# Level Planarity Testing in Linear Time*

Michael Jünger[1], Sebastian Leipert[2], and Petra Mutzel[3]

[1] Institut für Informatik, Universität zu Köln, 50969 Köln, Germany, `mjuenger@informatik.uni-koeln.de`
[2] Institut für Informatik, Universität zu Köln, 50969 Köln, Germany, `leipert@informatik.uni-koeln.de`
[3] Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany, `mutzel@mpi-sb.mpg.de`

**Abstract.** In a leveled directed acyclic graph $G = (V, E)$ the vertex set $V$ is partitioned into $k \leq |V|$ levels $V_1, V_2, \ldots, V_k$ such that for each edge $(u, v) \in E$ with $u \in V_i$ and $v \in V_j$ we have $i < j$. The level planarity testing problem is to decide if $G$ can be drawn in the plane such that for each level $V_i$, all $v \in V_i$ are drawn on the line $l_i = \{(x, k-i) \mid x \in \mathbb{R}\}$, the edges are drawn monotone with respect to the vertical direction, and no edges intersect except at their end vertices. If $G$ has a single source, the test can be performed in $\mathrm{O}(|V|)$ time by an algorithm of Di Battista and Nardelli (1988) that uses the $PQ$-tree data structure introduced by Booth and Lueker (1976). $PQ$-trees have also been proposed by Heath and Pemmaraju (1996a,b) to test level planarity of leveled directed acyclic graphs with several sources and sinks. It has been shown in Jünger, Leipert, and Mutzel (1997) that this algorithm is not correct in the sense that it does not state correctly level planarity of every level planar graph. In this paper, we present a correct linear time level planarity testing algorithm that is based on two main new techniques that replace the incorrect crucial parts of the algorithm of Heath and Pemmaraju (1996a,b).

## 1 Introduction

A fundamental issue in Automatic Graph Drawing is to display hierarchical network structures as they appear in software engineering, project management and database design. The network is transformed into a directed acyclic graph that has to be drawn with edges that are strictly monotone with respect to the vertical direction. Most applications imply a partition of the vertices into levels that have to be visualized by placing the vertices belonging to the same level on a horizontal line. These graphs are called leveled graphs. Testing whether such a graph is level planar, i.e. can be drawn without edge crossings, was solved by Di Battista and Nardelli (1988) for leveled graphs with a single source using the $PQ$-tree data structure.

* Supported by DFG-Grant Ju204/7-2, Forschungsschwerpunkt "Effiziente Algorithmen für diskrete Probleme und ihre Anwendungen"

$PQ$-trees have also been proposed by Heath and Pemmaraju (1996a,b) to test level planarity of leveled directed acyclic graphs with several sources and sinks. It has been shown in Jünger, Leipert, and Mutzel (1997) that this algorithm is not correct in the sense that it does not state correctly level planarity of every level planar graph. In this paper, we present a correct linear time level planarity testing algorithm that is based on two main new techniques that replace the incorrect crucial parts of the algorithm of Heath and Pemmaraju (1996a,b).

This paper is organized as follows. In the next section we give a short introduction to the $PQ$-tree data structure and the level planarity test presented by Heath and Pemmaraju (1996a,b) and we summarize the incorrect crucial parts of this algorithm. In the third section we present a correct algorithm and show how to obtain linear running time. In the last section we make some remarks on the construction of a level planar embedding based on our algorithm.

## 2 Preliminaries

Let $G = (V, E)$ be a directed acyclic graph (dag). A leveling of $G$ is a function $lev : V \to \mathbb{Z}$ mapping the nodes of $G$ to integers such that $lev(v) > lev(u)$ for all $(u, v) \in E$. A leveling of $G$ is called proper if $lev(v) = lev(u)+1$ for all $(u, v) \in E$. $G$ is called a *leveled dag* if a leveling has been assigned to it. If $lev(v) = j$, then $v$ is a *level-$j$ vertex*. Let $V_j = lev^{-1}(j)$ denote the set of level-$j$ vertices. Each $V_j$ is a *level* of $G$.

For the rest of this paper, we assume w.l.o.g. that $G$ is a proper leveled dag with $k \in \mathbb{N}$ levels. We will show in the last section, that our algorithm can be adapted to non proper leveled dags without any modification, preserving the linear running time. However, the algorithm is easier to understand for proper hierarchies.

An embedding of $G$ in the plane is called *leveled* if the vertices of every $V_j$, $1 \leq j \leq k$, are placed on a horizontal line $l_j = \{(x, k - j) \mid x \in \mathbb{R}\}$, and every edge $(u, v) \in E$, $u \in V_j$, $v \in V_{j+1}$ is drawn as a straight line segment between the lines $l_j$ and $l_{j+1}$. A leveled embedding of $G$ is called *level planar* if no two edges cross except at common endpoints. A leveled dag is level planar, if it has a level planar embedding. The dag $G$ is obviously level planar, if all its components are level planar. We therefore assume that $G$ is connected.

A leveled embedding of $G$ determines for every $V_j$, $1 \leq j \leq k$, a total order $\leq_j$ of the vertices of $V_j$, given by the left to right order of the nodes on $l_j$. In order to test whether a leveled embedding of $G$ is level planar, it is sufficient to find an ordering of the vertices of every set $V_j$, $1 \leq j < k$, such that for every pair of edges $(u_1, v_1), (u_2, v_2) \in E$ with $lev(u_1) = lev(u_2) = j$ and $u_1 \leq_j u_2$ it follows that $v_1 \leq_{j+1} v_2$. Apparently, the ordering $\leq_j$, $1 \leq j \leq k$, describes a permutation of the vertices of $V_j$. Let $G_j$ denote the subgraph of $G$, induced by $V_1 \cup V_2 \cup \ldots \cup V_j$. Unlike $G$, $G_j$ is not necessarily connected.

The basic idea of the level planarity testing algorithm presented by Heath and Pemmaraju (1996a,b) is to perform a top-down sweep, processing the levels in the order $V_1, V_2, \ldots, V_k$ and computing for every level $V_j$, $1 \leq j \leq k$, a set of permutations of the vertices of $V_j$ that appear in some level planar embedding of $G_j$. In case that the set of permutations of $G_k$ is not empty, the graph $G = G_k$ is obviously level planar.

A $PQ$-tree is a data structure that represents the permutations of a finite set $U$ in which the members of specified subsets occur consecutively. This data structure has been introduced by Booth and Lueker (1976) to solve the problem of testing for the consecutive ones property. A $PQ$-tree contains three types of nodes: leaves, $P$-nodes, and $Q$-nodes. The leaves are in one to one correspondence with the elements of $U$. The $P$- and $Q$-nodes are internal nodes. A $P$-node is allowed to permute its children arbitrarily, while the order of the children of a $Q$-node is fixed and only may be reversed. In subsequent figures, $P$-nodes are drawn as circles while $Q$-nodes are drawn as rectangles.

The set of leaves of a $PQ$-tree $T$ read from left to right is denoted by frontier($T$) and yields a permutation on the elements of the set $U$. The frontier of a node $X$, denoted by frontier($X$), is the sequence of its descendant leaves. Given a $PQ$-tree $T$ over the set $U$ and given a subset $S \subseteq U$, Booth and Lueker (1976) developed a pattern matching algorithm called *reduction* and denoted by REDUCE($T, S$) that computes a $PQ$-tree $T'$ representing all permutations of $T$ in which the elements of $S$ form a consecutive sequence.

If $G_j$ is a hierarchy, the set of permutations of the vertices of $V_j$ that appear in some level planar embedding of $G_j$ can be represented by a $PQ$-tree $T_j$ according to Di Battista and Nardelli (1988) as follows:

1. identify with every vertex of $V_j$ exactly one leaf,
2. identify with every cut vertex in $G_j$ a $P$-node,
3. identify with every maximal biconnected components in $G_j$ a $Q$-node,

In order to test whether the hierarchy $G_{j+1}$ is level planar, Di Battista and Nardelli (1988) add for every edge $(v_i, w)$, $w \in V_{j+1}$, $v_i \in V_j$, $i = 1, 2, \ldots, \mu$, $\mu \geq 1$ a virtual vertex $w_{v_i}$ labeled $w$ and virtual edges $(v_i, w_{v_i})$ to the graph $G_j$. The authors then try to compute for every vertex $w \in V_{j+1}$ a sequence of permutations of components around cutvertices and swappings of maximal biconnected components such that all virtual vertices labeled $w$ form a consecutive sequence on the horizontal line $l_{j+1}$. If such a sequence can be found, it is obvious that the vertex $w$ can be added to $G_j$ without destroying level planarity. The process of computing the prescribed sequence can be efficiently done using the $PQ$-tree $T_j$, yielding a linear time algorithm.

In case that $G_j$, $1 \leq j < k$, consists of more than one connected component, Heath and Pemmaraju suggest to use a $PQ$-tree for every component and formulate a set of rules of how to merge components $F_1$ and $F_2$, respectively their corresponding $PQ$-trees $T_1$ and $T_2$, if $F_1$ and $F_2$ both are adjacent to some vertex $v \in V_{j+1}$.

Heath and Pemmaraju (1996a,b) reduce during a *First Merge Phase* the leaves of $T_1$ and $T_2$ corresponding to the vertex $v$, called the *pertinent* leaves. After successfully performing the reduction, the consecutive sequence of pertinent leaves is replaced by a single pertinent representative in both $T_1$ and $T_2$. Going up one of the trees $T_i$, $i \in \{1, 2\}$, from its pertinent representative, an appropriate position is searched, allowing the tree $T_j$, $j \neq i$, to be placed into $T_i$. After successfully performing this step the resulting tree $T'$ has two pertinent leaves corresponding to the vertex $v$, which again are reduced and replaced by a single representative. If any of the steps fails, Heath and Pemmaraju state that the graph $G$ is not level planar.

Merging two $PQ$-trees $T_1$ and $T_2$ corresponds to merging the two components $F_1$ and $F_2$ and is accomplished using certain information that is stored at the nodes of the $PQ$-trees. For any subset $S$ of the set of vertices in $V_j$, $1 \leq j \leq m$, that belongs to a component $F$, define $\mathrm{ML}(S)$ to be the greatest $d \leq j$ such that $V_d, V_{d+1}, \dots, V_j$ induces a dag in which all nodes of $S$ occur in the same connected component. For a $Q$-node $q$ in the corresponding $PQ$-tree $T_F$ with ordered children $r_1, r_2, \dots, r_t$ integers denoted by $\mathrm{ML}(r_i, r_{i+1})$, $1 \leq i < t$, are maintained satisfying $\mathrm{ML}(r_i, r_{i+1}) = \mathrm{ML}(\mathrm{frontier}(r_i) \cup \mathrm{frontier}(r_{i+1}))$. For a $P$-node $p$ a single integer denoted by $\mathrm{ML}(p)$ that satisfies $\mathrm{ML}(p) = \mathrm{ML}(\mathrm{frontier}(p))$ is maintained. Furthermore, define $\mathrm{LL}(F)$ to be the smallest $d$ such that $F$ contains a vertex in $V_d$ and maintain this integer at the root of the corresponding $PQ$-tree. The *height* of a component $F$ in the subgraph $G_j$ is $j - \mathrm{LL}(F)$. Using these LL- and ML-values, Heath and Pemmaraju (1996a,b) describe a set of rules how to connect two $PQ$-trees claiming that the pertinent leaves of the new tree $T'$ are reducible if and only if the corresponding component $F'$ is level planar.

In Jünger, Leipert, and Mutzel (1997) we have shown that the order of merging the components is important for testing a leveled dag. Moreover, it is easy to see that using different orderings while merging three or more components results in different $PQ$-trees. So even if every order of merging $PQ$-trees with pertinent leaves labeled $v$, $lev(v) = j$, $1 \leq j < k$, results in a reducible $PQ$-tree, a $PQ$-tree may be constructed such that the leaves of some vertex $l$, $lev(l) > j$ are not reducible, although the graph $G$ is level planar. Hence the algorithm presented by Heath and Pemmaraju (1996a,b) may state incorrectly the non level planarity of a level planar graph.

Furthermore, components of $G_j$, that have just one level-$j$ vertex are not treated properly. In fact, they may be inserted at wrong positions in other $PQ$-trees. This is due to the fact that during the first merge phase the algorithm reduces for every $PQ$-tree all leaves with the same label and replaces them by a single representative. Clearly, this replacement corresponds to the construction of new interior faces in the corresponding subgraph. However, $PQ$-trees are not designed to carry information about interior faces, hence the information about the "space" within these interior faces gets lost. It is easy to see that situations may occur where components being adjacent to just one level-$j$ vertex have to be embedded within one of these interior faces. The approach of Heath and Pemmaraju (1996a,b) does not detect this fact, which is another reason that it may incorrectly state the non level planarity of a level planar graph.

Heath and Pemmaraju (1996a,b) claim that their algorithm can be implemented using only $\mathrm{O}(|V|)$ time. This is true for the merge and reduce operations. However, considering two $PQ$-trees $T_1$, $T_2$ both having a leaf labeled $v$ and a leaf labeled $w$, Heath and Pemmaraju (1996b) suggest to merge the trees $T_1$ and $T_2$

at the leaves labeled $v$ constructing a new $PQ$-tree $T$ and then reduce $T$ with respect to the leaves labeled $v$ as well as with respect to the leaves labeled $w$. It is not clear how the update operations that are necessary for detecting both pairs of leaves can be done in $O(|V|)$ time, Heath and Pemmaraju (1996a,b) do not discuss this matter.

We will combine two new strategies to eliminate the problems we encountered in the algorithm of Heath and Pemmaraju (1996a,b).

## 3 A Correct Linear Time Level Planarity Test

In this section we discuss how to construct a correct algorithm LEVEL-PLANAR-TEST that tests a leveled dag $G = (V_1, V_2, \dots, V_k; E)$ for level planarity. Since $G_j$ is not necessarily connected, let $m_j$ denote the number of components of $G_j$ and let $F_i^j$, $i = 1, 2, \dots, m_j$, denote the components of $G_j$. Number the vertices of level $j+1$ arbitrarily from 1 to $|V_{j+1}|$. We refer to the vertices of $V_{j+1}$ by their numbers. Let $H_i^j$ be the component formed by adding to $F_i^j$ all edges with one end in $F_i^j$ and the other end in $V_{j+1}$, keeping the ends in $V_{j+1}$ separate. These edges are called virtual edges and their ends in $V_{j+1}$ are called virtual vertices. The virtual vertices are labeled as their counterparts in $V_{j+1}$, but they are kept separate. Thus there may be several virtual vertices with the same label, adjacent to different components of $G_j$ and each with exactly one entering edge. The component $H_i^j$ is called the *extended form* of $F_i^j$ and the set of virtual vertices of $H_i^j$ is called frontier$(H_i^j)$. Let $B_i^j$ be a level planar embedding of $H_i^j$. Obviously, all virtual vertices of $H_i^j$ are placed on the same horizontal line on the outer face. The set of virtual vertices of $H_i^j$ that are labeled $v \in V_{j+1}$ is denoted by $S_i^v$. Figure 1 shows an example of an extended form $H_i^j$ and its corresponding $PQ$-tree, representing all permutations of the virtual vertices that appear in some level planar embedding of $H_i^j$. The form $H_i^j$ has two virtual vertices labeled $v$.

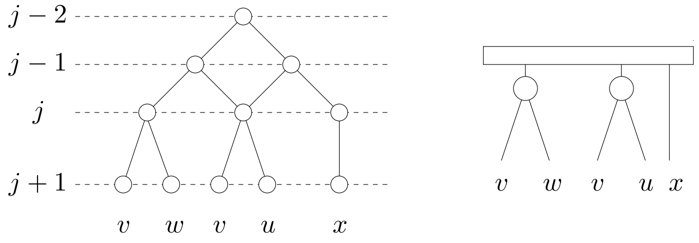

**Fig. 1.** An extended form $H_i^j$ and its $PQ$-tree.

The component that is created from an extended form $H_i^j$ by identifying for various $v \in V_{j+1}$ all virtual vertices with the label $v$ to a single vertex $v_i$ with label $v$ is called *reduced extended form* and denoted by $R_i^j$. The form $R_i^j$ is called *proper* if for all $v \in V_{j+1}$ the virtual vertices with the same label $v$ have been identified, otherwise $R_i^j$ is called *sloppy*. The set of virtual vertices of $R_i^j$ is denoted by frontier$(R_i^j)$. If the virtual vertices labeled $v$ have been identified in $R_i^j$ to a vertex $v_i$, we denote by $S_i^v = \{v_i\}$ the set of vertices with label $v$ of $R_i^j$. Figure 2 shows an example of a reduced extended form $R_i^j$ and its corresponding $PQ$-tree. The form $R_i^j$ has been constructed from the extended form $H_i^j$ shown in Fig. 1 by identifying the two virtual vertices labeled $v$. The corresponding $PQ$-tree has been constructed by reducing the two leaves labeled $v$ applying the pattern matching algorithm of Booth and Lueker (1976).

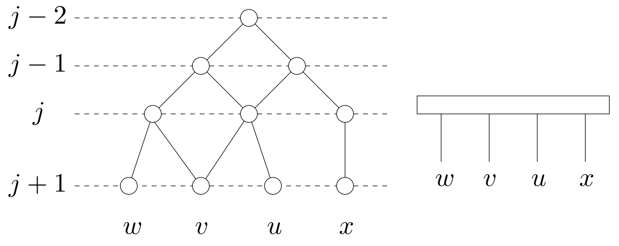

**Fig. 2.** A proper reduced extended form $R_i^j$ and its $PQ$-tree.

Identifying the sets $S_i^v \neq \emptyset$ and $S_l^v \neq \emptyset$ of two reduced extended forms $R_i^j$ and $R_l^j$, $i \neq l$, $i, l \in \{1, 2, \dots, m_j\}$, to a single vertex $v_{\{i,l\}}$ with label $v$ is denoted by $R_i^j \cup_v R_l^j$. We call $R_i^j \cup_v R_l^j$ a *merged reduced component*. If $\mathrm{LL}(R_i^j) \leq \mathrm{LL}(R_l^j)$ we say $R_l^j$ is *v-merged* into $R_i^j$. The component that is created by $v$-merging $R_l^j$ into $R_i^j$ is again a reduced extended component and denoted by $R_i^j$ (thus renaming $R_i^j \cup_v R_l^j$ with the name of the "higher" component). If $R_i^j$, $i \in \{1, 2, \dots, m_j\}$ is a reduced extended component, such that $S_i^v \neq \emptyset$ for some $v \in V_{j+1}$ and $S_i^w = \emptyset$ for all $w \in V_{j+1} - \{v\}$, then $R_i^j$ is called *v-singular*.

A collection $C(G_j)$, $1 \leq j \leq k$, denotes the set of level planar embeddings of all components of $G_j$. One of our results is that in case that $G_j$ is level planar, a $PQ$-tree $T(F_i^j)$ can be associated with every $F_i^j$ of $G_j$ describing the set of level planar embeddings of $F_i^j$. As has been shown in Booth and Lueker (1976), it is straightforward to construct from $T(F_i^j)$ a $PQ$-tree $T(H_i^j)$ associated with $H_i^j$. Thus the leaves of $T(H_i^j)$ correspond to the virtual vertices of $H_i^j$ and we label the leaves of $T(H_i^j)$ as their counterparts in $H_i^j$. By construction, $C(G_j)$ is a set of $PQ$-trees. Considering a function CHECK-LEVEL that computes for every level $j$, $j = 2, 3, \dots, k$ the set $C(G_j)$ of level planar embeddings of the components $G_j$, the algorithm LEVEL-PLANAR-TEST can be formulated as follows.

```
Bool LEVEL-PLANAR-TEST($G = (V_1, V_2, \dots, V_k; E)$)
begin
    Initialize $C(G_1)$;
    for $j := 1$ to $k - 1$ do
        $C(G_{j+1}) =$ CHECK-LEVEL$(C(G_j), V_{j+1})$;
        if $C(G_{j+1}) = \emptyset$ then
            return "$G$ is not level planar.";
    return "$G$ is level planar.";
end.
```

We introduce two new strategies that lead to a correct algorithm as well as new techniques for obtaining linear running time. One strategy is to sort all $PQ$-trees with a leaf labeled $v$ in their frontier according to their LL-values and merge them according to this ordering. We show that the new $PQ$-tree constructed by the application of this ordering represents all possible level planar embeddings of the corresponding new component. Our second strategy for a correct treatment of $v$-singular components consists of keeping at every single representative the size of the largest interior face that has been constructed by identifying the corresponding virtual vertices. When merging a $PQ$-tree of a $v$-singular component into another $PQ$-tree with lower LL-value, this information is checked first. When merging two non singular components, this information has to be updated when introducing a new single representative. Here we have to take in account that merging two components results into something that we call a *cavity*. Considering the intersection $\mathcal{C}$ of the halfspace $\{x \in \mathbb{R}^2 \mid x_2 \geq k-j-1\}$ and the outer face of the current embedding, a $v$-cavity is defined to be a region of $\mathcal{C}$ such that $v$ is adjacent to the region. Obviously $v$ can be adjacent to several such regions. Moreover, these regions are not unique, since they depend on the current embedding. This is no drawback, since we only need to maintain the size of the largest $v$-cavity which can be easily implemented using the $PQ$-trees and the LL- and ML-values of Heath and Pemmaraju (1996a,b). Figure 3 shows such a $v$-cavity. The arrow on the right side of the figure depicts the height of the cavity. A $v$-singular component can only be level planar embedded within this cavity, if it is smaller than the height of the cavity.

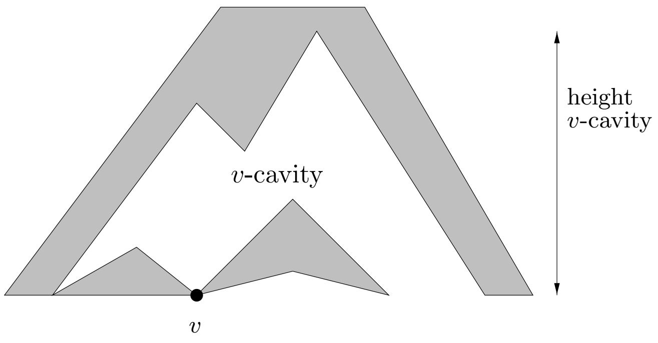

**Fig. 3.** A $v$-cavity.

As we have mentioned in the previous section, merging two $PQ$-trees at leaves labeled $v$ may result in a $PQ$-tree $T$ with several leaves labeled $w \neq v$. Linearity of the algorithm is achieved by not applying the strategy of reducing the leaves labeled $w$, since it is not clear if the detection of these leaves reveals linear running time. We reduce these leaves labeled $w$ only when considering their $PQ$-tree $T$ for a merge operation at $w$. Thus we first merge all leaves labeled $w$ in every tree and then merge these trees at $w$. We show that the modified algorithm works correctly. When merging $PQ$-trees, update operations have to be applied to the leaves of the new tree, since the leaves must know the $PQ$-tree that they belong to. To avoid the usage of *Fast-Union-Find-Set* operations which sum up to $O(|V|\alpha(|V|, |V|))$ operations, we apply the following strategy. Leaves are updated only when they are involved in a reduce or merge operation. In order to update the leaves, we traverse all nodes from the considered leaf to its root. Let $U$ be a set of $PQ$-trees with leaves labeled $w$ in their frontier. We show that if this strategy is applied for all leaves except for the leaves in the $PQ$-tree with the lowest LL-value in $U$, the number of operations is proportional to the number of operations needed to reduce all these leaves. We do not need to know the $PQ$-tree with the lowest LL-value in $U$. It is easy to see that this tree is implicitly defined. Hence we can avoid for every merge operation the traversal of the tree corresponding to the highest component. Thus the total number of operations needed to perform the updates is bounded by $O(|V|)$.

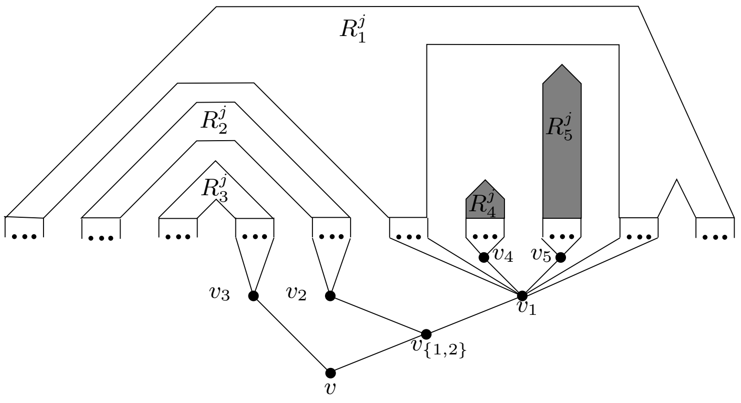

**Fig. 4.** The forms $R_i^j$, $i = 1, 2, \ldots, 5$ are merged at vertex $v$. The $v$-singular components are drawn shaded and placed into an interior face of $R_1^j$. The non singular components are merged according to their height. The vertices $v_i$, $i = 1, 2, \ldots, 5$ and $v_{\{1,2\}}$ each correspond to a new leaf that replaces the sequence of pertinent leaves in the corresponding $PQ$-tree.

The procedure CHECK-LEVEL is divided into two phases. The *First Reduction Phase* constructs the $PQ$-trees corresponding to the reduced extended forms of $G_j$. Every $PQ$-tree $T(F_i^j)$ that represents all level planar embeddings of some component $F_i^j$ is transformed into a $PQ$-tree $T(H_i^j)$ representing all level planar embeddings of the extended form $H_i^j$. We continue to reduce in every $PQ$-tree $T(H_i^j)$ all leaves with the same label, thereby constructing a new $PQ$-tree, representing all level planar embeddings of $H_i^j$, where leaves with the same label occupy consecutive positions. If one of the reductions fails, $G$ cannot be level planar. Leaves with the same label $v$ are replaced by a single representative $v_i$. Such a single representative $v_i$ gets the same label $v$, storing either a value $\text{PML}(v_i) = \text{ML}(R)$ if the root $R$ of the pertinent subtree is a $P$-node or a value $\text{QML}(v_i) = \min\{\text{ML}(x, y) \mid x, y$ consecutive children of $R$, $x$ pertinent or $y$ pertinent$\}$, if the root $R$ is a $Q$-node. The default value of $\text{QML}(v_i)$ and $\text{PML}(v_i)$ is set to $k + 1$. These values store the height of the largest new interior face that is constructed by merging the vertices labeled $v$ and are needed to handle singular components correctly.

$PQ$-trees of different components are merged in the *Second Reduction Phase* using a function INSERT, if the components are adjacent to the same vertex $v$ on level $j+1$. Given the set of leaves labeled $v$, we first determine their corresponding $PQ$-trees. If some leaves labeled $v$ are in the frontier of the same $PQ$-tree, we reduce them and replace them by a single representative. The $PQ$-trees are then merged pairwise in the order of their sizes. We show that using this ordering a $PQ$-tree $T(F)$ is constructed, that represents all possible level planar embeddings of the merged components. If there is more than one $v$-singular reduced extended form, $v \in V_{j+1}$, we only need to merge the largest one of these forms. If it is possible to embed this form level planar, all other $v$-singular forms obviously can be embedded level planar as well. Even though $v$ may not be the only common vertex in the merged components, we do not reduce leaves with label $w \neq v$ in the $PQ$-tree in order to obtain a linear time algorithm. If one of the reduce or merge operations fails while applied in this phase, the graph $G$ is not level planar. Figure 4 illustrates the merge process. The PML- and QML-values are updated by using a function UPDATE. Finally we add for every source of $V_{j+1}$ its corresponding $PQ$-tree. Thus the set of $PQ$-trees constructed by the function CHECK-LEVEL represents all level planar embeddings of the components $G_{j+1}$. The following code fragment contains operations that perform on the graph $G$. They are kept in the code for documentation purposes. Any implementation would of course rely only on the manipulation of the $PQ$-trees.

```
$C(G_{j+1})$ **CHECK-LEVEL**($C(G_j), V_{j+1}$)
begin
    **First Reduction Phase**
    for every component $F_i^j$ in $G_j$ and its corresponding $PQ$-tree in $T(F_i^j)$ do
        construct $H_i^j$; construct $T(H_i^j)$;
    for every $v \in V_{j+1}$ do
        for every extended form $H_i^j$ do
            if $S_i^v \neq \emptyset$ then
                if REDUCE($T(H_i^j), S_i^v$) $= \emptyset$ then return $\emptyset$;
            else
                replace $S_i^v$ in $T(H_i^j)$ by a single representative $v_i$;
                set PML($v_i$) or QML($v_i$); $S_i^v := \{v_i\}$;
        for every extended form $H_i^j$ do $T(R_i^j) := T(H_i^j)$;
    **Second Reduction Phase**
    for $v := 1$ to $|V_{j+1}|$ do
        for every leaf labeled $v$ do find the corresponding $PQ$-tree;
        for every found $PQ$-tree $T_i$, $i \in \{1, 2, \ldots, m_j\}$ do
            if $S_i^v \geq 2$ then
                if REDUCE($T_i$,$S_i^v$) $= \emptyset$ then return $\emptyset$;
                else
                    let $v_T$ be a new single representative of $S_i^v$;
                    UPDATE($S_i^v$,$v_T$); replace $S_i^v$ in $T_i$ by $v_T$; $S_i^v := \{v_T\}$;
        let $R_i^j$, $i := 1, 2, \ldots, p$, be the sloppy reduced extended forms;
        let $o$ be the number of $v$-singular reduced extended forms;
        eliminate all $v$-singular $R_i^j$ except for the one with the lowest LL-value;
        renumber the remaining $R_i^j$ from 1 to $p - o + 1$; $p := p - o + 1$;
        sort the $R_i^j$, such that LL($R_1^j$) $\leq$ LL($R_2^j$) $\leq$ LL($R_3^j$) $\leq \ldots \leq$ LL($R_p^j$);
        $F := R_1^j$; $T(F) := T(R_1^j)$;
        for $i := 1$ to $p - 1$ do
            $T(F) :=$ INSERT($T(F), T(R_{i+1}^j), v$); $F := F \cup_v R_{i+1}^j$;
            if REDUCE($T(F), S_F^v$) $= \emptyset$ then return $\emptyset$;
            else
                let $v_F$ be a new single representative of $S_F^v$;
                UPDATE($S_F^v$,$v_F$); replace $S_F^v$ in $T(F)$ by $v_F$; $S_F^v := \{v_F\}$;
    update the root pointers of the leaves; add all sources of $V_{j+1}$ in $G$ to $H$;
    add for every source a corresponding $PQ$-tree to $C(G_j)$;
    $C(G_{j+1}) := C(G_j)$;
    return $C(G_{j+1})$;
end.
```

We now describe in detail how to merge the *PQ*-trees corresponding to two components. All five rules presented by Heath and Pemmaraju can be adapted, but contrary to their algorithm, we have to deal with the fact that a *PQ*-tree may correspond to a singular component. Merging two *PQ*-trees is handled by the method INSERT. Let LL($T_{large}$) and LL($T_{small}$) be two *PQ*-trees such that $S_{large}^v \neq \emptyset$ and $S_{small}^v \neq \emptyset$, and LL($T_{large}$) $\leq$ LL($T_{small}$). Assume further that $S_{large}^v$ and $S_{small}^v$ have been reduced and replaced by a single representative $v_{large}$ resp. $v_{small}$ and that $S_{large}^v = \{v_{large}\}$ and $S_{small}^v = \{v_{small}\}$. INSERT returns a new *PQ*-tree $T_{merge}$. The method does not reduce the pertinent sequence, nor does it replace pertinent leaves by a single leaf. Observe that in case frontier($T_{small}$) $= S_{small}^v$, we do not really add $T_{small}$ to $T_{large}$, since the component corresponding to $T_{small}$ can be embedded in an interior face or a $v$-cavity of the component corresponding to $T_{large}$.

```
$T_{merge}$ INSERT($T_{large}, T_{small}, v$)
begin
    if frontier($T_{small}$) $\neq S^v_{small}$ then
        attach $T_{small}$ to $T_{large}$ as described in Heath and Pemmaraju (1996a,b);
    else if PML($v_{large}$) $\neq k+1$ then
        if PML($v_{large}$) $<$ LL($T_{small}$) then do nothing;
        else attach $T_{small}$ to $T_{large}$ as described in Heath et al. (1996a,b);
    else if QML($v_{large}$) $\neq k+1$ then
        if QML($v_{large}$) $<$ LL($T_{small}$) then do nothing;
        else attach $T_{small}$ to $T_{large}$ as described in Heath et al. (1996a,b);
    return the new $PQ$-tree $T_{merge}$;
end.
```

The method UPDATE is applied after two $PQ$-trees have been successfully merged and reduced at their leaves labeled $v$. UPDATE computes the PML- and QML-value of the new single representative. The values $\text{PML}_r$ and $\text{QML}_r$ that appear in the code, are used to compute the height of the largest new cavity.

```
UPDATE($S^v_F$,$v_F$)
begin
    $\text{PML}_{\min} := \min\{\text{PML}(\tilde{v}) \mid \tilde{v} \in S^v_F\}$; $\text{QML}_{\min} := \min\{\text{QML}(\tilde{v}) \mid \tilde{v} \in S^v_F\}$;
    let $r$ be the root of the pertinent subtree;
    if $r$ is a $P$-node then $\text{PML}_r := \text{ML}(r)$;
    else if $r$ is a $Q$-node then
        $\text{QML}_r := \min\left\{\text{ML}(x,y) \mid \begin{array}{l} x, y \text{ consecutive children of } r, \\ x \text{ pertinent or } y \text{ pertinent} \end{array}\right\}$;
    if $\min\{\text{PML}_{\min}, \text{PML}_r\} < \min\{\text{QML}_{min}, \text{QML}_r\}$ then
        $\text{PML}(v_F) := \min\{\text{PML}_{\min}, \text{PML}_r\}$; $\text{QML}(v_F) := k+1$;
    else
        $\text{QML}(v_F) := \min\{\text{QML}_{\min}, \text{QML}_r\}$; $\text{PML}(v_F) := k+1$;
end.
```

The following theorem shows the correctness of our level planarity test.

**Theorem 1.** *Let $G = (V, E)$ be a directed level planar graph with $k \geq 2$ levels. The algorithm LEVEL-PLANAR-TEST tests $G$ for level planarity.*

*Proof.* We use an inductive argument. Clearly, every component $F$ of $G$ that is induced by a source and its neighbors is a hierarchy. According to Di Battista and Nardelli (1988) we have a $PQ$-tree for every component $F$ representing all level planar embeddings. So we need to show that in every iteration, the $PQ$-trees are correctly maintained and the set of permissible permutations of a $PQ$-tree always represents the set of level planar embeddings of the corresponding form.

Let $F^j_i$, $i \in \{1, 2, \ldots, m_j\}$, be an arbitrary component of $G_j$, $1 \leq j < k$, $H^j_i$ be its extended form and $R^j_i$ be its proper reduced extended from. It is straightforward to show that if $T_i$ is a $PQ$-tree representing all level-planar embeddings of $H^j_i$, then there exists a $PQ$-tree $T'_i$, equivalent to $T_i$, such that for all $v \in \{1, 2, \dots, |V_{j+1}|\}$, the leaves corresponding to $S_i^v$ occupy consecutive positions. Thus the $PQ$-tree constructed from $T_i'$ by reducing every set $S_i^v$ and replacing it by a single representative $v_i$ represents all level planar embeddings of $R_i^j$ implying the correctness of the first merge phase.

Now let $v$ be an arbitrary vertex of level $j+1$, where $j < k$. Let $R_1^j, R_2^j, \dots, R_p^j$, $p \geq 2$ be reduced extended components with their virtual vertices on level $j+1$ kept separate such that $S_i^v \neq \emptyset$ for all $i = 1, 2, \dots, p$ and $|S_i^w| \leq 1$ for all $w = 1, 2, \dots, v$ and $i = 1, 2, \dots, p$. Let $F$ be the component induced by $R_1^j, R_2^j, \dots, R_p^j$ with $v$ being the only identified vertex. All other vertices with common label are kept separate. Assume w.l.o.g. that $\mathrm{LL}(R_1^j) \leq \mathrm{LL}(R_2^j) \leq \mathrm{LL}(R_3^j) \leq \dots \leq \mathrm{LL}(R_p^j)$. Assume further, that $F$ is constructed by first merging $R_1^j$ and $R_2^j$ to $R_{\{1,2\}}^j$ identifying the sets of virtual vertices $S_1^v$ and $S_2^v$ to one vertex $v$, and then merging for every $i = 3, 4, \dots, p$ the reduced extended forms $R_{\{1,2,\dots,i-1\}}^j$ and $R_i^j$ to $R_{\{1,2,\dots,i\}}^j$ identifying the sets of virtual vertices $S_i^v$ to $v$. Let $T_1, T_2, \dots, T_p$ be the $PQ$-trees corresponding to $R_1^j, R_2^j, \dots, R_p^j$. It can be shown by induction on the number of components that the $PQ$-tree $T(F)$ constructed by using the function INSERT on all $T_i$, in the order $1, 2, \dots, p$, reducing the leaves labeled $v$ after every merge step and replacing them with a single representative represents all level planar embeddings of $F$. When proving the induction we differentiate between $v$-singular components and nonsingular components. In the latter case let $S_i$, $1 \leq i \leq p$, be the set of virtual vertices of $R_i^j$ except for vertex $v$, and let $S_{\{1,2,\dots,i\}}$, $1 \leq i \leq p$, be the set of virtual vertices of $R_{\{1,2,\dots,i\}}^j$ except for vertex $v$. The correctness of the merge operation can then be shown by proving first that if $\pi_{\{1,2,\dots,i\}}$, $i \leq p$, represents a level planar embedding of $R_{\{1,2,\dots,i\}}^j$, then the vertices of $S_i$ form a consecutive sequence in $\pi_{\{1,2,\dots,i\}}$ and the vertex $v$ is incident to $S_i$. Notice that this result is not true, if the order of the merge process is changed. This proves that the operations for merging $PQ$-trees are correct.

For completing the proof of correctness, let $v \in V_{j+1}$, $j < k$, and let $R_i^j$ be a level planar reduced extended form with $S_i^v \neq \emptyset$ and $|S_i^w| \leq 1$ for all $w = 1, 2, \dots, v-1$ such that $R_i^j$ has been constructed by $w$-merging several reduced extended forms. Let $T_i$ be the corresponding $PQ$-tree, representing all level planar embeddings of $R_i^j$. Let $F$ be the component constructed from $R_i^j$ by identifying all virtual vertices labeled $v$ to a single vertex $v$. It can be shown that the $PQ$-tree $T(F)$ constructed from $T_i$ by reducing $S_i^v$ in $T_i$, replacing the sequence of leaves corresponding to $S_i^v$ by a single representative, represents all level planar embeddings of $F$. □

Before determining the time complexity of the algorithm LEVEL-PLANAR-TEST, we determine the number of calls for REDUCE. Obviously, the number of calls for REDUCE in the first reduction phase is bounded by $|V|$, while the number of calls of REDUCE performed upon an successful INSERT operation is bounded by $s-1$, where $s$ denotes the number of sources of $G$. We show that at most $s-1$ extra REDUCE operations are necessary if the reduced extended forms are merged according to their size. This is not a trivial result, as has been stated by Heath and Pemmaraju (1996b). They observe that only one extra reduction is possible after every INSERT operation. This is only true for the first INSERT operation at a vertex $v$. If more components have to be merged, their observation is not true in general.

**Theorem 2.** *The algorithm LEVEL-PLANAR-TEST can be implemented to run in* $\mathrm{O}(|V|)$ *time for any proper leveled graph* $G = (V, E)$.

*Proof.* The linear time follows from an amortized analysis. We use the observation of Di Battista and Nardelli (1988) that in a level planar graph $|E| \leq 2|V| - 4$ holds. Heath and Pemmaraju (1996a,b) have shown that the overall number of operations that have to be performed on all calls of INSERT is bounded by $\mathrm{O}(|V|)$. The number of all operations performed on all calls of REDUCE during the first reduction phase is bounded by $\mathrm{O}(|V|)$ which follows from a similar argument as used by Booth and Lueker (1976) for the simple planarity test. The number of operations performed on all calls of REDUCE after a call of INSERT is proportional to the amount of work that has to be done in INSERT. Since the number of extra calls of REDUCE is bounded by one for each call of INSERT and the amount of work that has to be done for these extra calls is also proportional to the number of operations in INSERT, we conclude that the number of operations performed on all calls of REDUCE is bounded by $\mathrm{O}(|V|)$. Furthermore, we know that the total number of operations in order to update the leaves before merging *PQ*-trees is bounded by the number of operations performed in all calls of REDUCE. The total number of update operations that have to be performed after a merge phase of a level is complete is obviously bounded by $\mathrm{O}(|E|)$. Hence the total number of operations is bounded by $\mathrm{O}(|V|)$. □

## 4 Remarks

For simplicity, we restricted ourselves in this paper to the level planarity testing of proper leveled graphs. Of course, every non proper leveled graph can be transformed into a proper one by inserting dummy vertices. This strategy should not be applied since the resulting number of vertices may be quadratic in the original number of vertices. The following theorem shows that our level planarity test works on non proper leveled graphs as well as on proper leveled graphs, having a linear running time for both classes of leveled graphs.

**Theorem 3.** *The algorithm LEVEL-PLANAR-TEST tests any leveled graph* $G = (V, E)$ *for level planarity in* $\mathrm{O}(|V|)$ *time.*

*Proof.* Consider an edge $e = (v, w)$, $v \in V_j$, $w \in V_l$, $1 \leq j < l - 1 \leq k - 1$, traversing one or more levels. Inserting dummy vertices for $e$ in order to construct a proper hierarchy would result in a graph $G'$ such that every dummy vertex $u_i^e$, $i \in \{j+1, j+2, \ldots, l-1\}$ has exactly one incoming edge and one outgoing edge. However, the reduction of a *PQ*-tree $T$ with respect to a set $U$ with $|U| = 1$ and replacing the set by a new set $U'$ with $|U'| = 1$ is trivial and does not modify the $PQ$-tree. Hence we do not need to consider the dummy vertices and we therefore do not introduce them at all, yielding a linear level planarity test for general leveled graphs. □

An embedding of a general level planar graph $G = (V, E)$ can be computed in linear time as follows:

1. Add an extra vertex $t$ on an extra level $k + 1$ and compute a hierarchy by adding an outgoing edge to every sink without destroying level planarity.
2. Add an extra vertex $s$ on an extra level 0 and compute an $st$-graph by adding the edge $(s, t)$ and an incoming edge to every source without destroying the level planarity.
3. Compute a planar embedding using the algorithm by Chiba et al. (1985).
4. Construct a level planar embedding from the planar embedding.

The difficult part is to insert edges without destroying level planarity. We apply the following strategy. The idea is to determine the position of a sink $t \in V_j$, $j \in \{1, 2, \dots, k - 1\}$ by inserting an indicator as a leaf into the $PQ$-trees. This indicator is ignored throughout the application of the level planarity test and will be removed either with the leaves corresponding to the incoming edges of some vertex $v \in V_l$, $l \in \{j + 1, j + 2, \dots, k\}$, or it can be found in the final $PQ$-tree. However, this strategy is accompanied by a set of difficult case distinctions that are to be discussed in another paper. Nevertheless, the time needed to compute a level planar embedding is bounded by $\mathrm{O}(|V|)$ since the number of extra edges is bounded by the number of sinks and sources in $G$.

## References

[1976]K. Booth and G. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of Computer and System Sciences*, 13:335–379, 1976.

[1985]N. Chiba, T. Nishizeki, S. Abe, and T. Ozawa. A linear algorithm for embedding planar graphs using PQ-trees. *Journal of Computer and System Sciences*, 30:54–76, 1985.

[1988]G. Di Battista and E. Nardelli. Hierarchies and planarity theory. *IEEE Transactions on systems, man, and cybernetics*, 18(6):1035–1046, 1988.

[1996a]L.S. Heath and S.V. Pemmaraju. Recognizing leveled-planar dags in linear time. In F. J. Brandenburg, editor, *Proc. Graph Drawing '95*, volume 1027 of *Lecture Notes in Computer Science*, pages 300–311. Springer Verlag, 1996a.

[1996b]L.S. Heath and S.V. Pemmaraju. Stack and queue layouts of directed acyclic graphs: Part II. Technical report, Department of Computer Science, Virginia Polytechnic Institute & State University, 1996b.

[1997]M. Jünger, S. Leipert, and P. Mutzel. Pitfalls of using PQ-trees in Automatic Graph Drawing. In G. DiBattista, editor, *Graph Drawing '97*, volume 1353 of *Lecture Notes in Computer Science*, pages 193–204. Springer Verlag, 1997.