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


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 (⋆) 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
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 $\Gamma$ of $G$ is level planar if there exists a level-planar drawing of $G$ with planar embedding $\Gamma$. We then call $\Gamma$ a level-planar embedding. For single-source level graphs, level-planar embeddings are equivalence classes of topologically equivalent level-planar drawings.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure1.png}
\caption{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.}
\end{figure}

\begin{lemma}[$\ast$]
The level-planar drawings of a single-source level graph correspond bijectively to its level-planar combinatorial embeddings with $s$ on the outer face.
\end{lemma}

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.

\begin{lemma}[$\ast$]
An embedding $\Gamma$ of $G$ is level-planar if and only if there exists a level-planar embedding $\Gamma'$ of $G'$ that extends $\Gamma$ where $(s,t)$ is incident to the outer face.
\end{lemma}

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. \cite{Angelini2015}. 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 $T$ that consists of two nodes $\mu_1$ and $\mu_2$ connected by an undirected arc as follows. For $i = 1, 2$ node $\mu_i$ is equipped with a multigraph $\skel(\mu_i) = H_i + e_i$, called its skeleton, where $e_i = (u,v)$ is called a virtual edge. The arc $(\mu_1, \mu_2)$ links the two virtual edges $e_i$ in $\skel(\mu_i)$ with each other. We also say that the virtual edge $e_1$ corresponds
to $\mu_2$ and likewise that $e_2$ corresponds to $\mu_1$. The idea is that $\text{skel}(\mu_1)$ provides a more abstract view of $G$ where $e_1$ serves as a placeholder for $H_2$. More generally, there is a bijection $\text{corr}_\mu: E(\text{skel}(\mu)) \to N(\mu)$ that maps every virtual edge of $\text{skel}(\mu)$ to a neighbor of $\mu$ in $T$, and vice versa. If it is $\text{corr}_\mu(u, v) = \nu$, then $\nu$ is said to have poles $u$ and $v$ in $\mu$. If $\mu$ is clear from the context we simply say that $\nu$ 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 $(\nu, \mu)$ of $T$, the virtual edges $e_1, e_2$ with $\text{corr}_\nu(e_1) = \nu$ 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 $T$, 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 $(\mu_1, \mu_2)$ of $T$, forming a new node $\mu$ with a larger skeleton as follows. Let $e_1, e_2$ be twin edges in $\text{skel}(\mu_1), \text{skel}(\mu_2)$. The skeleton of $\mu$ is the union of $\text{skel}(\mu_1)$ and $\text{skel}(\mu_2)$ 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 $\mu$ be a node of a decomposition tree with a virtual edge $e$ with $\text{corr}_\mu(e) = \nu$. The expansion graph of $e$ and $\nu$ in $\mu$, denoted by $G(e)$ and $G(\mu, \nu)$, respectively, is the graph obtained by removing the twin of $e$ from $\text{skel}(\nu)$ and contracting all arcs in the subtree that contains $\nu$.

Each skeleton of a decomposition tree of $G$ is a minor of $G$. So if $G$ is planar, each skeleton of a decomposition tree $T$ of $G$ is planar as well. If $(\mu_1, \mu_2)$ is an arc of $T$, and $\text{skel}(\mu_1)$ and $\text{skel}(\mu_2)$ have fixed planar embeddings $\Gamma_1$ and $\Gamma_2$, respectively, then the skeleton of the node $\mu$ obtained from contracting $(\mu_1, \mu_2)$ can be equipped with an embedding $\Gamma$ by merging these embeddings along the twin edges corresponding to $(\mu_1, \mu_2)$; see Figure 2 on the right. This requires at least one of the virtual edges $e_1$ in $\text{skel}(\mu_1)$ with $\text{corr}_{\mu_1}(e_1) = \mu_2$ or $e_2$ in $\text{skel}(\mu_2)$ 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 $T$ of $G$ represents (not necessarily all) planar embeddings of $G$ by choosing a planar embedding of each skeleton and contracting all arcs. Let $e_{\text{ref}}$ be an edge of $G$. Rooting $T$ at the unique node $\mu_{\text{ref}}$ whose skeleton contains the real edge $e_{\text{ref}}$ identifies a unique parent virtual edge in each of the remaining nodes; all other virtual edges are called child virtual edges. The arcs of $T$ become directed from the parent node to the child node. Restricting the embeddings of the skeletons so that the parent virtual edge (the edge $e_{\text{ref}}$ in case of $\mu_{\text{ref}}$) is incident to the outer face, we obtain a representation of (not necessarily all) planar embeddings of $G$ where $e_{\text{ref}}$ is incident to the outer face. Let $\mu$ be a node of $T$ and let $e$ be a child virtual edge in $\text{skel}(\mu)$ with $\text{corr}_\mu(e) = \nu$. Then the expansion graph $G(\mu, \nu)$ is simply referred to as $G(\nu)$.

![Figure 2](image-url) 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 $\text{skel}(\mu)$ or, equivalently, flipping $(\lambda, \mu)$, yields the same decomposition tree with a different embedding of $\text{skel}(\mu)$. Contract $(\lambda, \mu)$ to obtain the embedding on the right.
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 \((\lambda, \mu)\) whose target \( \mu \) 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 \( T \) bottom-up. Consider the contraction of an arc \((\lambda, \mu)\) whose child \( \mu \) used to be an R-node in \( T \). At this point, \( \text{skel}(\mu) \) is equipped with a planar embedding \( \Gamma_\mu \). If the embedding should be flipped, we reflect the embedding \( \Gamma_\mu \) before contracting \((\lambda, \mu)\), otherwise we simply contract \((\lambda, \mu)\).

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 \( \mu \) is equipped with a reference embedding \( \Gamma_\mu \), and the choice of embeddings for such a node is limited to either \( \Gamma_\mu \) or its reflection. Unlike in SPQR-trees, the skeleton of \( \mu \) 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 \( T \) of \( G \). Explicitly label each node of \( T \) 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...
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 \(T\) 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 \(T\). 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 \(\mu\) be a node of a rooted decomposition tree and let \(u\) and \(v\) be the poles of \(\mu\). Define \(V(\mu) = V(G(\mu)) \setminus \{u,v\}\). The height of \(\mu\) and of the child virtual edge \(e\) with \(\text{corr}(e) = \mu\) is \(d(\mu) = d(e) = d(V(\mu))\). If \(\mu\) is a leaf Q-node it is \(V(\mu) = \emptyset\) and we define the height of \(\mu\) as \(\ell(u)\).

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

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

Lemma 3 motivates the following modification of a decomposition tree \(T\). Take a P-node \(\mu\) with poles \(u,v\) that has a child edge whose height is at least \(\ell(v)\). Denote by \(\lambda\) the parent of \(\mu\). Further, let \(e_{\text{max}}\) be a child virtual edge with maximum height and let \(e_{\text{parent}}\) denote the parent edge of \(\text{skel}(\mu)\). Obtain a new decomposition tree \(T'\) by splitting \(\mu\) into two nodes \(\mu_1\) and \(\mu_2\) representing the subgraph \(H_1\) consisting of the edges \(e_{\text{max}}\) and \(e_{\text{parent}}\), and the subgraph \(H_2\) consisting of the remaining child virtual edges, respectively; see Figure 4. Note that the skeleton of \(\mu_1\), 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 \(\text{skel}(\mu)\)
that is obtained from choosing embeddings for \( \text{skel}(\mu_1) \) and \( \text{skel}(\mu_2) \) and contracting the arc \((\mu_1, \mu_2)\), the edge \( e_{\text{max}} \) is the first or last child edge. Conversely, because \( \mu_2 \) is a P-node, all embeddings where \( e_{\text{max}} \) is the first or last child edge are still represented by \( T' \). Apply this decomposition iteratively, creating new R-nodes on the way, until each P-node \( \mu \) with poles \( u \) and \( v \) has only child virtual edges \( e \) that have height at most \( \ell(v) - 1 \). We say that a node \( \nu \) 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.

\[\textbf{Theorem 4.} \] Let \( G \) be a biconnected single-source graph with unique apex \( t \). There exists a decomposition tree \( T \) that represents all level-planar embeddings of \( G \) such that all children of P-nodes in \( T \) 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 \( \Gamma \) of \( G \) and consider a new embedding \( \Gamma' \) that is obtained from \( \Gamma \) by reordering the children of P-nodes, then also \( \Gamma' \) 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.

\[\textbf{Theorem 5.} \] Let \( G \) be a biconnected single-source graph with a unique apex. There exists a decomposition tree \( T \) 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 \( T \) are embedded so that contracting all arcs of \( T \) yields a level-planar embedding, then the children of all P-nodes in \( T \) can be arbitrarily permuted and then contracting all arcs of \( T \) 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 \( T \) 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 \( T \) 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 \( \Gamma \) of \( G \). Recall that our definition of level-planar embeddings involves demands. Computing a level-planar embedding \( \Gamma' \) 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 \( \mu \) with the reference embedding \( \Gamma_{\mu} \) such that contracting all arcs yields the embedding \( \Gamma \). For the remainder of this section we will work with the arc-based embedding representation. As a first step, we contract any arc \((\lambda, \mu)\) of \( T \) where \( \lambda \) is an R-node and \( \mu \) 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 \( T \). This step makes the correctness proof easier. Any remaining arc \((\lambda, \mu)\) of \( T \) is contracted based upon two properties of \( \mu \), namely the height of \( G(\mu) \) and the space around \( \mu \) in the level-planar embedding \( \Gamma \), which we define next. The resulting node is again labeled as an
\section{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 \textit{restricted decomposition tree} \(T\) 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 \(T\) are only those that can be created by flipping only at flexible arcs. We denote by \(T_r\) the restricted decomposition tree obtained from \(T\) by marking only the edges in \(F_i\) as flexible.

Initially, \(F_0 = \emptyset\), and therefore \(T\) represents exactly the reference embedding \(\Gamma_{\text{ref}}\) 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 \textit{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 \textit{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

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure5.png}
\caption{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 \(\lambda\) is \(\ell(a_1)\), the space around \(\mu\) is \(\ell(v)\) and the space around \(\nu\) is \(\ell(a_5)\).}
\end{figure}

\begin{theorem} \textcolor{blue}{(*)}. Let \(G\) be a biconnected, single-source, level-planar graph. The LP-tree \(\Gamma\) of \(G\) represents exactly the level-planar embeddings of \(G\) and can be computed in linear time. \end{theorem}

The next subsection is dedicated to proving the correctness of Theorem 6. The above algorithm considers every arc of \(T\) once. The height of \(\mu\) and the space around \(\mu\) in \(\Gamma\) 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.
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 $T$ for $G$ (d).

The idea is to show that if there is (is not) enough space around a node $\mu$ 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 $T_i$ satisfies the following five conditions.
1. All embeddings represented by $T_i$ are level planar.
2. Let $(\lambda, \mu)$ be an arc that is labeled as flexible. Let $\Gamma$ be an embedding represented by $T_{i-1}$ and let $\Pi$ be any level-planar embedding of $G(\mu)$. Then $\Gamma + \Pi$ and $\Gamma + \bar{\Pi}$ are level planar.
3. Let $(\lambda, \mu)$ be an arc that is labeled as rigid. Let $\Gamma$ be an embedding represented by $T_{i-1}$ and let $\Pi$ be a level-planar embedding of $G(\mu)$ so that $\Gamma + \Pi$ is level planar. Let all skeletons of $T_i$ be embedded according to $\Gamma + \Pi$. Then $\text{skel}(\mu)$ has the reference embedding and $\Gamma + \bar{\Pi}$ is not level planar.
4. The space around each node $\mu$ of $T_i$ is the same across all embeddings represented by $T_i$.
5. Let $\Gamma$ be a level-planar embedding of $G$ so that there exists a level-planar embedding $\Gamma_p$ of $G$ that (i) is obtained from $\Gamma$ by reordering the children of $P$-nodes, and (ii) satisfies $\Gamma_p = \Gamma_{\text{ref}}(\pi_1, \pi_2, \ldots, \pi_m)$ where $\pi_j$ 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 $\Gamma$ is represented by $T_i$.

The restricted decomposition tree $T_m$ represents only level-planar embeddings by Property 1 of Lemma 7. Because no arc of $T_m$ is unlabeled, it also follows that all level-planar embeddings of $G$ are represented by $T_m$. Contracting all arcs labeled as rigid in $T_m$ 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 $\tilde{e}, \tilde{f}, \tilde{g}$ correspond to distinct $Q$-nodes $\mu_e, \mu_f, \mu_g$ in $\mathcal{T}$. There is exactly one node $\nu$ of $\mathcal{T}$ that lies on all paths connecting two of these Q-nodes. Furthermore, $\tilde{e}, \tilde{f}, \tilde{g}$ belong to the expansion graphs of three distinct virtual edges $\hat{e}, \hat{f}, \hat{g}$ of $\text{skel}(\nu)$. The order of $\hat{e}, \hat{f}$ and $\hat{g}$ in the planar embedding represented by $\mathcal{T}$ is determined by the order of $\hat{e}, \hat{f}, \hat{g}$ in $\text{skel}(\nu)$, i.e., by the embedding of $\text{skel}(\nu)$. Fixing the relative order of $\hat{e}, \hat{f}, \hat{g}$ therefore imposes certain constraints on the embedding of $\text{skel}(\mu)$. 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 \( H \). 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 \( G \) 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 \( v \) as in \( H \). This definition of partial level planarity is distinct from but (due to Lemma 1 (⋆)) 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 \( G \) correspond to distinct Q-nodes of the LP-tree \( T' \) for \( G \). Again, there is a unique node \( v \) of \( T' \) 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 \( T' \). 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.

\begin{theorem}
Partial level planarity can be solved in linear running time for biconnected single-source level graphs.
\end{theorem}

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.

\begin{constrained}
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 \( u \prec_i v \).
\end{constrained}

\begin{theorem} (⋆)
Constrained level planarity can be solved in linear running time for biconnected single-source level graphs.
\end{theorem}

\begin{proofsketch}
Consider a depth-first-search tree \( D \) 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 \( D \) and use a similar approach as for partial level planarity.
\end{proofsketch}
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 in 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 $x_\mu$ to $S$ for every node of $T$ with the semantics that $x_\mu$ is true if $\text{skel}(\mu)$ has its reference embedding $\Gamma_\mu$, and false if the embedding of $\text{skel}(\mu)$ is the reflection of $\Gamma_\mu$. The restriction imposed by $e$ on $G(\lambda)$ and $G(\mu)$ can then be modeled as a 2-Sat constraint on the variables $x_\lambda$ and $x_\mu$. 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 $e$ in 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 $x_e^{f_1} \lor x_e^{f_2}$, $x_{e'}^{f_1} \lor x_{e'}^{f_2}$ 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 $T$. 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.
due to the presence of exclusive edges, $E_1$ and $E_2$ are not necessarily level planar. To make sure that $E_1$ and $E_2$ 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 $\nu$. In the same way as described in Section 3.2, this restricts some arcs in $T$. This can be described as an implication on the variables $x_f^e$ and $x_{\nu}$. 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 $E_1$ and $E_2$ to be level planar. On the other hand, suppose that one of $E_1$ or $E_2$, say $E_1$, is not level planar. Because the restriction of $E_1$ 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.

$\blacktriangleright$ **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**


