Level Planar Embedding in Linear Time*

Michael Jünger and Sebastian Leipert

Institut für Informatik, Universität zu Köln, 50969 Köln, Germany,
{mjuenger,leipert}@informatik.uni-koeln.de

Abstract. In a level 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 monotonically with respect to the vertical direction, and no edges intersect except at their end vertices.
In order to draw a level planar graph without edge crossings, a level planar embedding of the level graph has to be computed. Level planar embeddings are characterized by linear orderings of the vertices in each V^i $(1 \leq i \leq k)$. We present an $\mathcal{O}(|V|)$ time algorithm for embedding level planar graphs. This approach is based on a level planarity test by Jünger, Leipert, and Mutzel [6].

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. Many 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. The corresponding graphs are called level graphs. Using the *PQ*-tree data structure, Jünger et al. [6] have given an algorithm that tests in linear time whether such a graph is level planar, i.e. can be drawn without edge crossings.

In order to draw a level planar graph without edge crossings, a level planar embedding of the level graph has to be computed. Level planar embeddings are characterized by linear orderings of the vertices in each V^i $(1 \leq i \leq k)$. We present a linear time algorithm for embedding level planar graphs. Our approach is based on the level planarity test and augments a level planar graph G to an *st*-graph G_{st}, a graph with a single sink and a single source, without destroying the level planarity. Once the *st*-graph has been constructed, we compute a planar embedding of the *st*-graph. This is done by applying the embedding algorithm of Chiba et al. [2] for general graphs, obeying the topological ordering of the

* Supported by DFG-Grant Ju204/7-3, Forschungsschwerpunkt "Effiziente Algorithmen für diskrete Probleme und ihre Anwendungen"

J. Kratochvíl (Ed.): GD'99, LNCS 1731, pp. 72–81, 1999.

vertices in the *st*-graph. Exploiting the embedding of the *st*-graph G_{st}, we are able to determine a level planar embedding of G.

This extended abstract is organized as follows. After summarizing the necessary preliminaries in the next section, including a short introduction to the PQ-tree data structure and the level planarity test presented by Jünger et al. [6], we present in the third section the concept of the linear time level planar embedding algorithm. The fourth section concentrates on some details concerning the embedding algorithm. We close with some remarks on how to produce a level planar drawing using the results of our algorithm.

2 Preliminaries

Let $G = (V, E)$ be a directed acyclic graph. A *leveling* of G is a topological numbering $\text{lev} : V \to \mathbb{Z}$ mapping the vertices of G to integers such that $\text{lev}(v) \geq \text{lev}(u) + 1$ for all $(u, v) \in E$. G is called a *level graph* if it has a leveling. If $\text{lev}(v) = j$, then v is a *level-j vertex*. Let $V^j = \text{lev}^{-1}(j)$ denote the set of level-j vertices. Each V^j is a *level* of G. If $G = (V, E)$ has a leveling with k being the largest integer such that V^k is not empty, G is said to be a *k-level graph*. For a k-level graph G, we sometimes write $G = (V^1, V^2, \ldots, V^k; E)$. A level graph $G = (V, E)$ is said to be *proper* if every edge $e \in E$ connects only vertices belonging to consecutive levels. A *hierarchy* is a level graph such that all sources belong to the first level V^1.

A drawing of G in the plane is a *level drawing* 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 $(u, v) \in E$, $u \in V^i$, $v \in V^j$, $1 \leq i < j \leq k$, is drawn as a monotone curve between the lines l_i and l_j. A level drawing of G is called *level planar* if no two edges cross except at common endpoints. A level graph is *level planar* if it has a level planar drawing. A level graph G is obviously level planar if and only if all its components are level planar.

A level drawing 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 vertices on l_j. A *level embedding* consists of a permutation of the vertices of V^j for every $j \in \{1, 2, \ldots, k\}$ with respect to a level drawing. A level embedding with respect to a level planar drawing is called *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 [1] 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 may only 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 [1] 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.

Let G^j denote the subgraph of G induced by $V^1 \cup V^2 \cup \cdots \cup V^j$. The strategy of testing the level planarity 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, and every component of G^k the set of permutations of the vertices of V^j that appear in some level planar embedding of G^j. Obviously, the graph $G = G^k$ is level planar, if and only if the set of permutations for G^k is not empty. This test implies for every level planar component of G^j that the set of its level planar embeddings can be represented by a PQ-tree.

As long as different components of G^j are not adjacent to a common vertex on level $j+1$, standard PQ-tree techniques using the function REDUCE are applied for constructing the PQ-tree of every component. If two or more components are adjacent to a common vertex v on level $j+1$, they have to be "merged" and a new PQ-tree has to be constructed from the two corresponding PQ-trees.

The merge operation 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, $1 \leq j \leq k$, that belong to a component R_i^j (the i^{th} component of G^j), define ML(S) to be the greatest $d \leq j$ such that $V^d, V^{d+1}, \ldots, V^j$ induces a subgraph in which all vertices of S occur in the same connected component. The level ML(S) is said to be the *meet level* of S. For a Q-node Y in the corresponding PQ-tree $T(R_i^j)$ with ordered children $Y_1, Y_2, \ldots, Y_t$ integers denoted by ML(Y_i, Y_{i+1}), $1 \leq i < t$, are maintained satisfying ML(Y_i, Y_{i+1}) = ML(frontier(Y_i)$\cup$frontier(Y_{i+1})). For a P-node X a single integer denoted by ML(X) that satisfies ML(X) = ML(frontier(X)).

Furthermore, define LL(R_i^j), the *low indexed level*, to be the smallest d such that R_i^j contains a vertex in V^d and maintain this integer at the root of the corresponding PQ-tree. The *height* of a component R_i^j in the subgraph G^j is $j - \text{LL}(R_i^j)$. The LL-value merely describes the size of the component.

Using these LL- and ML-values, Heath and Pemmaraju [5] have developed operations for merging PQ-trees. These merge operations have been modified and adapted into a larger framework by Jünger et al. [6] to develop an $\mathcal{O}(|V|)$ time algorithm for testing level planarity of not necessarily proper level graphs.

3 Concept of the Algorithm

One can easily obtain the following naive embedding algorithm for level planar graphs. Choose any total order on V^k that is consistent with the set of permutations of V^k that appear in a level planar embedding of $G^k = G$. Choose then any total order on V^{k-1} that is consistent with the set of permutations of V^{k-1} that appear in a level planar embedding of G^{k-1} and that, together with the chosen order of V^k implies a level planar embedding on the subgraph of G

induced by $V^{k-1} \cup V^k$. Extend this construction one level at a time until a level planar embedding of G results.

However, to perform this algorithm, it is necessary to keep track of the set of PQ-trees of every level l, $1 \leq l \leq k$. Besides, an appropriate total order of the vertices of V^j, $1 \leq j < k$, can only be detected by reducing subsets of the leaves of G^j, where the subsets are induced by the adjacency lists of the vertices of V^{j+1}. More precisely, for every pair of consecutive edges $e_1 = (v_1, w)$, $e_2 = (v_2, w)$, $v_1, v_2 \in V^j$, in the adjacency list of a vertex $w \in V^{j+1}$, we have to reduce the set of leaves corresponding to the vertices v_1, v_2 in $\mathcal{T}(G^j)$. This immediately yields an $\Omega(n^2)$ algorithm for nonproper level graphs, with $\Omega(n^2)$ dummy vertices for long edges, since we are forced to consider for every long edge its exact position on the level that is traversed by the long edge.

Instead, we proceed as follows: Let $G = (V, E)$ be a level planar graph with leveling $\mathrm{lev}_G : V \to \{1, 2, \ldots, k\}$. We augment G to a planar directed acyclic st-graph $G_{st} = (V_{st}, E_{st})$ where $V_{st} = V \uplus \{s, t\}$ and $E \subset E_{st}$ such that every source in G has exactly one incoming edge in $E_{st} \setminus E$, every sink in G has exactly one outgoing edge in $E_{st} \setminus E$, $\mathrm{lev}_{G_{st}}(s) = 0$, $\mathrm{lev}_{G_{st}}(t) = k+1$ and for all $v \in V$ we have $\mathrm{lev}_{G_{st}}(v) = \mathrm{lev}_G(v)$. This process, in which two vertices and $\mathcal{O}(|V|)$ edges are added to G, is the nontrivial part of the algorithm that will be explained in section 4.

We compute a topological sorting, i.e., an onto function $\mathrm{ts}_{G_{st}} : V \to \{0, 1, \ldots, |V|+1\}$. The function $\mathrm{ts}_{G_{st}}$ is comparable with $\mathrm{lev}_{G_{st}}$ in the sense that for every $v, w \in V_{st}$ we have $\mathrm{ts}_{G_{st}}(v) \leq \mathrm{ts}_{G_{st}}(w)$ if and only if $\mathrm{lev}_{G_{st}}(v) \leq \mathrm{lev}_{G_{st}}(w)$. Obviously $\mathrm{ts}_{G_{st}}$ is an st-numbering of G_{st}. Using this st-numbering, we can obtain a planar embedding $\mathcal{E}_{st}$ of G_{st} with the edge (s, t) on the boundary of the outer face by applying the algorithm of Chiba et al. [2].

From the planar embedding we obtain a level planar embedding of G_{st} by applying a function "CONSTRUCT-LEVEL-EMBED" that uses a depth first search procedure starting at vertex t and proceeding from every visited vertex w to the unvisited neighbor that appears first in the clockwise ordering of the adjacency list of w in $\mathcal{E}_{st}$. Initially, all levels are empty. When a vertex w is visited, it is appended to the right of the vertices assigned to level $\mathrm{lev}_{G_{st}}(w)$. The restriction of the resulting level orderings to the levels 1 to k yields a level planar embedding of G.

It is clear that the described algorithm runs in $\mathcal{O}(|V|)$ time if the nontrivial part, namely the construction of G_{st} can be achieved in $\mathcal{O}(|V|)$ time. After adding the vertices s and t we augment G to a hierarchy by adding an outgoing edge to every sink of G without destroying level planarity using a function AUGMENT, processing the graph top to bottom. Using the same function AUGMENT again, we process the graph bottom to top and augment G_{st} to an st-graph by adding the edge (s, t) and an incoming edge to every source of G without destroying the level planarity. Thus our level planar embedding algorithm can be sketched as follows.

$\mathcal{E}_l$ **LEVEL-PLANAR-EMBED**$(G = (V^1, V^2, \ldots, V^k; E))$
begin
  ignore all isolated vertices;
  expand G to G_{st} by adding $V^0 = \{s\}$ and $V^{k+1} = \{t\}$;
  AUGMENT(G_{st});
  if AUGMENT fails then
    return $\mathcal{E}_l = \emptyset$;
  //G_{st} is now a hierarchy;
  orient the graph G_{st} from the bottom to the top;
  AUGMENT(G_{st});
  //G_{st} is now an st-graph;
  orient the graph G_{st} from the top to the bottom;
  add edge (s, t);
  compute a topological sorting of V_{st};
  compute a planar embedding $\mathcal{E}_{st}$ according to Chiba et al. [2]
    using the topological sorting as an st-numbering;
  $\mathcal{E}_l$ = CONSTRUCT-LEVEL-EMBED($\mathcal{E}_{st}, G_{st}$);
  return $\mathcal{E}_l$;
end.

4 Augmentation

In order to add an outgoing edge for every sink of G without destroying level planarity, we need to determine the position of a sink $v \in V^j$, $j \in \{1, 2, \ldots, k-1\}$, in the PQ-trees. This is done by inserting an indicator as a leaf into the PQ-trees. The 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 $w \in V^l$, $l > j$, or it can be found in the final PQ-tree.

We explain the idea of the approach by an example. Let $v \in V^j$ be a sink. The leaf corresponding to the sink v will be removed from the PQ-tree before testing the graph G^{j+1} for level planarity. Instead of removing the leaf, we keep it in the tree and ignore its presence in the PQ-tree from now on. Such a leaf that marks the position of a sink v in a PQ-tree is called a *sink indicator* and denoted by $\mathrm{si}(v)$. The indicator of v may appear within the sequence of leaves corresponding to incoming edges of a vertex $w \in V^l$. The indicator of v is interpreted as a leaf corresponding to an edge $e = (v, w)$ and G is augmented by e. Adding the edge e to G does not destroy the level planarity and provides an outgoing edge for the sink v.

When replacing a leaf corresponding to a sink by a sink indicator, a P- or Q-node X may be constructed in the PQ-tree such that frontier(X) consists only of sink indicators. The presence of such a node in the PQ-tree is ignored as well. A node of a PQ-tree is an *ignored* node if and only if its frontier contains only sink indicators. By definition, a sink indicator is also an ignored node.

In order to achieve linear time for the level planar embedder, we must avoid searching for sink indicators that can be considered for augmentation. Conse-

quently, only the indicators $\mathrm{si}(v)$, $v \in V$, that appear within the pertinent subtree of a PQ-tree with respect to a vertex $w \in V$ are considered for augmentation. It is easy to see that the edges added this way do not destroy level planarity (see Leipert [7]).

However, not all sink indicators are considered for edge insertion. Some of the indicators remain in the final PQ-tree that represents all possible permutations of vertices of V^k in the level planar embeddings of G. It is straightforward to see that if $\mathrm{si}(v)$ is a sink indicator of a vertex $v \in V^j$, $1 < j < k$ and if $\mathrm{si}(v)$ is in the final PQ-tree T, the edge $e = (v, t)$ can be added without destroying level planarity.

While the treatment of sink indicators during the application of the template matching algorithm is rather easy in principle, this does not hold for merge operations. We consider one of the merge operations and discuss necessary adaptations in order to treat the sink indicators correctly. For manipulating sink indicators and ignored nodes correctly during the merge process, ML-values as they have been introduced for nonignored nodes are introduced for ignored nodes as well.

Let R_1^j and R_2^j be two components of G^j that are incident to a vertex w on level $j+1$ and let T_1 and T_2 be their corresponding PQ-trees. Without loss of generality, we may assume that $\mathrm{LL}(T_1) \leq \mathrm{LL}(T_2)$. Thus component R_2^j is the smaller component and an embedding of R_1^j must be found such that R_2^j can be nested within the embedding of R_1^j. This corresponds to adding the root of T_2 as a child to a node of the PQ-tree T_1 constructing a new PQ-tree T'. In order to find an appropriate location to insert T_2 into T_1, we start with the leaf labeled w (that corresponds to the vertex w in R_1^j) in T_1 and proceed upwards in T_1 until a node X' and its parent X are encountered that satisfy certain conditions.

In this extended abstract, we consider only the most difficult condition: Let X be a Q-node with ordered children $X_1, X_2, \ldots, X_\eta$, $X' = X_\lambda$, $1 < \lambda < \eta$, and $\mathrm{ML}(X_{\lambda-1}, X_\lambda) < \mathrm{LL}(T_2)$ and $\mathrm{ML}(X_\lambda, X_{\lambda+1}) < \mathrm{LL}(T_2)$. The node X_λ is replaced by a Q-node Y with two children, X_λ and the root of T_2.

Let $I_1, I_2, \ldots, I_\mu$, $\mu \geq 0$, be the sequence of ignored nodes between $X_{\lambda-1}$ and X_λ in which $X_{\lambda-1}$ and I_1 are direct siblings, and X_λ and I_μ are direct siblings. Let $J_1, J_2, \ldots, J_\rho$, $\rho \geq 0$, be the sequence of ignored nodes between X_λ and $X_{\lambda+1}$ in which X_λ and J_1 are direct siblings, and $X_{\lambda+1}$ and J_ρ are direct siblings.

Let R_{X_i}, $i \in \{1, 2, \ldots, \eta\}$, denote the subgraph of R_1 corresponding to X_i. As illustrated in Fig. 1, there may exist a ν, $1 \leq \nu \leq \mu$, such that for every sink indicator

$$\mathrm{si}(v) \in \bigcup_{i=\nu}^{\mu} \mathrm{frontier}(I_i) \ , \quad v \in \bigcup_{i=1}^{\mathrm{lev}(w)-1} V^i \ ,$$

G has to be augmented by an edge $e = (v, w)$ if R_2 is embedded between $R_{X_{\lambda-1}}$ and R_{X_λ}.

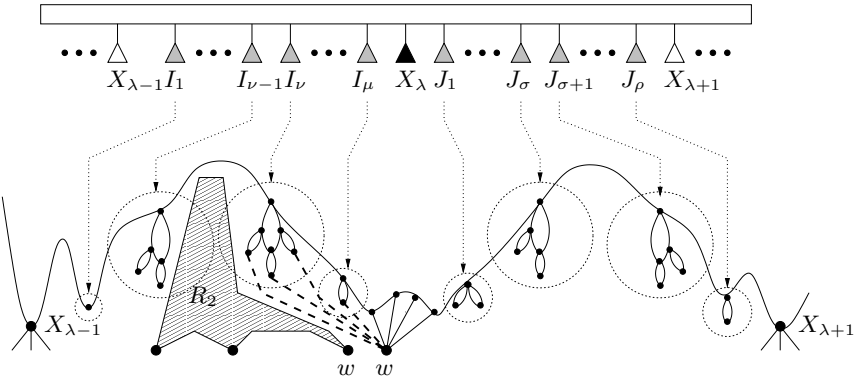

Fig. 1. Merging R_2 into R_1 and embedding it between $R_{X_{\lambda-1}}$ and R_{X_λ} forces G to be augmented by the edges drawn as dotted lines.

Obviously, R_2 can also be embedded between R_{X_λ} and $R_{X_{\lambda+1}}$, and there may exist a σ, $1 \leq \sigma \leq \rho$, such that for every sink indicator

$$\mathrm{si}(v) \in \bigcup_{i=1}^{\sigma} \mathrm{frontier}(J_i) \ , \quad v \in \bigcup_{i=1}^{\mathrm{lev}(w)-1} V^i \ ,$$

G has to be augmented by an edge $e = (v, w)$.

However, it is not possible to decide which set of sink indicators has to be considered for edge augmentation. Proceeding the level planarity test down the levels $V^{\mathrm{lev}(w)+1}$ to V^k may embed the component R_2 on either of the two sides of R_{X_λ}. Since the side is unknown during the merge operation, we have to keep the affected sink indicators in mind. Furthermore, we must devise a method that allows the determination of the correct embedding during subsequent reductions.

The sequences $I_\nu, I_{\nu+1}, \ldots, I_\mu$ and $J_1, J_2, \ldots, J_\sigma$ are called the *reference sequences* of R_2 and denoted by $\mathrm{rseq}(R_2)$. We refer to $I_\nu, I_{\nu+1}, \ldots, I_\mu$ as the *left reference sequence* of R_2 denoted by $\mathrm{rseq}(R_2)^{left}$, and to $J_1, J_2, \ldots, J_\sigma$ as the *right reference sequence* denoted by $\mathrm{rseq}(R_2)^{right}$. The union $\bigcup_{i=\nu}^{\mu} \mathrm{frontier}(I_i) \cup \bigcup_{i=1}^{\sigma} \mathrm{frontier}(J_i)$ is called the *reference set* of R_2 and denoted by $\mathrm{ref}(R_2)$. The *left* and *right reference set* $\mathrm{ref}(R_2)^{left}$ and $\mathrm{ref}(R_2)^{right}$, respectively, are defined analogously to the left and right reference sequence.

In order to solve the decision problem in the described merge operation, we examine how R_2 is fixed to either side of the vertex $w \in V$ in a level planar embedding of G. The following two nontrivial lemmas (see [7]) are based on two template replacement patterns Q2 and Q3 as they are used in the template reduction of Boot and Lueker [1].

Lemma 1. *The subgraph R_2 must be fixed in its embedding at one side of R_{X_λ} with respect to R_X if and only if the Q-node Y is removed from the tree T during the application of the template matching algorithm using template Q2 or template*

Q3, and the parent of Y did not become a node with Y as the only nonignored child.

Lemma 2. *The subgraph R_2 is not fixed to any side of R_{X_λ} with respect to R_X if and only if one of the following conditions applies during the application of the template matching algorithm.*

- *The Q-node Y gets ignored.*
- *The Q-node Y is a nonignored node of the final PQ-tree.*
- *The Q-node Y has only one nonignored child.*
- *The parent of Y has only Y as a nonignored child.*

Lemmas 1 and 2 reveal a solution for solving the problem of deciding whether or not R_2 is fixed to one side of R_{X_λ} with respect to R_X. A strategy is developed for detecting on which side of R_{X_λ} the subgraph R_2 has to be embedded. One endmost child of Y can clearly be identified with the side where the root of T_2 has been placed, while the other endmost child of Y can be identified with the side were X_λ is. Every reversion of the Q-node Y corresponds to changing the side were R_2 must be embedded and all we need to do is to detect the side of Y that belongs to R_2, when finally removing Y from the tree by applying one of the templates Q2 or Q3. The strategy is to mark the end of Y belonging to R_2 with a special ignored node. Such a special ignored node is called a *contact* of R_2 and denoted by $\mathrm{c}(R_2)$. During the merge operation, it is placed as an endmost child of Y next to the root of T_2. Thus the Q-node Y has now three children instead of two. See Fig. 2 for an illustration.

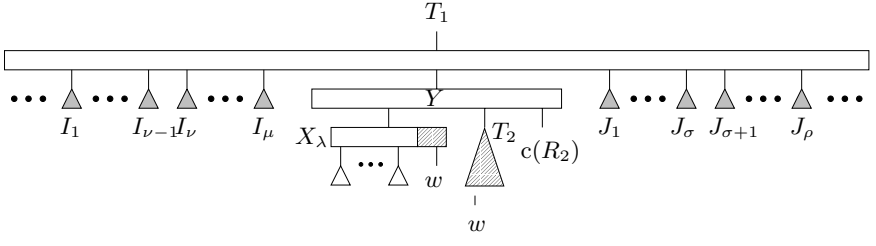

Fig. 2. Contact $\mathrm{c}(R_2)$ is added as a child to Y next to the root of T_2 after the PQ-trees T_1 and T_2 have been merged.

Before gathering some observations about contacts, it is necessary to show that the involved ignored nodes remain in the relative position of Y within the Q-node, and are therefore not moved or removed.

Lemma 3. *The ignored nodes of* $\mathrm{rseq}(R_2)^{left}$ *and* $\mathrm{rseq}(R_2)^{right}$ *stay siblings of Y until one of the templates Q2 or Q3 is applied to Y and its parent.*

A contact has some special attributes that are immediately clear and very useful for our approach. In the following observations we again assume that Y and its parent have not been subject to another merge operation.

Observation 1 *Since the contact is an endmost child of a Q-node Y, it will remain an endmost child of the same Q-node Y, unless the node Y is eliminated by applying one of the templates Q2 or Q3.*

Observation 2 *If the node Y is eliminated by applying one of the templates Q2 or Q3, the contact $\mathrm{c}(R_2)$ determines the side were R_2 must be embedded next to R_{X_λ} with respect to R_X. The contact is then a direct sibling to $\mathrm{rseq}(R_2)^i$ for some $i \in \{left, right\}$ and $\mathrm{ref}(R_2)^i$ must be considered for edge augmentation.*

Observation 3 *If one of the four cases mentioned in Lemma 2 applies to Y or its parent, R_2 can be embedded on any side R_{X_λ} with respect to R_X and therefore either $\mathrm{ref}(R_2)^{left}$ or $\mathrm{ref}(R_2)^{right}$ must be considered for edge augmentation.*

Besides placing $\mathrm{c}(R_2)$ as an endmost child next to the root of T_2, $\mathrm{c}(R_2)$ is equipped with a set of four pointers, denoting the beginning and the end of both the left and the right reference sequence of R_2. After performing a reduction by applying one of the templates Q2 or Q3 to the node Y, the contact is either a direct sibling of I_μ or a direct sibling of J_1. In the first case, we scan the sequence of ignored siblings starting at I_μ until the ignored node I_ν is detected. In the latter case, the sequence of ignored siblings is scanned by starting at J_1 until the node J_σ is detected. Figure 3 illustrates this strategy for the latter case. By storing pointers of the ignored nodes $I_\nu, I_\mu, J_1, J_\sigma$ at $\mathrm{c}(R_2)$, we are able to identify the reference set $\mathrm{ref}(R_2)$.

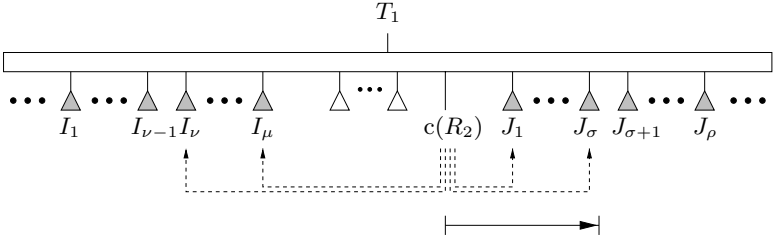

Fig. 3. Identification of the reference set that has to be chosen for augmentation. Contact $\mathrm{c}(R_2)$ is adjacent to the ignored node J_1 after the application of template Q2 or Q3. We chose $\mathrm{ref}(R_2)^{right}$ for augmentation. The dotted lines denote the pointers of $\mathrm{c}(R_2)$ to $I_\nu, I_\mu, J_1, J_\sigma$.

For clarity, we omitted the concatenation of merge operations. However, straightforward methods in the application of contacts are able to handle such concatenations (see [7] for details).

Theorem 4. *The algorithm LEVEL-PLANAR-EMBED computes a level planar embedding of a level planar graph $G = (V, E)$ in $\mathcal{O}(|V|)$ time.*

Proof. The correctness follows from the correctness of the function AUGMENT and the discussion of section 3. The linear running time of AUGMENT follows from an amortized analysis based on the linear running time of the level planar testing algorithm.

5 Remarks

Once a level graph has been level planar embedded, we want to visualize it by producing a level planar drawing. There is a nice and quick solution to this problem that uses some extra information that is computed by our level planar embedding algorithm. Instead of drawing the graph G directly, we draw the st-graph G_{st} using the planar embedding of G_{st}, and remove the vertices s and t as well as all edges in $E_{st} \setminus E$ afterwards.

Suitable approaches for drawing an st-graph G_{st} have been presented in [3] and [4]. These algorithms construct a planar upward polyline drawing of a planar st-graph according to a topological numbering of the vertices. The vertices of the st-graph are assigned to grid coordinates and the edges are drawn as polygonal chains. If we assign a topological numbering to the vertices according to their leveling, the algorithm presented by Di Battista and Tamassia [3] produces in $\mathcal{O}(|V|)$ time a level planar polyline grid drawing of G_{st} such that the number of edge bends is at most $6n - 12$ and every edge has at most two bends. This approach can be improved to produce in $\mathcal{O}(|V|)$ time a level planar polyline grid drawing of G_{st} such that the drawing of G_{st} has $\mathcal{O}(|V|^2)$ area, the number of edge bends is at most $(10n-31)/3$, and every edge has at most two bends. Thus once we have augmented G to the st-graph G_{st}, we can immediately produce a level planar drawing of G in $\mathcal{O}(|V|)$ time.

References

1. 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.
2. 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.
3. G. Di Battista and R. Tamassia. Algorithms for plane representations of acyclic digraphs. *Theoretical Computer Science*, 61:175–198, 1988.
4. G. Di Battista, R. Tamassia, and I. G. Tollis. Constrained visibility representations of graphs. *Information Processing Letters*, 41:1–7, 1992.
5. 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, 1995.
6. M. Jünger, S. Leipert, and P. Mutzel. Level planarity testing in linear time. In S. Whitesides, editor, *Graph Drawing '98*, volume 1547 of *Lecture Notes in Computer Science*, pages 224–237. Springer Verlag, 1998.
7. S. Leipert. *Level Planarity Testing and Embedding in Linear Time.* PhD thesis, Universität zu Köln, 1998.