th International Workshop on Graph-Theoretic Concepts in Computer Science*, volume 6986 of *Lecture Notes in Computer Science*, pages 283–294. Springer, 2011. doi:10.1007/978-3-642-25870-1_26.

20 Carsten Gutwenger, Petra Mutzel, and René Weiskircher. Inserting an edge into a planar graph. *Algorithmica*, 41(4):289–308, 2005. doi:10.1007/s00453-004-1128-8.

21 Seok-Hee Hong, Brendan McKay, and Peter Eades. A linear time algorithm for constructing maximally symmetric straight line drawings of triconnected planar graphs. *Discrete & Computational Geometry*, 36(2):283–311, September 2006. doi:10.1007/s00454-006-1231-5.

22 John Edward Hopcroft and Robert Endre Tarjan. Dividing a graph into triconnected components. *SIAM Journal on Computing*, 2(3):135–158, 1973. doi:10.1137/0202012.

23 Michael D. Hutton and Anna Lubiw. Upward planar drawing of single-source acyclic digraphs. *SIAM Journal on Computing*, 25(2):291—311, February 1996. doi:10.1137/S0097539792235906.

24 Michael Jünger and Sebastian Leipert. Level planar embedding in linear time. *Journal of Graph Algorithms and Applications*, 6(1):67–113, 2002. doi:10.7155/jgaa.00045.

25 Michael Jünger, Sebastian Leipert, and Petra Mutzel. Level planarity testing in linear time. In Sue H. Whitesides, editor, *Graph Drawing*, pages 224–237. Springer, 1998.

26 Saunders Mac Lane. A structural characterization of planar combinatorial graphs. *Duke Mathematical Journal*, 3(3):460–472, 1937. doi:10.1215/S0012-7094-37-00336-3.

27 Bert Randerath, Ewald Speckenmeyer, Endre Boros, Peter Hammer, Alex Kogan, Kazuhisa Makino, Bruno Simeone, and Ondrej Cepek. A satisfiability formulation of problems on level graphs. *Electronic Notes in Discrete Mathematics*, 9:269–277, 2001.

28 Roberto Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM Journal on Computing*, 16(3):421–444, 1987. doi:10.1137/0216030.

29 William Thomas Tutte. *Connectivity in Graphs*. University of Toronto Press, 1966.