Angewandte Mathematik und
Informatik
Universität zu Köln

Report No. 99.374

Level Planar Embedding in linear Time

by

Michael Jünger, and Sebastian Leipert

1999

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

Institut für Informatik
Universität zu Köln
Pohligstraße 1
50969 Köln

1991 Mathematics Subject Classification: 05C85, 68R10, 90C35
Keywords: Level Planararity, Level Planar Embedding, PQ-Trees, Graph Drawing

Level Planar Embedding in Linear Time

Michael Jünger[a]
Sebastian Leipert[b]

[a] Institut für Informatik, Universität zu Köln, mjuenger@informatik.uni-koeln.de
[b] Institut für Informatik, Universität zu Köln, leipert@informatik.uni-koeln.de

Abstract

A level graph $G = (V, E, \phi)$ is a directed acyclic graph with a mapping $\phi : V \to \{1, 2, \dots, k\}$, $k \geq 1$, that partitions the vertex set V as $V = V^1 \cup V^2 \cup \dots \cup V^k$, $V^j = \phi^{-1}(j)$, $V^i \cap V^j = \emptyset$ for $i \neq j$, such that $\phi(v) \geq \phi(u) + 1$ for each edge $(u, v) \in E$. 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 [1998].

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, Leipert, and Mutzel [1998] 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. [1985] for general graphs, obeying the topological ordering of the vertices in the st-graph. Exploiting the planar embedding of the st-graph G_{st}, we are able to determine a level planar embedding of G.

This paper is organized as follows. After summarizing the necessary preliminaries in the next section, including the PQ-tree data structure we give a short introduction to the level planarity test presented by Jünger et al. [1998] in the third section. In the fourth section, we present the concept of the linear time level planar embedding algorithm. The Sections 5 to 8 contain the details of the embedding algorithm. We close with some remarks on how to produce a level planar drawing using the result of our algorithm.

2 Preliminaries

A level graph $G = (V, E, \phi)$ is a directed acyclic graph with a mapping $\phi : V \rightarrow \{1, 2, \ldots, k\}$, $k \geq 1$, that partitions the vertex set V as $V = V^1 \cup V^2 \cup \cdots \cup V^k$, $V^j = \phi^{-1}(j)$, $V^i \cap V^j = \emptyset$ for $i \neq j$, such that $\phi(v) \geq \phi(u) + 1$ for each edge $(u, v) \in E$. A vertex $v \in V^j$ is called a *level-j vertex* and V^j is called the *j-th level* of G. For a level graph $G = (V, E, \phi)$, we sometimes write $G = (V^1, V^2, \ldots, V^k; E)$.

A drawing of a level graph 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 monotonically decreasing 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. We therefore may assume in the following without loss of generality that G is connected.

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 level graph $G = (V, E)$ is said to be proper, if every edge $e \in E$ connects only vertices belonging to consecutive levels. Usually, k-level graph G have sinks and sources placed on various levels of the graph.

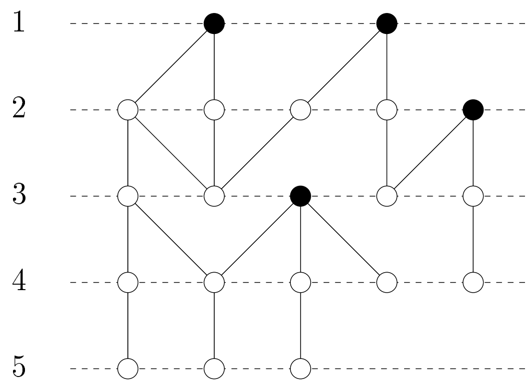


Figure 1: An examples of a level graph. Sources are drawn black.

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 (see, e.g., Fulkerson and Gross [1965]). A PQ-tree is a rooted and ordered tree that 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. In subsequent figures, P-nodes are drawn as circles while Q-nodes are drawn as rectangles.

The *frontier* of a PQ-tree T, denoted by frontier(T), is the sequence of all leaves of T read from left to right, and the frontier of a node X, denoted by frontier(X), is the sequence of its descendant leaves read from left to right. The frontier of a PQ-tree is a permutation of the set U. We use the notion frontier(T) and frontier(X) also to denote the set of elements in frontier(T) and frontier(X), respectively, its meaning being evident by context. An *equivalence transformation* specifies a legal reordering of the nodes within a PQ-tree. The only legal equivalence transformations are

(i) any permutation of the children of a P-node, and

(ii) the reverse permutation of the children of a Q-node.

Two PQ-trees T and T' are *equivalent* if and only if their underlying trees are equal and T can be transformed into T' by a sequence of equivalence transformations. The equivalence of two PQ-trees is denoted $T \equiv T'$. The set of *consistent permutations* of a PQ-tree is the set of all frontiers that can be obtained by a sequence of equivalence transformations and is denoted by

$$\mathrm{PERM}(T) \quad = \quad \{\mathrm{frontier}(T') \mid T' \equiv T\} \ .$$

If two nodes X and Y of a PQ-tree have the same parent, they are *siblings*. The nodes are called *adjacent* or *direct* if they are siblings and appear consecutively in the order of children of their parent.

Let $\Pi_{S_i} := \{\pi \in \Pi \mid \text{all elements of } S_i \text{ are consecutive within } \pi\}$. Given any PQ-tree T over a finite set U and a subset $S \subseteq U$, the function REDUCE(T, S) computes a PQ-tree T' such that PERM(T') = PERM$(T) \cap \Pi_S$. The function REDUCE applies a sequence of *templates* to the nodes of a PQ-tree starting at the leaves, and proceeding upwards until the root of the pertinent subtree is reached. Each template has a *pattern* and a *replacement*. If a node matches the pattern of a template, the pattern is replaced within the tree by the replacement of the template. The return value of REDUCE is a new PQ-tree. It is the *null tree*, a tree with no nodes at all, if the original tree could not be reduced for the specified set S. If a null tree is returned, the set of permissible permutations on the set U is empty and the null tree represents an empty set of permutations. Therefore it is convenient to denote the null tree by $\emptyset$. A node X in T is said to be *full* if frontier$(X) \subseteq S$. A node X is said to be *empty* if frontier$(X) \cap S = \emptyset$. A node X is *partial* if it is neither empty nor full. Nodes are said to be *pertinent* if they are either full or partial.

Each template specifies a local change within the tree. Only the node X that has to be matched and its children are altered. The patterns to which nodes are matched depend upon the set S and the frontier of the subtree rooted at the particular node X. The matched pattern is selected by examining the node X and its children after the children themselves have been matched. Depending on the situation in the frontier of X the node is labeled indicating whether X is empty, full, or partial. This bottom-up strategy ensures that all information on the situation in the frontier of the children of X is available when processing X.

In Fig. 2 and Fig. 3 we illustrate two of the template matchings, the templates Q2 and Q3 (see Booth and Lueker [1976] for the templates P1 – P6 and Q1). The pattern at the left hand side is to be transformed into a pattern at the right hand side. A full node or a full subtree is hatched, and a partial Q-node that roots a pertinent subtree is hatched partially. We use a triangle for symbolizing a subtree. A subtree is either full or empty, so its precise form has no effect on the templates.

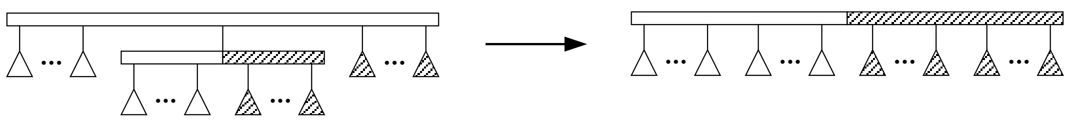


Figure 2: Template Q2.

Theorem 2.1 (Booth and Lueker [1976]). *The data structure PQ-tree and the template matchings can be implemented such that the class of permutations in which the elements of each set S_i of a family $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ of subsets of U occur as a consecutive sequence can be computed in $\mathcal{O}(|U| + n + \sum_{i=1}^{n} |S_i|)$ time.*

One result that is achieved in the proof of Theorem 2.1 is the following corollary that

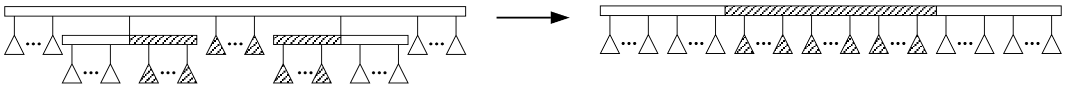


Figure 3: Template Q3.

is needed later when proving the correctness of a level planar embedding algorithm.

Corollary 2.2 (Booth and Lueker [1976]). *Let X be a child of a Q-node Y. Throughout the template matching algorithm X remains a child of a Q-node.*

3 A Linear Time Level Planarity Test

In this section we give a short introduction to a level planarity test as it has been presented in Jünger et al. [1998, 1999]. Let G^j denote the subgraph of G induced by $V^1 \cup V^2 \cup \cdots \cup V^j$. The basic idea is to perform a top-down sweep, processing the levels in the order $V^1, V^2, \ldots, V^k$. The graph G^j, $1 \leq j < k$, is not necessarily connected, and a separate PQ-tree is introduced for every component F of G^j to represent the set of permutations of the vertices of F in V^j that appear in some level planar embedding of G^j. The P-nodes of such a PQ-tree correspond to cut vertices in F, the Q-nodes to connected components with a fixed embedding that can only be reversed. The leaves correspond either to level-j vertices or to incoming edges of level-l vertices, with $l > j$.

Performing the top-down sweep, standard PQ-tree techniques are applied, as long as different components of G^j are not adjacent to a common vertex on level j. If two components are adjacent to a common vertex v on level j, they have to be merged and a new PQ-tree has to be constructed from the two corresponding PQ-trees. The new PQ-tree then represents all level planar embeddings of the merged component. Applying a combination of reduce operations and merge operations for combining PQ-trees, we maintain for every level V^j and for every component F of G^j the set of permutations of the vertices of F in V^j that appear in some level planar embedding of G^j. If the set of permutations for G^k is not empty, the graph $G = G^k$ is obviously level planar.

Before we describe our algorithm, called LEVEL-PLANARITY-TEST, let us introduce some new terminology. Let X be a Q-node in T_i corresponding to a subgraph B of G^j, $1 \leq j \leq k$. The children of X each correspond to a cut vertex on the border of the outer face of B (see also Booth and Lueker [1976], Leipert [1998]). If X is not the root, then there exists an extra cut vertex on the border of the outer face of b that separates the subgraph G' induced by the subtree rooted at X from $G^j - G'$. This cut vertex is called the *connective cut vertex* of B.

Since G^j is not necessarily connected, let m_j denote the number of components of G^j and let F_i^j, $i = 1, 2, \ldots, m_j$, denote the components of G^j. Figure 4 shows a G^4 with $m_4 = 2$ components F_1^4 and F_2^4. The set of vertices in F_i^j is denoted by $V(F_i^j)$. Define $\mathrm{LL}(F_i^j)$, the *low indexed level*, to be the smallest d such that F_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 F_i^j in the subgraph G^j is $j - \mathrm{LL}(F_i^j)$. The LL-value merely describes the size of the component. The LL-values of the two components shown in Fig. 4 are $\mathrm{LL}(F_1^4) = 1$ and $\mathrm{LL}(F_2^4) = 2$.

Let H_i^j be the graph arising from F_i^j as follows: For each edge $e = (u, v)$, where u is a vertex in F_i^j and $v \in V^l$, $l \geq j + 1$, we introduce a *virtual vertex* with label v and a *virtual edge* that connects u and this virtual vertex. 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 form H_i^j is called the *extended form* of F_i^j and the set of virtual vertices of H_i^j is denoted by $\mathrm{frontier}(H_i^j)$. Figure 5 shows possible extended forms H_1^4 and H_2^4 of the example in Fig. 4. The virtual vertices on level 5 are denoted by their labels. The frontier of H_1^4 consists of one virtual vertex labeled u, two vertices labeled v, and two vertices labeled w. The set of virtual vertices of H_i^j that are labeled $v \in V^{j+1}$ is denoted by S_i^v.

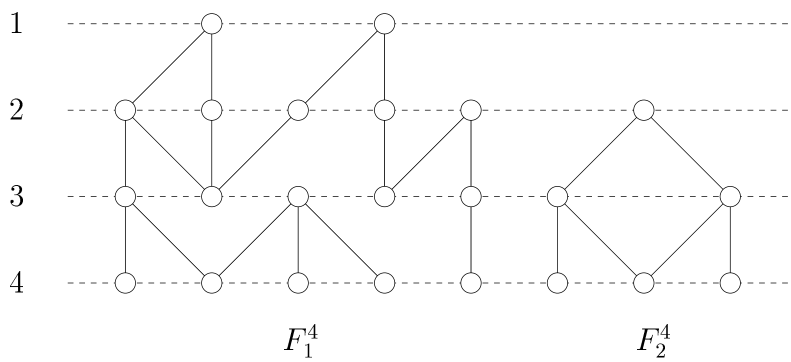


Figure 4: A G^4 with $m_2 = 2$ components F_1^4 and F_2^4.

The graph that is created from an extended form H_i^j by identifying all virtual vertices with the same label to a single vertex is called *reduced extended form* and denoted by R_i^j. To construct R_1^4 from the example component H_1^4, the vertices labeled w have to be identified and the vertices labeled v have to be identified. In order to identify the two vertices labeled x in H_2^4 for the construction of R_2^4, it is necessary to permute the left most vertex labeled x and the vertex labeled v. Both forms R_1^4 and R_2^4 then have exactly one vertex labeled v.

The set of virtual vertices of R_i^j is denoted by $\mathrm{frontier}(R_i^j)$. If S_i^v of H_i^j is not empty, we denote the vertex with label v of R_i^j (i.e., the vertex that arose from identifying all virtual vertices of S_i^v) by v_i and update $S_i^v = \{v_i\}$. The graph arising from the identification of two virtual vertices v_i and v_l (labeled v) of two reduced extended

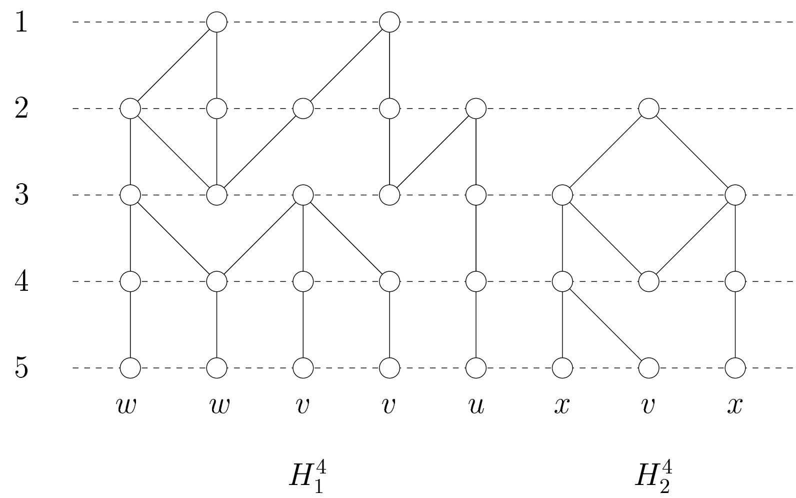


Figure 5: Two extended forms H_1^4 and H_2^4 of Fig. 4.

forms R_i^j and R_l^j is denoted $R_i^j \cup_v R_l^j$. We call $R_i^j \cup_v R_l^j$ a *merged reduced form*. The vertex arising from the identification of v_i and v_l is denoted by $v_{\{i,l\}}$ (and labeled by v of course). 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 form that is created by v-merging R_l^j into R_i^j and identifying all virtual vertices with the same label $w \neq v$ is again a reduced extended form and denoted by R_i^j (thus renaming $R_i^j \cup_v R_l^j$ with the name of the "higher" form). Figure 6 shows the resulting merged reduced extended form $R_1^4 \cup_v R_2^4$ after R_2^4 (the smaller form) has been v-merged into R_1^4 (the higher form). Since R_1^4 is the higher form, $R_1^4 \cup_v R_2^4$ is renamed into R_1^4.

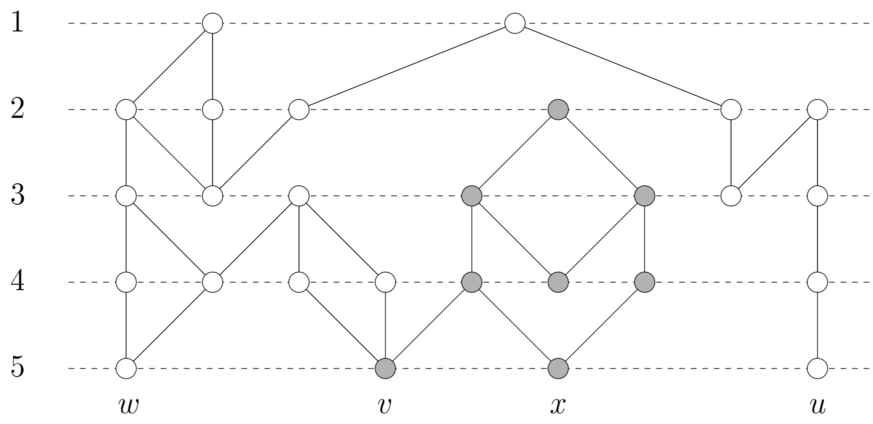


Figure 6: A merged reduced extended form $R_1^4 \cup_v R_2^4$ after R_2^4 has been v-merged into R_1^4. The former vertices of R_2^4 are drawn shaded.

We omit scanning for leaves with the same label after we have v-merged several reduced extended forms. This is done in order to achieve linear running time. However, this strategy results in improper reduced extended forms, possibly having several virtual vertices with the same label. These forms are called *partially reduced extended forms*.

If any reduced extended form has been v-merged into R_i^j, the form R_i^j is called *v-connected*, otherwise R_i^j is called *v-unconnected*. The form R_1^4 shown in Fig. 6 is v-connected.

A reduced extended form R_i^j that is v-unconnected for all $v \in V^{j+1}$ is called *primary*. A reduced extended form R_i^j that is v-connected for at least one $v \in V^{j+1}$ is called *secondary*. Again, R_1^4 shown in Fig. 6 is secondary. Let R_i^j be a reduced extended form such that $S_i^v \neq \emptyset$ for some $v \in V^{j+1}$ and $S_i^w = \emptyset$ for all $w \in V^l - \{v\}$, $j+1 \leq l \leq k$, then R_i^j is called *v-singular*.

Let $\mathcal{T}(G^j)$ be the set of level planar embeddings of all components of G^j. In case that G^j is level planar, the set of permutations of level-j vertices in level planar embeddings of each component F_i^j of G^j as well as its extended form H_i^j can be described by a PQ-tree $T(F_i^j)$ or $T(H_i^j)$, respectively (Jünger et al. [1998, 1999]). The 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, $\mathcal{T}(G^j)$ is a set of PQ-trees. Considering a function CHECK-LEVEL that computes for every level j, $j = 2, 3, \ldots, k$, the set $\mathcal{T}(G^j)$ of level planar embeddings of the components G^j, the algorithm LEVEL-PLANARITY-TEST can be formulated as follows.

Bool **LEVEL-PLANARITY-TEST**($G = (V^1, V^2, \ldots, V^k; E)$)
begin
  Initialize $\mathcal{T}(G^1)$;
  for $j := 1$ to $k-1$ do
    $\mathcal{T}(G^{j+1}) =$ CHECK-LEVEL($\mathcal{T}(G^j), V^{j+1}$);
    if $\mathcal{T}(G^{j+1}) = \emptyset$ then
      return "false";
  return "true";
end.

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 (see also Booth and Lueker [1976]).

PQ-trees of different components are merged in the *Second Reduction Phase* 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. Using this ordering a PQ-tree $T(F)$ is constructed that represents all possible level planar embeddings of the merged components. 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. The function REPLACE removes all leaves with a common label v after these leaves have been reduced (and therefore are consecutive in all permissible permutations) and replaces them by a single representative (Booth and Lueker [1976]). 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 every component of G^{j+1} (see Jünger et al. [1998, 1999]).

A short description of the pairwisemerge operations of Heath and Pemmaraju [1995, 1996] for non singular forms is now given. Singular components are handled by examining certain information on interior faces and regions of the outer face (see Jünger et al. [1999]). Let $G = (V, E)$ be a k-level graph and R_1^j and R_2^j be two components of G^j, $1 \leq j < k$, both being adjacent to the same vertex $v \in V^{j+1}$. Let T_1 and T_2 be the PQ-trees of R_1^j and R_2^j, both representing all level planar embeddings of their corresponding forms after the application of the first reduction phase for the level $j+1$. Identifying the vertices labeled v of the forms R_1^j and R_2^j constructs a new form $R_1^j \cup_v R_2^j$. For this new component $R_1^j \cup_v R_2^j$ a new PQ-tree T is needed that represents all level planar embeddings of $R_1^j \cup_v R_2^j$. The construction of the PQ-tree $T(R_1^j \cup_v R_2^j)$ is is based on the trees T_1 and T_2.

The merge operation is accomplished using information that is stored at the nodes of the PQ-trees. For any subset S of the set of vertices in $V^{j+1} \cup V^{j+2} \cup \cdots \cup V^k$ that belongs to a form H_i^j or R_i^j, define $\mathrm{ML}(S)$ to be the greatest $d \leq j$ such that $V^d, V^{d+1}, \ldots, V^j$ induces a subgraph in which all nodes of S occur in the same connected component. The level $\mathrm{ML}(S)$ is said to be the *meet level* of S. For a Q-node Y in the corresponding PQ-tree $T(H_i^j)$ or $T(R_i^j)$ with ordered children $Y_1, Y_2, \ldots, Y_t$ integers denoted by $\mathrm{ML}(Y_i, Y_{i+1})$, $1 \leq i < t$, are maintained satisfying $\mathrm{ML}(Y_i, Y_{i+1}) = \mathrm{ML}(\mathrm{frontier}(Y_i) \cup \mathrm{frontier}(Y_{i+1}))$. For a P-node X a single integer denoted by $\mathrm{ML}(X)$ that satisfies $\mathrm{ML}(X) = \mathrm{ML}(\mathrm{frontier}(X))$ is maintained.

Figure 7 shows the PQ-trees corresponding to the forms H_1^j and H_2^j of Fig.5 together with the ML-values that are stored at the nodes. The maintenance of the ML-values during the pattern matching algorithm REDUCE is straightforward.

For describing how to merge T_1 and T_2 corresponding to R_1^j and R_2^j we may assume without loss of generality that $\mathrm{LL}(T_1) \leq \mathrm{LL}(T_2)$. Thus the form R_2^j is the smaller form and an embedding of R_1^j has to be found such that R_2^j can be nested within

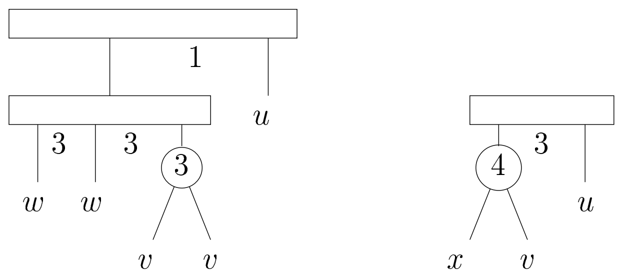


Figure 7: PQ-trees corresponding to H_1^4 and H_2^4 shown in Fig. 5.

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 v in T_1 and proceed upwards in T_1 until a node X' and its parent X are encountered satisfying one of the following five conditions.

Merge Condition A The node X is a P-node with $\mathrm{ML}(X) < \mathrm{LL}(T_2)$. Attach T_2 as child of X in T_1.

Merge Condition B The node X is a Q-node with ordered children $X_1, X_2, \ldots, X_t$, $X' = X_1$, and $\mathrm{ML}(X_1, X_2) < \mathrm{LL}(T_2)$. Replace X' in T_1 by a Q-node Y having two children, X' and the root of T_2. The case where $X' = X_t$ and $\mathrm{ML}(X_{t-1}, X_t) < \mathrm{LL}(T_2)$ is symmetric.

Merge Condition C The node X is a Q-node with ordered children $X_1, X_2, \ldots, X_t$, $X' = X_i$, $1 < i < t$, and $\mathrm{ML}(X_{i-1}, X_i) < \mathrm{LL}(T_2)$ and $\mathrm{ML}(X_i, X_{i+1}) < \mathrm{LL}(T_2)$. Replace X' by a Q-node Y with two children, X' and the root of T_2.

Merge Condition D The node X is a Q-node with ordered children $X_1, X_2, \ldots, X_t$, $X' = X_i$, $1 < i < t$, and

$$\mathrm{ML}(X_{i-1}, X_i) < \mathrm{LL}(T_2) \leq \mathrm{ML}(X_i, X_{i+1}) \ .$$

Attach the root of T_2 as child of X between X_{i-1} and X'.
In case that

$$\mathrm{ML}(X_i, X_{i+1}) < \mathrm{LL}(T_2) \leq \mathrm{ML}(X_{i-1}, X_i) \ ,$$

attach the root of T_2 as child of X between X' and X_{i+1}.

Merge Condition E The node X' is the root of T_1. Reconstruct T_1 by inserting a Q-node Y as new root of T_1 with two children X' and the root of T_2.

Based on the following lemma, we devellop a linear time algorithm for embedding a level planar graph.

Theorem 3.1 (Jünger, Leipert, and Mutzel [1998]). *The algorithm LEVEL-PLANAR-TEST tests any (not necessarily proper) level graph $G = (V, E, \phi)$ in $\mathcal{O}(n)$ time for level planarity.*

4 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 level planar embeddings 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 level planar embeddings 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 trace 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 traversing one or more levels, 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 $G = (V, E, \phi)$ be a level planar graph with leveling $\phi_G : V \to \{1, 2, \ldots, k\}$. We augment G to a planar directed acyclic st-graph $G_{st} = (V_{st}, E_{st}, \phi_{G_{st}})$ 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 $E_{st} \setminus E$, $\phi_{G_{st}}(s) = 0$, $\phi_{G_{st}}(t) = k + 1$, $(s, t) \in E_{st}$, and for all $v \in V$ we have $\phi_{G_{st}}(v) = \phi_G(v)$. This process, in which two vertices and $\mathcal{O}(n)$ edges are added to G, is the nontrivial part of the algorithm that will be explained in Sections 6 - 8.

We compute a topological sorting, i.e., an onto function $\mathrm{ts}_{G_{st}} : V_{st} \to \{1, 2 \ldots, n+2\}$. The function $\mathrm{ts}_{G_{st}}$ is comparable with $\phi_{G_{st}}$ 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 $\phi_{G_{st}}(v) \leq \phi_{G_{st}}(w)$. Obviously $\mathrm{ts}_{G_{st}}$ is an st-numbering of G_{st} (see, e.g., Even and Tarjan [1976]). 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. [1985].

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 $\phi_{G_{st}}(w)$. 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 $\mathcal{O}(n)$ time if the nontrivial part, namely the construction of G_{st} can be achieved in $\mathcal{O}(n)$ 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**$(G = (V^1, V^2, \dots, V^k; E))$
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. [1985]
        using the topological sorting as an st-numbering;
    $\mathcal{E}_l$ = CONSTRUCT-LEVEL-EMBED($\mathcal{E}_{st}, G_{st}$);
    return $\mathcal{E}_l$;
end.

Augmenting a level graph G to an st-graph G_{st} is divided into two phases. In the first phase an outgoing edge is added to every sink of G. Using the same algorithmic concept as in the first phase, an incoming edge is added to every source of G in the second phase.

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 $v \in V^j$, $j \in \{1, 2, \dots, k-1\}$, 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.

The idea of the approach can be explained best by an example. Figure 8 shows a small part of a level graph with a sink $v \in V^j$ and the corresponding part of the PQ-tree. Since v is a sink, the leaf corresponding to v will be removed from the PQ-tree before testing the graph G^{j+1} for level planarity. Instead of removing the leaf, the leaf is kept in the tree ignoring its presence from now on in the PQ-tree. Such a leaf for keeping the position of a sink v in a PQ-tree is called a *sink indicator* and denoted by $\mathrm{si}(v)$.

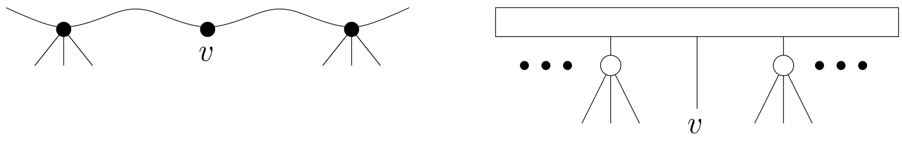


Figure 8: A sink v in a level graph G and the corresponding PQ-tree.

As shown in Fig. 9 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.

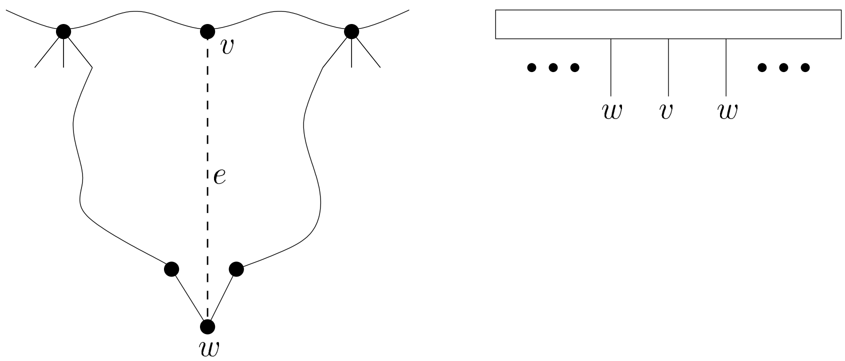


Figure 9: Adding an edge $e = (v, w)$ without destroying level planarity.

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 $\mathrm{frontier}(X)$ consists only of sink indicators. The presence of such a node is ignored in the PQ-tree 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.

5 Sink Indicators in Template Reductions

In order to achieve linear time for the level planar embedder, we have to avoid searching for sink indicators that can be considered for augmentation. Consequently, only those indicators $\mathrm{si}(v)$, $v \in V$, are considered for augmentation that appear within the pertinent subtree of a PQ-tree with respect to a vertex $w \in V$. We show that every edge added this way does not destroy level planarity. The first lemma considers sink indicators appearing within the sequence of pertinent leaves.

Lemma 5.1. *Let* $\mathrm{si}(v)$ *be a sink indicator of a vertex* $v \in V^j$*,* $1 < j < k$*, in a* PQ*-tree* T *corresponding to an extended form* H*. Adding the edge* $e = (v, w)$ *to* G *does not destroy level planarity if one of the following two conditions holds.*

(i) $\mathrm{si}(v)$ *is a descendant of a full node in the pertinent subtree with respect to a vertex* $w \in V^l$*,* $j < l \leq k$*.*

(ii) $\mathrm{si}(v)$ *is a descendant of a partial* Q*-node in the pertinent subtree with respect to a vertex* $w \in V^l$*,* $j < l \leq k$*, and* $\mathrm{si}(v)$ *appears within the pertinent sequence.*

Proof. Since $\mathrm{si}(v)$ is child of a full node or appears at least within a pertinent sequence of full nodes, adding the edge $e = (v, w)$ does not destroy level planarity of the reduced extended form R corresponding to H. Thus it remains to show that adding the edge has no effect on merge operations.

For every embedding $\mathcal{E}$ of R, the edge e is embedded either between two incoming edges of w or next to the consecutive sequence of incoming edges of w. If e is embedded between two incoming edges, the edge e obviously does not affect the level planar embedding of any nonsingular form and u-singular form with $u \neq w$.

If e is embedded next to the consecutive sequence of incoming edges of w, then $\mathrm{si}(v)$ must be a descendant of a full node X. If X is a P-node, there exists an embedding of R such that the edge e can be embedded between two incoming edges of w. Thus adding the edge does not affect the level planar embedding of any nonsingular form and any u-singular form, with $u \neq w$.

Consider now a full Q-node X. By construction, $\mathrm{si}(v)$ is a descendant leaf at one end of X. The Q-node X corresponds to a subgraph B. The vertex v must be on the boundary of the outer face of the subgraph B and there exists a path $P = (v = u_1, u_2, \ldots, u_\mu = w)$, $\mu \geq 2$, on the boundary of the outer face of B such that $\phi(u_i) < l$ for all $i = 1, 2, \ldots, \mu - 1$. Thus none of the nodes u_i, $i = 1, 2, \ldots, \mu - 1$, is considered for a merge operation. Hence, replacing the path P by an edge (v, w) at the boundary of the outer face does not affect the level planar embedding of any nonsingular form and any u-singular form, with $u \neq w$. Figure 10 illustrates the insertion of an edge $e = (v, w)$ if $\mathrm{si}(v)$ is the endmost child of a Q-node.

Considering w-singular forms, we have that adding the edge e produces one more face but the height of the largest interior face or the largest region of the outer face

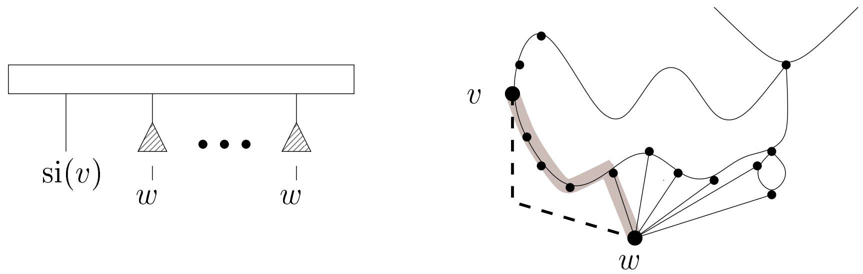


Figure 10: The sink indicator si(v) is an endmost child of a Q-node. The path P is drawn shaded, the edge $e = (v, w)$ is drawn as a dotted line.

with w being adjacent to this region remains valid. Thus a w-singular form that has to be embedded within an interior face or within a w-cavity can be embedded level planar after the insertion of e. □

Lemma 5.1 allows us to consider an edge for insertion if a sink indicator is a descendant of a full node or a descendant of a partial Q-node within the sequence of full children of the Q-node. The lemma does not consider a sink indicator si(v) that appears as a child of a partial Q-node X such that si(v) is a sibling to the pertinent sequence. Although the following lemma shows that edges corresponding to sink indicators that are endmost children at the full end of a partial Q-node can be added without destroying level planarity, the case where sink indicators are between the sequence of full and the sequence of empty children reveals problems.

Lemma 5.2. *Let* si(v) *be a sink indicator of a vertex* $v \in V^j$, $1 < j < k$, *in a* PQ*-tree* T *and let* si(v) *be a descendant of an ignored node* X *that is a child of a partial* Q*-node* Y *in the pertinent subtree with respect to a vertex* $w \in V^l$, $j < l \leq k$. *If* X *appears at the full end of the partial* Q*-node, the edge* $e = (v, w)$ *can be added without destroying level planarity.*

Proof. Analogous to the proof of Lemma 5.1 for the case in which si(v) is a descendant of a full Q-node. □

Consider now the situation of an extended reduced form R as shown in Fig. 11. The sink indicator si(v) is a child of a partial Q-node in the pertinent subtree of some vertex $w \in V^l$, $j < l \leq k$, and si(v) is adjacent to a full and an empty node. Adding the edge e does not a priori destroy level planarity in R, but it creates a new interior face, such that the large space between w and the rightmost vertex of the subgraph corresponding to the subtree rooted at X is destroyed. Now assume that a nonsingular form R' has to be w-merged into R, applying merge operation D. Although the ML-value between the leaf w and the node X allows us to add the form R' between w and X, there is, due to the insertion of e, not enough space between w and X. Hence a crossing is created and a nonlevel planar graph is constructed as is shown in Fig. 12. Consequently, a sink indicator that is found to be a sibling of a pertinent sequence and an empty sequence is never considered for edge augmentation.

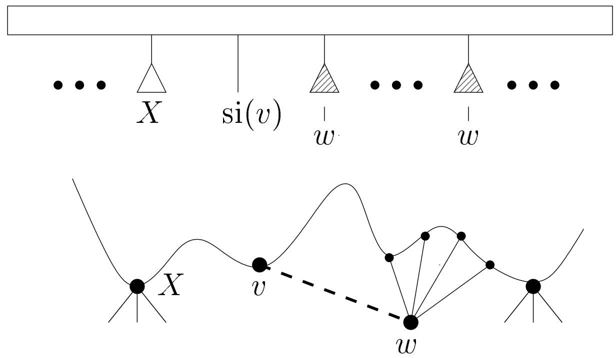


Figure 11: A doubly partial Q-node and its corresponding part of the form R. The new inserted edge $e = (v, w)$ is drawn as a dotted line.

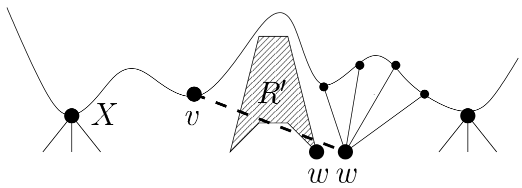


Figure 12: Merging R' into R with the new edge $e = (v, w)$ is not level planar.

By applying the results of Lemmas 5.1 and 5.2 during the template matching algorithm, 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. The following lemma allows us not only to insert edges (w, t) for every $w \in V^k$ but also to insert an edge (v, t) for every remaining sink indicator $\mathrm{si}(v)$.

Lemma 5.3. *Let* $\mathrm{si}(v)$ *be a sink indicator of a vertex* $v \in V^j$, $1 < j < k$. *If* $\mathrm{si}(v)$ *is in the final* PQ*-tree* T, *the edge* $e = (v, t)$ *can be added without destroying level planarity.*

Proof. Adding to every vertex $w \in V^k$ an edge (w, t) does not affect the level planarity of the graph. Thus consider testing the level V^{k+1} for level planarity. Obviously the pertinent subtree of T is equal to T and applying Lemma 5.1 proves Lemma 5.3. $\square$

6 Sink Indicators in Merge Operations

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 all merge operations and discuss necessary adaptions in order to treat the sink indicators correctly.

If sink indicators and ignored nodes have to be manipulated correctly during the merge process, ML-values as they have been introduced for nonignored nodes have to be introduced for ignored nodes as well. Consider a node X that becomes ignored. We make the following conventions.

(i) If X is a child of a P-node Y, the corresponding ML-value for X is $\mathrm{ML}(Y)$.

(ii) If X is a child of a Q-node, we distinguish two cases:

  (a) X does not have an adjacent ignored sibling. Let Z and Y be its direct nonignored siblings. Then we leave the values $\mathrm{ML}(Z,X)$ and $\mathrm{ML}(X,Y)$ at X, and replace according to the level planarity test the values $\mathrm{ML}(Z,X)$ and $\mathrm{ML}(X,Y)$ by a new value $\mathrm{ML}(Z,Y) = \min\{\mathrm{ML}(Z,X), \mathrm{ML}(X,Y)\}$ at Z and Y. The case where X has just one nonignored sibling is solved analogously.

  (b) X has adjacent ignored direct siblings. Let Z_I and Y_I be the direct ignored siblings and let Z and Y be its direct nonignored siblings with Z at the side where Z_I is, and Y at the side where Y_I is. Let $\mathrm{ML}(Z,X)$ and $\mathrm{ML}(X,Y)$ be the ML-values between Z and X, and X and Y, respectively. Let $\mathrm{ML}(Z_I,X)$ be the ML-value stored at Z_I, and let $\mathrm{ML}(X,Y_I)$ be the ML-value stored at Y_I. Then we replace at X the values $\mathrm{ML}(Z,X)$ by $\mathrm{ML}(Z_I,X)$ and $\mathrm{ML}(X,Y)$ by $\mathrm{ML}(X,Y_I)$, and replace according to the level planarity test the values $\mathrm{ML}(Z,X)$ and $\mathrm{ML}(X,Y)$ by a new value $\mathrm{ML}(Z,Y) = \min\{\mathrm{ML}(Z,X), \mathrm{ML}(X,Y)\}$ at Z and Y. The cases with only one nonignored or one ignored direct sibling are a handled analogously.

This strategy ensures that nonignored siblings Z and Y “know” the maximal height of the space between them, while the knowledge about the height of the space between the sinks and their corresponding indicators is left at the ignored nodes only.

Lemma 6.1. *Let X be an ignored node that is a child of a Q-node and let ML_l and ML_r be the ML-values that have been assigned to X by one of the rules (ii)(a) or (ii)(b) described above. Then the values ML_l and ML_r are valid for X.*

Proof. The sink indicators in frontier(X) can be interpreted as leaves corresponding to long edges. Thus the ML-values remain valid. □

Lemma 6.2. *Let X be an ignored node that is a child of a P-node Y. Then the value* $\mathrm{ML}(Y)$ *is valid for X.*

Proof. Analogous to the proof of Lemma 6.1. $\square$

Suppose now that two reduced forms R_1 and R_2 and their corresponding trees T_1 and T_2 with $\mathrm{LL}(T_1) \leq \mathrm{LL}(T_2)$ have to be w-merged. As described in 3, we start with the leaf labeled w in T_1 and proceed upwards in T_1 until a node X' and its parent X are encountered such that one of the five merge conditions as described in Section 3 applies. The merge operations are discussed in an order according to the difficulties that are encountered when handling involved sink indicators. Before starting with the less problematic ones, one more convention is made. If X is a node in a PQ-tree, R_X denotes the subgraph corresponding to the subtree rooted at the node X.

Merge Operation E

The tree T_1 is reconstructed by inserting a Q-node X as new root of T_1 with two children X' and the root of T_2. The following observation is trivial.

Observation 6.3. *There is no need to adapt the merge operation E in order to handle sink indicators correctly.*

Merge Operation A

The root of T_2 is attached as a child to a P-node X of T_1 thus we have that $\mathrm{ML}(X) < \mathrm{LL}(T_2)$. Obviously, all ignored nodes that are children of X are allowed to be permuted in the pertinent subtree. Thus the sink indicators in their frontier are allowed to be considered for edge augmentation. However, the ignored children can only be considered if all children of X are traversed in order to find the ignored children. This implies that all empty children of X have to be traversed as well, yielding a quadratic time algorithm. Thus ignored children of X are not considered for augmentation and we can make following observation.

Observation 6.4. *There is no need to adapt the merge operation A in order to handle sink indicators correctly.*

Merge Operation D

Let X be a Q-node of T_1 with ordered children $X_1, X_2, \ldots, X_\eta$, $\eta > 1$. Let $X' = X_\lambda$, $1 < \lambda < \eta$, and $\mathrm{ML}(X_{\lambda-1}, X_\lambda) < \mathrm{LL}(T_2) \leq \mathrm{ML}(X_\lambda, X_{\lambda+1})$. Thus R_2 has to be nested between the subgraphs $R_{X_{\lambda-1}}$ and R_{X_λ} and the root of T_2 is attached as a child to the Q-node X between $X_{\lambda-1}$ and X_λ.

Let $I_1, I_2, \ldots, I_\mu$, $\mu \geq 0$, be the sequence of ignored nodes between $X_{\lambda-1}$ and X_λ with $X_{\lambda-1}$ and I_1 being direct siblings, and X_λ and I_μ being direct siblings. As illustrated in Fig. 13 there may exist a $\nu \in \{1, 2, \ldots, \mu\}$ such that for every sink indicator

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

the graph has to be augmented by an edge $e = (v, w)$. Adding these edges does not destroy level planarity. Furthermore, augmenting the graph G for every $\mathrm{si}(v) \in \bigcup_{i=\nu}^{\mu} \mathrm{frontier}(I_i)$, with $\phi(v) \geq \mathrm{LL}(T_2)$, by an edge $e' = (v, u)$, $u \in \bigcup_{i=\phi(w)}^{k} V^i$, $u \neq w$ destroys level planarity.

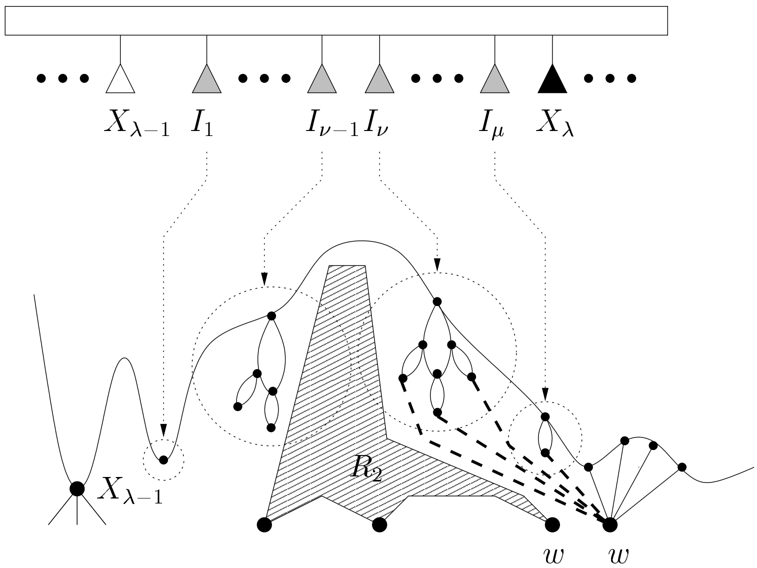


Figure 13: Merging the form R_2 into R_1 using the merge operation D forces us to augment G by the edges drawn as dotted lines.

Using the following lemma we are able to find all the sink indicators that have to be considered for edge insertion when applying the merge operation D.

Lemma 6.5. *Let X be a child of a Q-node and let Y be a direct nonignored sibling of X. Let $I_1, I_2, \ldots, I_\mu$, $\mu \geq 0$, be the sequence of ignored nodes between X and Y with X and I_1 being direct siblings, and Y and I_μ being direct siblings. There exists a $\nu \in \{1, 2, \ldots, \mu + 1\}$ such that $\mathrm{ML}(X, Y) = \mathrm{ML}(I_{\nu-1}, I_\nu)$, with $I_0 = X$ and $I_{\mu+1} = Y$.*

Proof. The lemma follows immediately from Lemma 6.1. □

Remark 6.6. *We store at every pair of direct nonignored siblings X and Y pointers to the two ignored siblings $I_{\nu-1}$ and I_ν with $\mathrm{ML}(I_{\nu-1}, I_\nu) = \mathrm{ML}(X, Y)$. The maintenance during the application of template reduction algorithm and the merge* *operations is straightforward. We will see later how we benefit from this in the merge operations B and C.*

Placing the root of T_2 between $I_{\nu-1}$ and I_ν conctructs a PQ-tree such that the ignored nodes $I_\nu, I_{\nu+1}, \ldots, I_\mu$ appeare within the pertinent subtree. This allows to augment the graph G by an edge $e = (v, w)$ for every sink indicator $\mathrm{si}(v) \in \bigcup_{i=\nu}^{\mu} \mathrm{frontier}(I_i)$ during the reduction with respect to w.

Merge Operation C

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

As illustrated in Fig. 14 there may exist a ν, $1 \leq \nu \leq \mu$, such that for every sink indicator

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

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}.

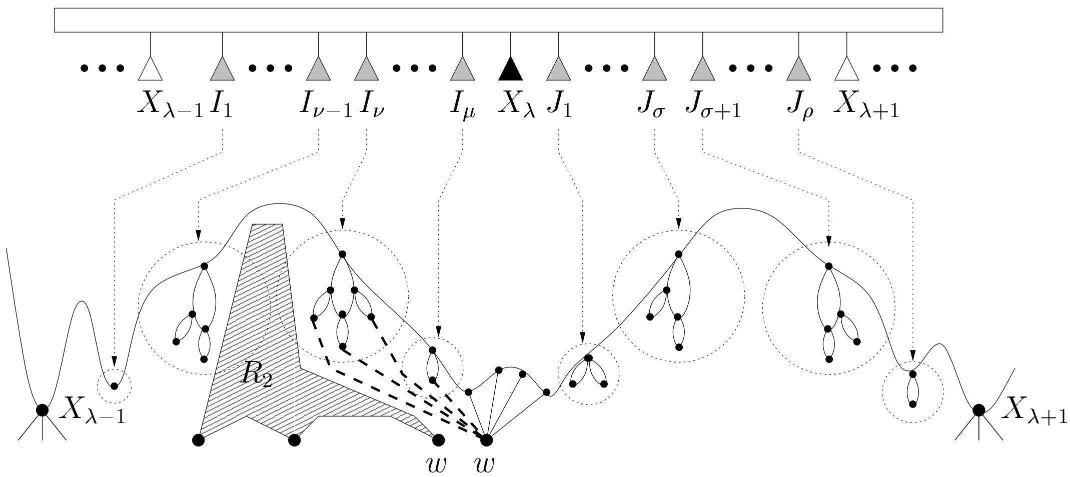


Figure 14: Merging the form R_2 into R_1 using the merge operation C and embedding it between $R_{X_{\lambda-1}}$ and R_{X_λ} forces G to be augmented by the edges drawn as dotted lines.

As is illustrated in Fig. 15, R_2 can 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

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

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

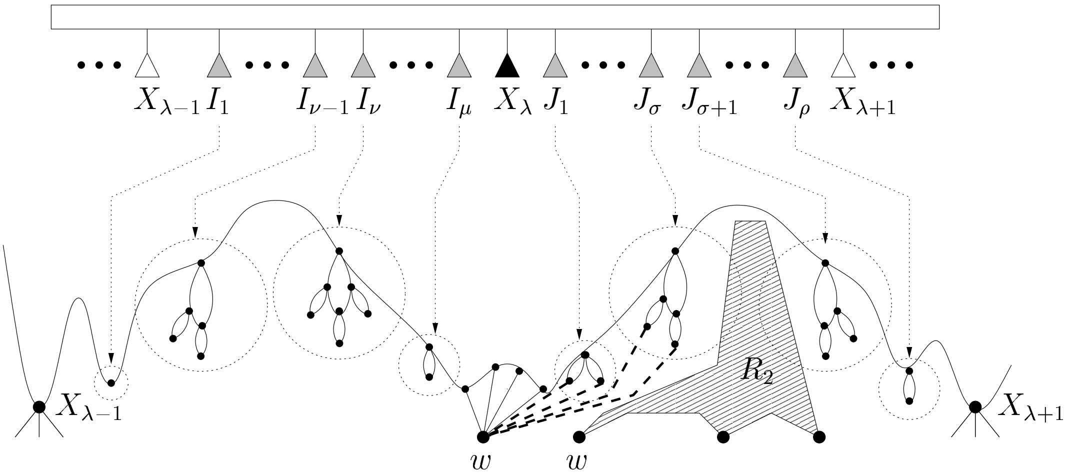


Figure 15: Merging the form R_2 into R_1 using the merge operation C and embedding it between R_{X_λ} and $R_{X_{\lambda+1}}$ forces G to be augmented by the edges drawn as dotted lines.

It is not possible to consider both sets of ignored nodes for edge augmentation. Consider for instance the example shown in Fig. 16, where edges for both sets $\bigcup_{i=\nu}^{\mu} \mathrm{frontier}(I_i)$ and $\bigcup_{i=1}^{\sigma} \mathrm{frontier}(J_i)$ have been added, yielding immediately a non-level planar graph.

However, deciding which set of sink indicators has to be considered for edge augmentation is not possible unless X_λ is a full node (see Leipert [1998]). Proceeding the level planarity test down the levels $V^{\phi(w)+1}$ 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 to recognize 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 sequence* 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 $\mathrm{rseq}(R_2)^{\mathit{left}}$, and to $J_1, J_2, \ldots, J_\sigma$ as the *right reference sequence* denoted by $\mathrm{rseq}(R_2)^{\mathit{right}}$. 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* $\mathrm{ref}(R_2)^{\mathit{left}}$ and $\mathrm{ref}(R_2)^{\mathit{right}}$, respectively, are defined analogously to the left and right reference sequence.

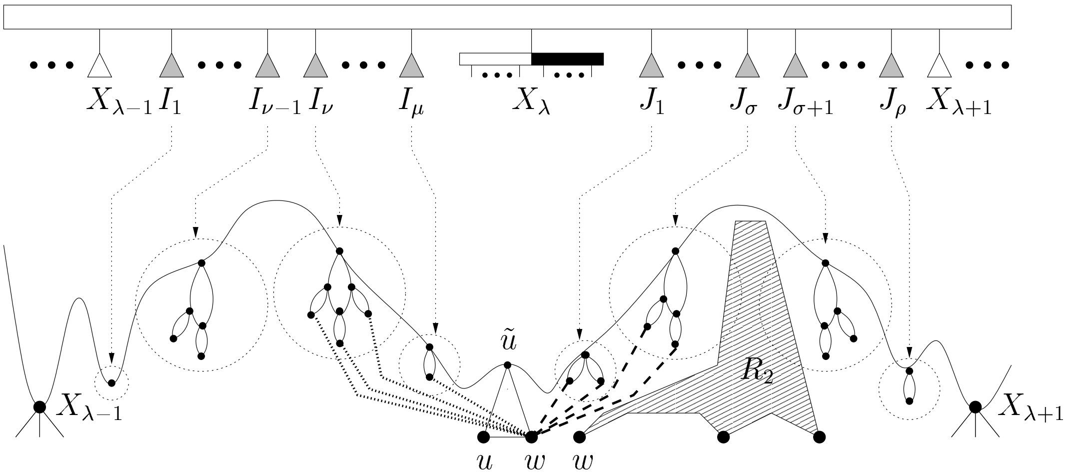


Figure 16: Merging the form R_2 into R_1 using the merge operation C does not allow to consider sinks on both sides of R_{X_λ} for edge augmentation. Independently on the chosen embedding of R_2 there are always crossings between a path connecting $\tilde{u}$ and u and the new edges.

In Section 7 a method using a special ignored indicator is developed for deciding which subset of ref(R_2) has to be considered for edge augmentation. Before continuing with the algorithmic solution, we finish by considering the merge operation B where exactly the same problem occurs as has been encountered for the merge operation C.

Merge Operation B

Let X be a Q-node with ordered children $X_1, X_2, \ldots, X_\eta$, and let $X' = X_1$, and $\mathrm{ML}(X_1, X_2) < \mathrm{LL}(T_2)$. The node X_1 is replaced by a Q-node Y having two children, X_1 and the root of T_2.

Let $I_1, I_2, \ldots, I_\mu$, $\mu \geq 0$, be the sequence of ignored nodes at one end of X with X_1 and I_μ being direct siblings and I_1 being an endmost child of X. Let $J_1, J_2, \ldots, J_\rho$, $\rho \geq 0$, be the sequence of ignored nodes between X_1 and X_2 with X_1 and J_1 being direct siblings, and X_2 and J_ρ being direct siblings.

Analogously to the merge operation C, there may exist sink indicators affected by merging R_2 into R_1 in both sets $I_1, I_2, \ldots, I_\mu$, $\mu \geq 0$, and $J_1, J_2, \ldots, J_\rho$, $\rho \geq 0$. Again it is not possible to decide if the left reference set ref$(R_2)^{left}$ or the right reference set ref$(R_2)^{right}$ has to be considered for edge augmentation.

7 Contacts

In order to solve the decision problem of the merge operations B and C, we examine how R_2 is fixed to either side of the vertex $w \in V$ in a level planar embedding of G. For the rest of this subsection we consider two PQ-trees T_1 and T_2, such that T_2 has been w-merged into T_1 using a merge operation B or C. Let X be the Q-node with children $X_1, X_2, \ldots, X_\eta$, $\eta \geq 2$, and let X_λ, $\lambda \in \{1, 2, \ldots, \eta\}$, be its child that is replaced by a new Q-node having two children X_λ and the root of T_2. Let R_{X_λ} be the subgraph of R_1 corresponding to the subtree rooted at X_λ before merging R_1 and R_2. Let R_X be the subgraph corresponding to the subtree rooted at X before merging R_1 and R_2.

Definition 7.1. *Define $\vec{R}_X$ to be the set of all vertices $u \in V$ such that there exists a vertex $v \in R_X$ and a (not necessarily directed) path P connecting u and v not using the connective cut vertex of X. Define further $D(R_{X_\lambda} \cup R_2) \subset \bigcup_{i=\phi(w)}^{k} V^i$ to be the set of vertices $u \in \bigcup_{i=\phi(w)}^{k} V^i$ such that the following two conditions hold.*

1. *There exists a directed path $P = (u_1, u_2, \ldots, u_\xi = u)$, $\xi > 1$, with $u_1 \in R_{X_\lambda} \cup R_2$.*

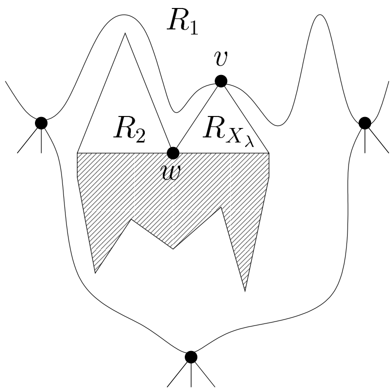
2. *There exists a vertex $\tilde{u} \in \bigcup_{i=\phi(w)}^{k} V^i$ and a directed path $\tilde{P} = (\tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_\iota = \tilde{u})$, $\iota > 1$, with $\tilde{u}_1 \in R_{X_\lambda} \cup R_2$, such that $\phi(\tilde{u}) \geq \phi(u)$ and the paths P and $\tilde{P}$ are vertex disjoint except for possibly u and $\tilde{u}$.*

The vertex set $D(R_{X_\lambda} \cup R_2)$ is called dependent set *of $R_{X_\lambda} \cup R_2$.*

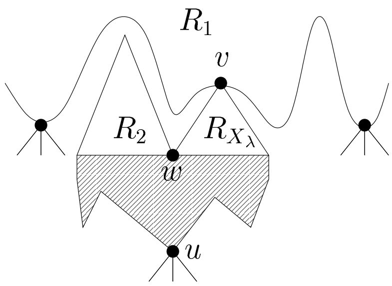
Figure 17 illustrates different kinds of dependent sets $D(R_{X_\lambda} \cup R_2)$. The dependent set $D(R_{X_\lambda} \cup R_2)$ is drawn shaded in all four cases. The vertex v in all four subfigures denotes the connective cut vertex of X_λ in $G^{\phi(w)}$ that allows to reverse the subgraph $R_{X_\lambda} \cup R_2$ with respect to R_X.

For simplicity, we make the overall assumption for the rest of this section that no vertex $u \in D(R_{X_\lambda} \cup R_2)$ is involved in a merge operation. This matter is discussed in the next section, handling concatenations of merge operations. However, subsequent merge operations to any other vertex not contained in $D(R_{X_\lambda} \cup R_2)$ are allowed after w-merging R_2 into R_1.

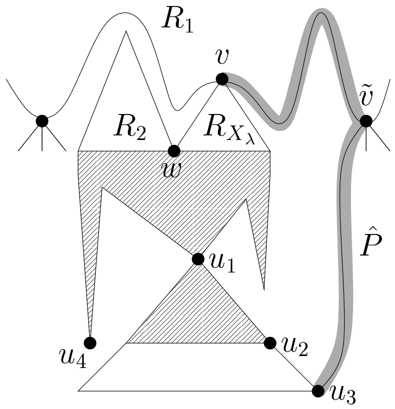
Figure 17(a) illustrates the case, where v is not only a cut vertex in $G^{\phi(w)}$ but a also a cut vertex in the graph G. Consequently, $R_{X_\lambda} \cup R_2 \cup D(R_{X_\lambda} \cup R_2)$ will be embedded within an interior face or the outer face with the option to chose its embedding unaffected from the embedding of the rest of the graph. Hence R_2 may be embedded on an arbitrary side of R_{X_λ} with respect to R_X.



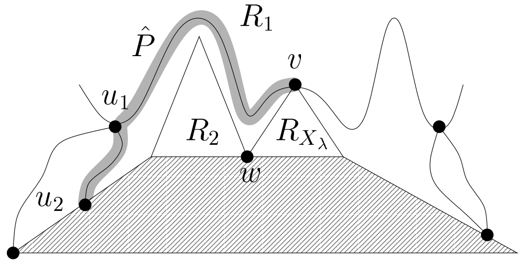
(a) Cut vertex v allows free embedding of $R_{X_\lambda} \cup R_2 \cup D(R_{X_\lambda} \cup R_2)$.



(b) Split pair u, v allows a free embedding of $R_{X_\lambda} \cup R_2 \cup D(R_{X_\lambda} \cup R_2)$.



(c) Split pair u, v allows a free embedding of R_2.



(d) Fixed embedding of $R_{X_\lambda} \cup R_2 \cup D(R_{X_\lambda} \cup R_2)$.

Figure 17: The figure illustrates different dependent sets $D(R_{X_\lambda} \cup R_2)$. The dependent sets are drawn shaded and path $\hat{P}$ is drawn grey.

Figure 17(b) illustrates the case, where v is not a cut vertex in the graph G but there exists a vertex $u \in D(R_{X_\lambda} \cup R_2)$ such that u and v form a split pair and we have that

$$\phi(\tilde{u}) < \phi(u) \text{ if } \tilde{u} \in D(R_{X_\lambda} \cup R_2) - \{u\} \ .$$

Thus $R_{X_\lambda} \cup R_2 \cup D(R_{X_\lambda} \cup R_2)$ forms a split component and its embedding may be chosen freely. Hence R_2 may be embedded on an arbitrary side of R_{X_λ} with respect to R_X.

Figure 17(c) illustrates a more delicate situation involving a split pair v and u_1. According to the definition of the dependent set, the vertex u_2 is contained in $D(R_{X_\lambda} \cup R_2)$ since there exists a vertex u_4 with $\phi(u_2) = \phi(u_4)$ and two directed paths P and $\tilde{P}$, with

(i) P connecting a vertex of $R_{X_\lambda} \cup R_2$ and u_2,

(ii) $\tilde{P}$ connecting a vertex of $R_{X_\lambda} \cup R_2$ and u_4, and

(ii) P and $\tilde{P}$ being disjoint.

Although $u_2 \in D(R_{X_\lambda} \cup R_2)$, the vertex u_2 is not contained in the split component of v and u_1. The vertex u_3, however, does not belong to the dependent set $D(R_{X_\lambda} \cup R_2)$ since any directed path connecting a vertex of $R_{X_\lambda} \cup R_2$ and a vertex in $\bigcup_{i=\phi(u_3)}^{k} V^i$ must contain the vertex u_1. Hence, the paths are not disjoint and we have that $u_3 \notin D(R_{X_\lambda} \cup R_2)$. Figure 17(c) shows a (not necessarily directed) path $\hat{P}$ connecting v and u_3 via $\tilde{v}$, such that $\hat{P}$ and $R_{X_\lambda} \cup R_2 \cup D(R_{X_\lambda} \cup R_2)$ are disjoint. This leads to the interesting situation that u_3 and therefore u_2 are fixed in their embedding to the side where $\tilde{v}$ is, while we are still able to flip the split component of u_1 and v around, choosing an arbitrary side where to embed R_2 next to R_{X_λ} with respect to R_X.

However, the existence of a split component does not guarantee a free choice of the embedding. In case that a (not necessarily directed) path $\tilde{P}$ exists, connecting the vertices v and u_2 via $\tilde{v}$ such that the path $\tilde{P}$ and $R_{X_\lambda} \cup R_2 \cup D(R_{X_\lambda} \cup R_2)$ are disjoint, and the path $\tilde{P}$ uses only vertices in $\bigcup_{i=1}^{\phi(u_2)} V^i$, we cannot flip the split component of v and u_1 anymore.

While Fig. 17(a),(b),(c) describe examples of dependent sets such that an embedding of R_2 can be chosen freely, Fig. 17(d) gives an example of a dependent set that has to be embedded such that R_2 is forced to be embedded on exactly one side of R_{X_λ} with respect to R_X. Consider a vertex $u_1 \in V - (R_{X_\lambda} \cup R_2 \cup D(R_{X_\lambda} \cup R_2))$ and a vertex $u_2 \in D(R_{X_\lambda} \cup R_2)$ such that there exists path $\hat{P}$ disjoint to $R_{X_\lambda} \cup R_2 \cup D(R_{X_\lambda} \cup R_2)$, connecting v and u_2 via u_1, and the path $\hat{P}$ uses only vertices in $\bigcup_{i=1}^{\phi(u_2)} V^i$. If there exists a vertex $u_3 \in D(R_{X_\lambda} \cup R_2)$, $u_3 \neq u_2$, with $\phi(u_3) \geq \phi(u_2)$, the path $\hat{P}$ forces R_2 to be embedded on one side of R_{X_λ} with respect to R_X.

Figure 17 implicitly assumes that the Q-node X remains a node with at least two nonignored children, one being the Q-node Y (the node that has been introduced when merging $T(R_1)$ and $T(R_2)$). The example of Fig. 18 shows a subgraph corresponding to the subtree rooted at X, where X has become a Q-node with only one nonignored child that is the node Y. Thus, there exists a split pair v and $\tilde{v}$ in G with $\tilde{v}$ being the connective cut vertex of R_X that allows reversing the split component containing $\vec{R}_X - (R_{X_\lambda} \cup R_2 \cup D(R_{X_\lambda} \cup R_2))$. This implies that R_2 may be embedded on either side of R_{X_λ} with respect to R_X. We note that a path $P = (v = u_1, u_2, \ldots, u_\mu = u)$, $\mu \geq 2$, may exist, connecting v and a vertex $u \in D(R_{X_\lambda} \cup R_2)$ such that P is disjoint to $D(R_{X_\lambda} \cup R_2)$, and the path P uses only vertices in $\bigcup_{i=1}^{\phi(u)} V^i$. Such a path has no effect on the embedding of R_2 next to R_{X_λ} with respect to R_X since P must traverse the connective cut vertex $\tilde{v}$ of R_X. Figure 18 shows the path P as a dotted line.

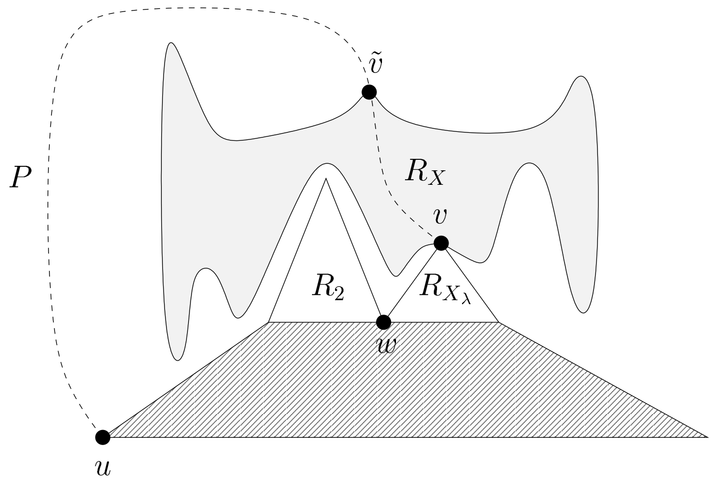


Figure 18: A graph corresponding to the situation where X became a Q-node with one nonignored child. The embedding of $R_{X_\lambda} \cup R_2 \cup D(R_{X_\lambda} \cup R_2)$ with respect to R_X may be chosen freely.

However, if there exists a vertex $\tilde{u} \in \vec{R}_X - (R_{X_\lambda} \cup R_2 \cup D(R_{X_\lambda} \cup R_2))$ such that for every vertex $u_i \in P$ the inequality $\phi(u_i) \leq \phi(\tilde{u})$ holds, the embedding of R_2 is fixed next to R_{X_λ} with respect to R_X.

Our discussion leads to the following observations.

Observation 7.2. *Let v be the connective cut vertex of R_{X_λ} and let $\tilde{v}$ be the connective cut vertex of R_X if X has a parent. The subgraph R_2 is not fixed to any side of R_{X_λ} with respect to R_X if and only if for every vertex u in the dependent set $D(R_{X_\lambda} \cup R_2)$ and every undirected path $P = (v = u_1, u_2, \ldots, u_\mu = u)$, $\mu \geq 2$, with $u_i \in \bigcup_{i=1}^{\phi(u)} V^i$ for all $i = 1, 2, \ldots, \mu$, one of the following conditions holds.*

(i) $u_{\mu-1} \in R_{X_\lambda} \cup R_2 \cup D(R_{X_\lambda} \cup R_2)$.

(ii) $\tilde{v} \in P$ *and for all* $v' \in \vec{R}_X - (R_{X_\lambda} \cup R_2 \cup D(R_{X_\lambda} \cup R_2))$ *the inequality* $\phi(v') < \phi(u)$ *holds.*

(iii) v *and* u *form a split pair in* G *and for all* $v' \in D(R_{X_\lambda} \cup R_2) - \{u\}$ *the inequality* $\phi(v') < \phi(u)$ *holds.*

Observation 7.3. *Let v be the connective cut vertex of R_{X_λ} and let $\tilde{v}$ be the connective cut vertex of R_X if X has a parent. The subgraph R_2 is fixed to a side of*

R_{X_λ} with respect to R_X if and only if there exists a vertex u in the dependent set $D(R_{X_\lambda} \cup R_2)$ and an undirected path $P = (v = u_1, u_2, \ldots, u_\mu = u)$, $\mu \geq 2$, with $u_i \in \bigcup_{i=1}^{\phi(u)} V^i$ for all $i = 1, 2, \ldots, \mu$, and all of the following three conditions hold.

(i) $u_{\mu-1} \notin R_{X_\lambda} \cup R_2 \cup D(R_{X_\lambda} \cup R_2)$.

(ii) *(a)* $\tilde{v} \notin P$, *or*

  (b) $\tilde{v} \in P$ *and there exists a* $v' \in \vec{R}_X - (R_{X_\lambda} \cup R_2 \cup D(R_{X_\lambda} \cup R_2))$ *such that* $\phi(v') \geq \phi(u)$.

(iii) There exists a vertex $v' \in D(R_{X_\lambda} \cup R_2) - \{u\}$ *such that the inequality* $\phi(v') \geq \phi(u)$ *holds.*

The path P connecting the vertex v and a vertex $u \in D(R_{X_\lambda} \cup R_2)$ uses only level-i vertices with $i \leq \phi(u)$. This implies that the last edge $(u_{\mu-1}, u)$ on the path P must be an incoming edge of u. We use this fact to determine to which side of R_{X_λ} the form R_2 is fixed with respect to R_X. During the reduction of the leaves corresponding to the vertex u we analyze the incoming edges of u, determining for each edge if it is the last edge of a path that is treated in one of the Observations 7.2 and 7.3. The following two lemmas help us to perform the case distinction in a very efficient way. We note that the parent of Y (Y is the Q-node that has been inserted by the merge operation) does not need to be the node X throughout the algorithm, e.g., it may have been removed from the PQ-tree when applying a reduction using one of the templates Q2 and Q3.

Lemma 7.4. *The subgraph R_2 has to 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.*

Proof. Let R_2 be fixed to a side of R_{X_λ} with respect to R_X. According to Observation 7.3, there exists a vertex u in the dependent set $D(R_{X_\lambda} \cup R_2)$ and an undirected path $P = (v = u_1, u_2, \ldots, u_\mu = u)$, $\mu \geq 2$, with $u_i \in \bigcup_{i=1}^{\phi(u)} V^i$ for all $i = 1, 2, \ldots, \mu$. The last edge $e = (u_{\mu-1}, u)$ on P is therefore an incoming edge of u, and $u_{\mu-1} \notin R_{X_\lambda} \cup R_2 \cup D(R_{X_\lambda} \cup R_2)$. Since u is in $D(R_{X_\lambda} \cup R_2)$, it must have a second incoming edge $\tilde{e}$, with $\tilde{e}$ being incident to a vertex $\tilde{u} \in \bigcup_{i=1}^{\phi(u)-1} V^i \cap (R_{X_\lambda} \cup R_2 \cup D(R_{X_\lambda} \cup R_2))$. Thus for the leaf $\tilde{l}$ in T corresponding to $\tilde{e}$ it follows that $\tilde{l} \in \text{frontier}(Y)$. Furthermore, the condition 7.3(i) guarantees that for the leaf l in T corresponding to e we have $l \notin \text{frontier}(Y)$.

Let Z be the smallest common ancestor of l and $\tilde{l}$ in the PQ-tree. Obviously, the Q-node Y is a descendant of Z and we have $Y \neq Z$.

Let $\tilde{X}$ be the parent of Y. If condition 7.3(ii)(a) holds, then $l \in \text{frontier}(\tilde{X})$, and $\tilde{v}$ (the connective cut vertex of R_X) and v (the connective cut vertex of R_{X_λ}) do not form a split pair in G. Thus the parent of Y did not become a node with Y as its only nonignored child.

If on the other hand condition 7.3(ii)(b) holds, then $l \notin \text{frontier}(\tilde{X})$, but there exists at least one empty child of $\tilde{X}$ containing a leaf in its frontier corresponding to a vertex $v' \in \vec{R}_X - (R_{X_\lambda} \cup R_2 \cup D(R_{X_\lambda} \cup R_2))$. Thus again, the parent of Y did not become a node with Y as the only nonignored child.

The node Y was a child of the Q-node X when it was introduced into the PQ-tree. Since the parent of Y did not become a node with Y as the only nonignored child, we have according to Lemma 2.2 that Y remains a child of a Q-node throughout the applications of the template matching algorithm. Due to the overall assumption that no vertex in $D(R_{X_\lambda} \cup R_2)$ is involved in another merge operation, Y remains a child of a Q-node throughout every merge operation.

Due to condition 7.3(iii) there exists an empty leaf in the frontier of the node Y. Thus Y is a partial node, and Y and its parent $\tilde{X}$ are traversed during the reduction with respect to the vertex u. Since $\tilde{X}$ is a Q-node that is contained in the pertinent subtree with respect to u, either template Q2 or template Q3 is applied to Y and $\tilde{X}$, removing Y from the PQ-tree.

Now let Y be removed from the tree during the reduction with respect to some vertex u by applying template Q2 or Q3 and and let the parent of Y never become a node with Y being its only nonignored child.

Since the parent of Y always has at least two children, condition 7.3(ii)(a) or (ii)(b) must hold. Furthermore, the application of template Q2 or Q3 implies that the template matching algorithm has traversed Y and its parent, which is a Q-node as well. Hence the root of the pertinent subtree must be a proper ancestor of Y. Thus there exists a pertinent leaf l not in the subtree of Y, and a path $P = (v = u_1, u_2, \ldots, u_\mu = u)$, $\mu \geq 2$, with $u_i \in \bigcup_{i=1}^{\phi(u)} V^i$ for all $i = 1, 2, \ldots, \mu$, such that $u_{\mu-1} \notin R_{X_\lambda} \cup R_2 \cup D(R_{X_\lambda} \cup R_2)$. Since one of the templates Q2 and Q3 has been applied in order to remove Y from the tree, Y itself must have been partial, and therefore must have had at least one empty leaf in its frontier. Thus condition 7.3(iii) holds. It follows that R_2 is fixed on one side of R_{X_λ} with respect to R_X. □

Lemma 7.5. *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 cases occurs during the application of the template matching algorithm.*

(i) The Q-node Y gets ignored.

(ii) The Q-node Y is a nonignored node of the final PQ-tree.

(iii) The Q-node Y has only one nonignored child.

(iv) The parent of Y has only Y as a nonignored child.

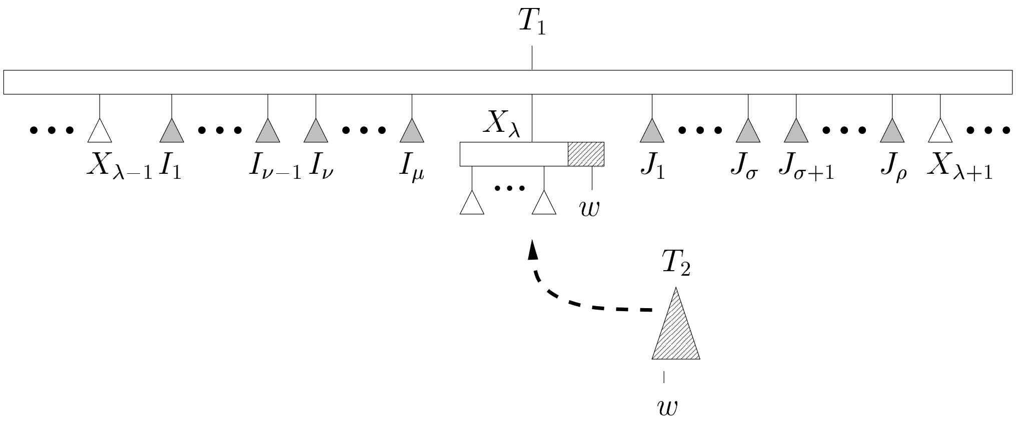
Proof. Let R_2 be a subgraph not fixed to any side of R_{X_λ}. According to Observation 7.2 the cases 7.2(i), 7.2(ii), or 7.2(iii) apply. If there exists a path P and a vertex $u \in D(R_{X_\lambda} \cup R_2)$ in G that satisfy condition 7.2(ii), it follows that the Q-node X was transformed into a node with only one nonignored child and possibly some ignored children. Then the case (iv) follows immediately. If there exists a vertex $u \in D(R_{X_\lambda} \cup R_2)$ that satisfies 7.2(iii) then there exists a level l, $\phi(w) < l \leq k$, (w being the vertex involved in merging R_{X_λ} and R_2) such that $D(R_{X_\lambda} \cup R_2) \cap \bigcap_{i=l}^{k} V^i = \emptyset$ and $|D(R_{X_\lambda} \cup R_2) \cap V^{l-1}| = 1$. Thus after completing the level planarity test for G^{l-1} the node Y is a Q-node with just one nonignored child.

Now assume that 7.2(i) holds for all paths in G connecting v and a vertex $u \in D(R_{X_\lambda} \cup R_2)$ and no path matches condition 7.2(ii) and 7.2(iii). It follows from 7.2(i) that 7.3(i) does not hold for any vertex $u \in D(R_{X_\lambda} \cup R_2)$. According to Lemma 7.4, the Q-node Y is not removed from the tree using one of the templates Q2 and Q3, and one of the following two cases must hold.

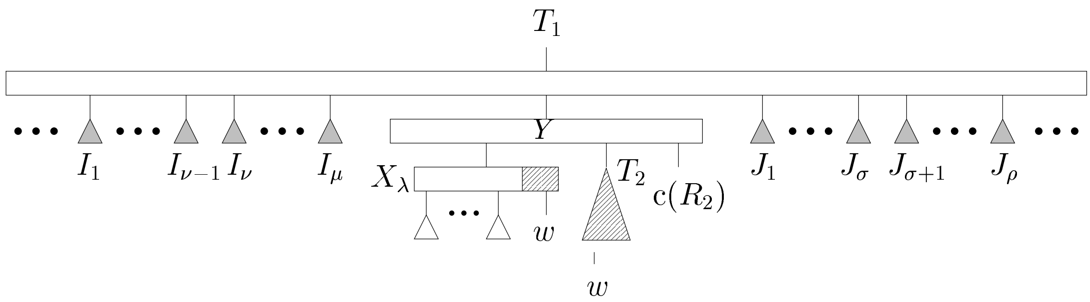
1. There exists a level l, $\phi(w) < l \leq k$, such that $D(R_{X_\lambda} \cup R_2) \cap \bigcap_{i=l}^{k} V^i = \emptyset$ and $|D(R_{X_\lambda} \cup R_2) \cap V^{l-1}| \geq 1$. Thus after completing the level planarity test for G^{l-1} the node Y is a Q-node with nonignored children. Two subcases occur
   (a) Every leaf in the frontier of the nonignored children of Y is replaced by a sink indicator before testing G^l for level planarity. It follows that case (i) applies.
   (b) All leaves in the frontier of Y except for the leaves in the frontier of one child of Y become ignored. Thus case (iii) applies
2. The node Y is found in the final PQ-tree.

In reversion, if one of the four cases applies to the Q-node Y, we have by Observation 7.2 that any embedding may be chosen. ☐

Lemmas 7.4 and 7.5 reveal a solution for solving the problem of deciding whether 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 clearly can 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 has to 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 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 $\mathrm{c}(R_2)$. It is placed as endmost child of Y during the merge operation B or C next to the root of T_2. Thus the Q-node Y has now three children instead of two. See Fig. 19 for an illustration.



(a) $\mathrm{ML}(X_{\lambda-1}, X_\lambda) < \mathrm{LL}(T_2)$ and $\mathrm{ML}(X_\lambda, X_{\lambda+1}) < \mathrm{LL}(T_2)$.



(b) Contact $\mathrm{c}(R_2)$ is added as a child to Y next to the root of T_2.

Figure 19: Adding a contact during the merge operation C.

Since the contact $\mathrm{c}(R_2)$ is related to a w-merge operation, the vertex w is called *related vertex* of $\mathrm{c}(R_2)$ and denoted by $\omega(\mathrm{c}(R_2))$. The corresponding w-merge operation is said to be *associated* with $\mathrm{c}(R_2)$. 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 7.6. *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.*

Proof. The ignored nodes of $\mathrm{rseq}(R_2)^{left}$ and $\mathrm{rseq}(R_2)^{right}$ are children of a Q-node, and therefore remain children of a Q-node keeping their order throughout the application of the template matching algorithm, unless either $\mathrm{rseq}(R_2)^{left}$ or $\mathrm{rseq}(R_2)^{right}$ are found to be within a pertinent sequence. However, this can only happen if the node Y becomes pertinent, provided that the node Y does not become ignored itself. □

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 an object of another merge operation B or C. Concatenation of contacts is discussed in the next subsection.

Observation 7.7. *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 applying one of the templates Q2 or Q3.*

Observation 7.8. *If the node Y is eliminated applying the templates Q2 or Q3, the contact* $\mathrm{c}(R_2)$ *determines the side were R_2 has to 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 $i \in \{\mathit{left}, \mathit{right}\}$ and* $\mathrm{ref}(R_2)^i$ *has to be considered for edge augmentation.*

Besides placing $\mathrm{c}(R_2)$ as 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. This is necessary, since direct nonignored siblings of Y may become ignored throughout the application of the algorithm. Let $\mathrm{rseq}(R_2)^{\mathit{left}} = \{I_\nu, I_{\nu+1}, \ldots, I_\mu\}$ be the left reference sequence and let $\mathrm{rseq}(R_2)^{\mathit{right}} = \{J_1, J_2, \ldots, J_\sigma\}$ be the right reference sequence. After performing a reduction applying template 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 20 illustrates this strategy for the latter case. 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 $\mathrm{ref}(R_2)$. The nodes $I_\nu, I_\mu, J_1, J_\sigma$ are called the *reference points* of the contact $\mathrm{c}(R_2)$. Analogously to the definition of a reference set of R_2, $\mathrm{ref}(R_2)$ is said to be the reference set of $\mathrm{c}(R_2)$ and denoted by $\mathrm{ref}(\mathrm{c}(R_2))$.

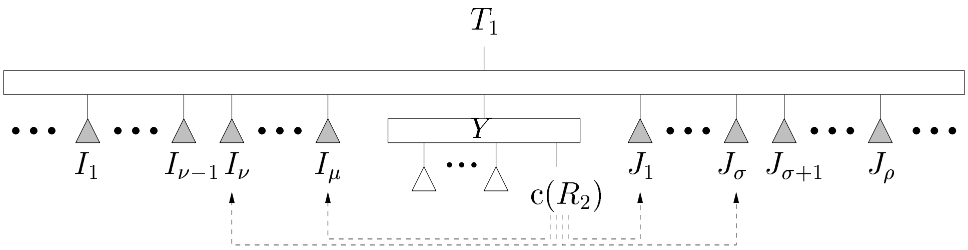
The section closes with a summary of the results.

Lemma 7.9. *Let* $\mathrm{c}(R_2)$ *be a contact related to a vertex w and let* $\mathrm{ref}(R_2)^{\mathit{left}}$ *be the left reference set of* $\mathrm{c}(R_2)$ *with reference points I_ν, I_μ and* $\mathrm{ref}(R_2)^{\mathit{right}}$ *be the right reference set of* $\mathrm{c}(R_2)$ *with reference points J_1, J_σ. Then the following statements are true.*

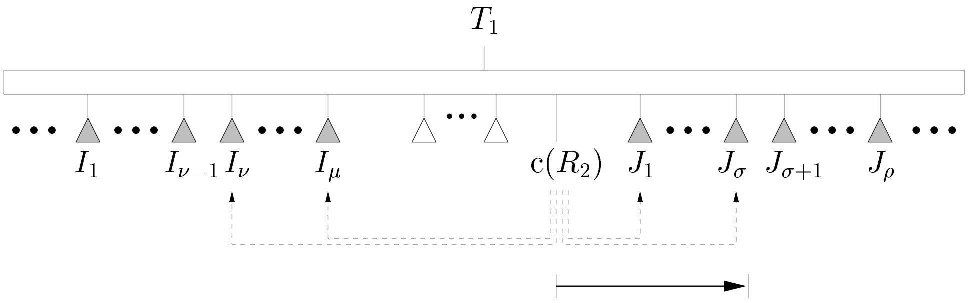
(i) If $\mathrm{c}(R_2)$ *is adjacent to I_μ, then augmenting G_{st} by an edge (u, w) for every* $\mathrm{si}(u) \in \mathrm{ref}(R_2)^{\mathit{left}}$ *does not destroy level planarity.*

(ii) If $\mathrm{c}(R_2)$ *is adjacent to J_1, then augmenting G_{st} by an edge (u, w) for every* $\mathrm{si}(u) \in \mathrm{ref}(R_2)^{\mathit{right}}$ *does not destroy level planarity.*

Proof. The lemma immediately follows from Lemmas 7.4 and 7.6. ∎



(a) The node Y with the contact $\mathrm{c}(R_2)$ before the application of a template Q2 or Q3.



(b) The contact $\mathrm{c}(R_2)$ is adjacent to the ignored node J_1. We chose $\mathrm{ref}(R_2)^{right}$ for augmentation.

Figure 20: The identification of the reference set that has to be chosen for augmentation. The dotted lines denote the pointers of $\mathrm{c}(R_2)$ to its reference points.

8 Concatenation of Contacts

For clarity, the previous section omitted the concatenation of merge operations applied to the vertices of the dependent set corresponding to a merge operation B or C. This section deals with the subject of concatenating merge operations.

Let R_1 be a reduced extended form that has been w_1-merged into a reduced extended form R applying a merge operation B or C. Let T and T_1 be the PQ-trees corresponding to R and R_1. Let X be the Q-node with children $X_1, X_2, \ldots, X_\eta$, $\eta \geq 2$, and let X_λ, $\lambda \in \{1, 2, \ldots, \eta\}$, be the child that is replaced by a new Q-node having two children X_λ and the root of T_1. Let R_i, $i = 2, 3, \ldots, \mu$, be reduced extended forms where every R_i has to be w_i-merged into R, and R_i is w_i-merged into R before R_{i+1} is w_{i+1}-merged into R, for all $i = 2, 3, \ldots, \mu - 1$.

Definition 8.1. *Let $D(R_{X_\lambda} \cup R_1) \subset \bigcup_{\nu=\phi(w_1)}^{k} V^\nu$ be the dependent set of $R_{X_\lambda} \cup R_1$. The dependent set of $R_{X_\lambda} \cup R_1 \cup R_2 \cup \cdots \cup R_i$, $i \in \{2, 3, \ldots, \mu\}$, is denoted by*

$D(R_{X_\lambda} \cup R_1 \cup R_2 \cup \cdots \cup R_i) \subset \bigcup_{\nu=\phi(w_1)}^{k} V^\nu$, *and is recursively defined to be the set of all vertices* $u \in \bigcup_{\nu=\phi(w_1)}^{k} V^\nu$ *such that the following conditions hold.*

1. $w_\nu \in D(R_{X_\lambda} \cup R_1 \cup R_2 \cup \cdots \cup R_{i-1})$.
2. *There exists a directed path* $P = (u_1, u_2, \ldots, u_\xi = u)$, $\xi > 1$, *with* $u_1 \in R_{X_\lambda} \cup R_1 \cup R_2 \cup \cdots \cup R_i$.
3. *There exists a vertex* $\tilde{u} \in \bigcup_{\nu=\phi(w_1)}^{k} V^\nu$ *and a directed path* $\tilde{P} = (\tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_\iota = \tilde{u})$, $\iota > 1$, *with* $\tilde{u}_1 \in R_{X_\lambda} \cup R_1 \cup R_2 \cup \cdots \cup R_i$, *such that* $\phi(\tilde{u}) \geq \phi(u)$ *and the paths* P *and* $\tilde{P}$ *are vertex disjoint except possibly for* u *and* $\tilde{u}$.

Definition 8.2. *A sequence of* w_i*-merge operations,* $i = 1, 2, \ldots \mu$, *of reduced extended forms* R_i *into a reduced extended form* R *is said to be a* concatenation *of merge operations if the following three conditions hold.*

(i) R_1 *is* w_1*-merged using a merge operation* B *or* C.

(ii) *For all* w_i*-merge operations we have* $w_i \in D(R_{X_\lambda} \cup R_1 \cup R_2 \cup \cdots \cup R_{i-1})$.

(iii) R_1 *has not been fixed to one side of* R_{X_λ} *with respect to* R_X *and it is unknown if its embedding can be chosen freely.*

Interestingly, a concatenation of merge operations does not really affect the results of Observations 7.2 and 7.3 and the Lemmas 7.4 and 7.5. This is immediately clear for the observations that we now give for the concatenated case.

Observation 8.3. *Let* v *be the connective cut vertex of* R_{X_λ} *and let* $\tilde{v}$ *be the connective cut vertex of* R_X *if* X *has a parent. Let* R_i, $i = 1, 2, \ldots, \mu$, *be a sequence of partially reduced extended forms that are* w_i*-merged into* R *and their merge operations are concatenations. The subgraph* R_1 *is not fixed to any side of* R_{X_λ} *with respect to* R_X *if and only if for every vertex* u *in the dependent set* $D(R_{X_\lambda} \cup R_1 \cup R_2 \cup \cdots \cup R_\mu)$ *and every undirected path* $P = (v = u_1, u_2, \ldots, u_\xi = u)$, $\xi \geq 2$, *with* $u_i \in \bigcup_{\nu=1}^{\phi(u)} V^\nu$ *for all* $i = 1, 2, \ldots, \xi$, *one of the following conditions holds.*

(i) $u_{\xi-1} \in R_{X_\lambda} \cup R_1 \cup R_2 \cup \cdots \cup R_\mu \cup D(R_{X_\lambda} \cup R_1 \cup R_2 \cup \cdots \cup R_\mu)$.

(ii) $\tilde{v} \in P$ *and for all* $v' \in \vec{R}_X - (R_{X_\lambda} \cup R_1 \cup R_2 \cup \cdots \cup R_\mu \cup D(R_{X_\lambda} \cup R_1 \cup R_2 \cup \cdots \cup R_\mu))$ *the inequality* $\phi(v') < \phi(u)$ *holds.*

(iii) v *and* u *form a split pair in* G *and for all* $v' \in D(R_{X_\lambda} \cup R_1 \cup R_2 \cup \cdots \cup R_\mu) - \{u\}$ *the inequality* $\phi(v') < \phi(u)$ *holds.*

Observation 8.4. *Let v be the connective cut vertex of R_{X_λ} and let $\tilde{v}$ be the connective cut vertex of R_X if X has a parent. Let R_i, $i = 1, 2, \ldots, \mu$, be a sequence of partially reduced extended forms that are w_i-merged into R and their merge operations are concatenations. The subgraph R_1 is fixed to a side of R_{X_λ} with respect to R_X if and only if there exists a vertex u in the dependent set $D(R_{X_\lambda} \cup R_1 \cup R_2 \cup \cdots \cup R_\mu)$ and an undirected path $P = (v = u_1, u_2, \ldots, u_\xi = u)$, $\xi \geq 2$, with $u_i \in \bigcup_{\nu=1}^{\phi(u)} V^\nu$ for all $i = 1, 2, \ldots, \xi$, and all three of the following conditions hold.*

(i) $u_{\xi-1} \notin R_{X_\lambda} \cup R_1 \cup R_2 \cup \cdots \cup R_\mu \cup D(R_{X_\lambda} \cup R_1 \cup R_2 \cup \cdots \cup R_\mu)$.

(ii) *(a)* $\tilde{v} \notin P$, *or*

(b) $\tilde{v} \in P$ *and there exists a* $v' \in \vec{R}_X - (R_{X_\lambda} \cup R_1 \cup R_2 \cup \cdots \cup R_\mu \cup D(R_{X_\lambda} \cup R_1 \cup R_2 \cup \cdots \cup R_\mu))$ *such that* $\phi(v') \geq \phi(u)$.

(iii) There exists a vertex $v' \in D(R_{X_\lambda} \cup R_1 \cup R_2 \cup \cdots \cup R_\mu) - \{u\}$ *such that the inequality* $\phi(v') \geq \phi(u)$ *holds.*

In order to see that the results of Lemmas 7.4 and 7.5 (up to minor differences) still hold, we show that the "local structure" of the PQ-tree at the Q-node Y and its parent X either does not change or, if it changes, the embedding of R_1 is fixed on one side of R_{X_λ} with respect to R_X. With an "unchanged local structure" we express (informally) that throughout concatenated merge operations the node Y (or any node that replaces Y), and X (or any node that replaces X) stay Q-nodes with Y (or its replacing node) remaining unchanged in the position of its siblings. The following lemma formally describes how the Q-node Y is changed during subsequent concatenated merge operations. The results of the lemma then immediately lead to results similar to the ones in Lemmas 7.4 and 7.5.

Lemma 8.5. *Let Y be a Q-node that has been introduced by w_1-merging the PQ-tree T_1 into T using a merge operation B or C, replacing a child X_λ of a Q-node X. Let T_i, $i = 2, 3, \ldots, \mu$, $\mu \geq 2$, be a sequence of PQ-trees that are w_i-merged into T such that the w_i-merge operations are concatenations. Let Y' be the node that occupies the position of Y in the PQ-tree after the w_μ-merge operation is complete. Then Y' and its parent are Q-nodes.*

Proof. Let R_i, $i = 1, 2, \ldots, \mu$, be the forms corresponding to the PQ-trees T_i. We prove the lemma by induction.

According to the definition of a concatenation, R_1 has not been embedded at one side of R_{X_λ} with respect to R_X and it is unknown if its embedding can be chosen freely. Lemma 7.5 therefore implies that the parent of Y did not become a node with Y as the only nonignored child, and according to Corollary 2.2 the parent of Y must be a Q-node. Furthermore, Lemma 7.5 implies that Y must have at least two leaves in its frontier both corresponding to different vertices in G.

When applying a w_2-merge operation to a vertex $w_2 \in D(R_{X_\lambda} \cup R_1)$ three cases are possible.

(i) Only descendants of Y are affected by the merge operation.

(ii) The node Y and its parent are affected by the merge operation.

(iii) Proper ancestors of Y are affected by the merge operation.

Consider the first case. If only proper descendants of Y are involved, neither Y nor its parent are affected. If Y and a child of Y are affected, the merge operations B, C or D are applied to the child of Y. Thus Y remains a Q-node with unchanged position in the PQ-tree.

Consider the second case. Since the parent of Y is a Q-node, the only allowed merge operations are B, C, and D. The operations B and C insert a new Q-node Y' at the position of Y. Reducing the leaves labeled w_2 after the merge operation does only affect the children of Y', since Y' is the root of the pertinent subtree. Therefore, the Q-node Y' remains unchanged in its position. However, if the merge operation D is applied, the template matching algorithm performed directly after the merge operation removes Y from the tree by applying one of the templates Q2 or Q3. Hence, according to Lemma 7.4, R_1 is embedded at one side of R_{X_λ} with respect to R_X. Therefore, the w_1-merge operation of T_1 and the w_2-merge operation of T_2 are not a concatenation.

If proper ancestors of Y are involved, the reduction of the PQ-tree with respect to w_2 removes Y from the PQ-tree by applying one of the templates Q2 or Q3. Again, the w_1-merge operation of T_1 and the w_2-merge operation of T_2 are not concatenated.

The lemma then follows by a simple inductive argument. □

The following lemmas are almost identical to the Lemmas 7.4 and 7.5, taking into account that the subgraph induced by the subtree rooted at Y (or any Q-node that replaces Y due to a merge operation) may have grown by concatenated merge operations.

Lemma 8.6. *Let Y be the Q-node that was introduced by a w_1-merge operation B or C of a PQ-tree T_1 into a tree T, replacing a node X_λ that was a child of a Q-node X in T. Let Y' be a Q-node occupying the position of Y, and Y' has been introduced during a merge operation concatenating the w_1-merge operation. The subgraph R_1 corresponding to T_1 has to be embedded at exactly 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.*

Proof. The lemma follows from Lemma 7.4 and Lemma 8.5. □

Lemma 8.7. *Let Y be the Q-node that was introduced by a w_1-merge operation B or C of a PQ-tree T_1 into a tree T, replacing a node X_λ that was a child of a Q-node X in T. Let Y' be a Q-node occupying the position of Y, and Y' has been introduced during a merge operation concatenating the w_1-merge operation. The subgraph R_1 corresponding to T_1 is not fixed to any side of R_{X_λ} if and only if one of the following cases occurs during the application of the template matching algorithm.*

(i) The Q-node Y' gets ignored.

(ii) The Q-node Y' is a nonignored node of the final PQ-tree.

(iii) The Q-node Y' has only one nonignored child.

(iv) The parent of Y' has only Y' as a nonignored child.

Proof. The lemma follows from Lemma 7.5 and Lemma 8.5. □

Let c be a contact that is a child of the Q-node Y. As long as concatenated merge operations only affect descendants of the Q-node Y, they have no effect on c and its reference sequence. However, if Y and its parent are subject to a merge operation B or C, there exists a coherence between the existing contact c and the new contact that is introduced by the merge operation.

Obviously, the merge operations A, D, and E can be performed one after another without worrying about the correct treatment of involved sink indicators. However, the merge operations B and C may "affect" each other. Consider for instance the example shown in Fig. 21, presenting three forms R_1, R_2, and R_3 that have been successively merged into a form R at the vertices w_1, w_2, w_3. For every form R_i, the example also gives the set of edges that have to be added as incoming edges to w_i, $i \in \{1, 2, 3\}$, in the given embedding. On the other hand, Fig. 22 gives the same example only with a different embedding showing different sets of edges that have to be added as incoming edges to w_i.

In the rest of this section we discuss how to handle sequences of the merge operations B and C that affect each other. We say that two contacts c_1 and c_2 *mutually influence* each other if $\mathrm{ref}(\mathrm{c}_1) \cap \mathrm{ref}(\mathrm{c}_2) \neq \emptyset$. Two merge operations B or C *mutually influence* each other if their corresponding contacts mutually influence each other.

Consider a Q-node Y that has been introduced as a child of a Q-node X applying one of the operations B or C. In case of a w-merge operation B or C, we only need to check if the node Y that has to be replaced by a new Q-node does have a contact as an endmost child. However, the contact is then separated from its reference sequence since Y is not a child of the Q-node X anymore. This seems to destroy the strategy of handling the contact and its reference sequence correctly.

However, the new Q-node is obviously the root of the pertinent subtree with respect to w. Since the two merge operations are concatenated, Lemma 7.5 does not apply to

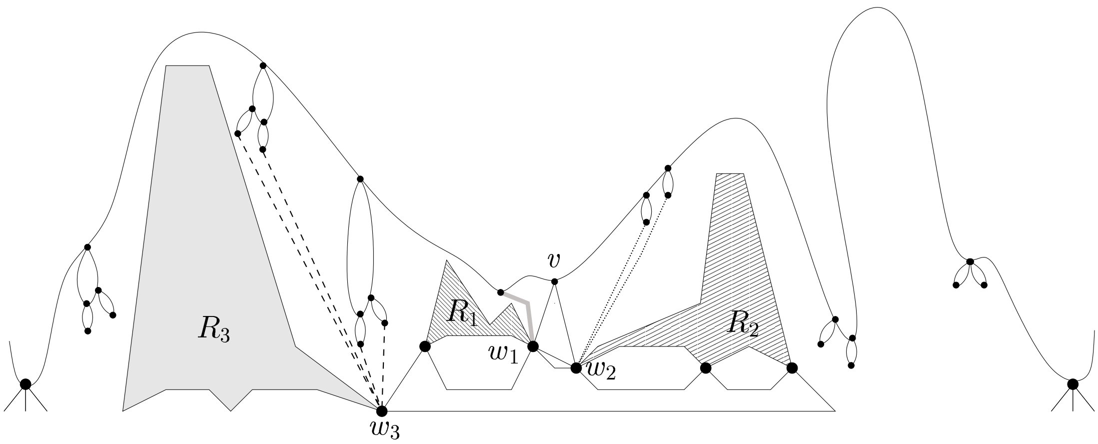


Figure 21: A concatenation of merge operations. First possible embedding of the forms R_1, R_2, and R_3 next to R_{X_λ} with respect to R_X.

Y_1. (Otherwise, simply remove the contact and either the left or right reference set.) It follows that the node Y_1 must be a partial Q-node. Therefore, template Q2 or Q3 is applied to Y_1 and its parent Y_2, and Y_1 is removed from the tree, and the children of Y_1 become children of Y_2. This ensures that after the reduction with respect to w, c_1 is again a child of a Q-node Y_2, where Y_2 is a child of X occupying the former position of Y_1. We consider the position of c_1 within the sequence of children of Y_2. Let c_2 be the contact associated with the merge operation that introduced Y_2. Two cases may occur during the reduction with respect to w.

1. The contact c_1 was at the empty end of Y_1 and since Y_1 was an endmost child of Y_2, c_1 is now an endmost child of Y_2.
2. The contact c_1 was at the full end of Y_1, and appears within the sequence of full children of Y_2.

After having finished the reduction with respect to w one of the following two rules is applied to the contact c_1.

Rule I If c_1 is an endmost child of Y_2, c_1 remains in its position as an endmost child of Y_2.

Rule II If c_1 is found within the sequence of pertinent nodes, c_1 is placed as a new endmost child of Y_2 next to c_2.

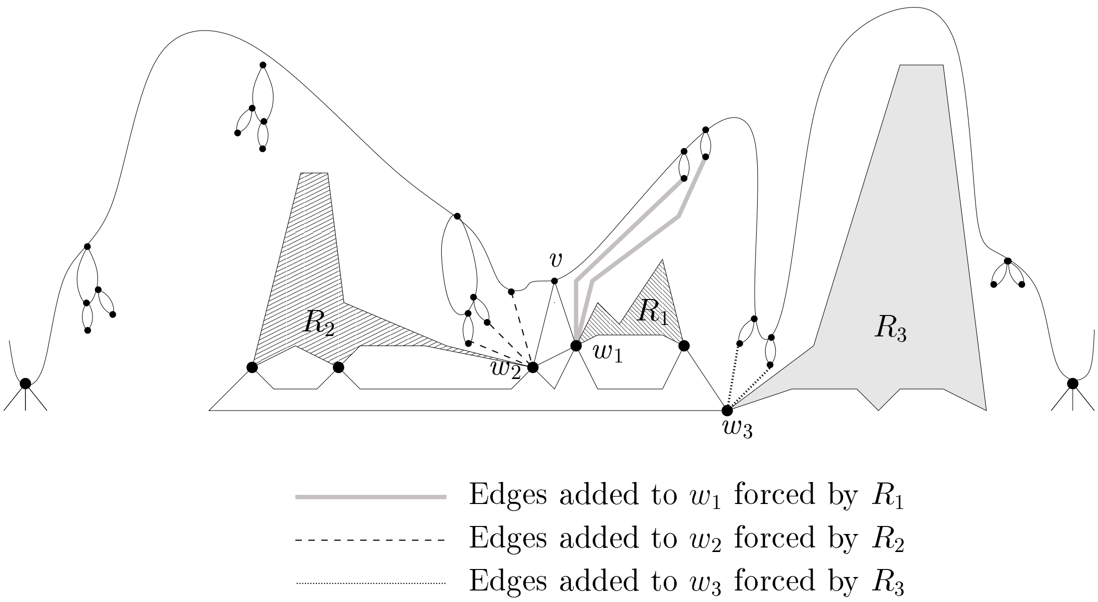


Figure 22: A concatenation of merge operations. Second possible embedding for the forms R_1, R_2, and R_3 next to R_{X_λ} with respect to R_X.

The rules are easily expanded to two or more contacts. Let Y_i be the i-th Q-node introduced by the i-th concatenating merge operation.

Rule I' If a sequence of contacts $c_1, c_2, \ldots, c_\xi$, $\xi \leq i-1$, is endmost at Y_i after the reduction is complete, the contacts $c_1, c_2, \ldots, c_\xi$ remain in their positions.

Rule II' If a sequence of contacts $c_1, c_2, \ldots, c_\xi$, $\xi \leq i-1$, is found within the sequence of pertinent nodes and c_1 was the former endmost child of Y_{i-1}, the sequence $c_1, c_2, \ldots, c_\xi$ is placed next to c_i that c_ξ and c_i are directly siblings and c_1 is a new endmost child of Y_i.

Lemma 8.8. *Let R_i, $i = 1, 2, \ldots, \mu$, be a sequence of partially reduced extended forms that are w_i-merged into a partially reduced extended form R of a level planar graph G such that their merge operations are concatenations. Let $T, T_1, T_2, \ldots, T_\mu$ be the PQ-trees corresponding to $R, R_1, R_2, \ldots, R_\mu$. Let X, $Y_1, Y_2, \ldots, Y_\mu$ be Q-nodes and X_λ be a node such that*

(a) X_λ was a child of X in a PQ-tree T before w_1-merging T_1 into T.

(b) X_λ was replaced by Y_1 when w_1-merging T_1 into T using the merge operation B or C.

(c) Y_i was replaced by Y_{i+1} when w_{i+1}-merging T_{i+1} into T using the merge operation B or C for all $i = 1, 2, \ldots, \mu - 1$.

Let $\mathrm{c}(R_i)$ *be the contact that is associated with the introduction of* Y_i*,* $i = 1, 2, \ldots, \mu$*. Let* R_{X_λ} *be the subgraph corresponding to* X_λ*, and assume that* $\mathrm{c}(R_1), \mathrm{c}(R_2), \ldots, \mathrm{c}(R_{i-1})$ *have been replaced by applying one of the Rules I' or II' when merging* $R_1, R_2, \ldots, R_i$ *into* R*,* $i \geq 2$*. Then exactly one of the following statements holds.*

(i) The contacts $\mathrm{c}(R_1)$ *and* $\mathrm{c}(R_i)$ *are both children at the same end of the* Q*-node* Y_i *if and only if their corresponding forms have to be embedded on the same side of* R_{X_λ} *with respect to* R_X *in every level planar embedding.*

(ii) The contacts $\mathrm{c}(R_1)$ *and* $\mathrm{c}(R_i)$ *are both children on opposite sides of the* Q*-node* Y_i *if and only if their corresponding forms have to be embedded on opposite sides of* R_{X_λ} *with respect to* R_X *in every level planar embedding.*

Proof. Since the merge operations are concatenated, Lemma 7.5 does not apply to Y_i, $i = 1, 2, \ldots, \mu - 1$. It follows that Y_i is a partial Q-node for every $i = 1, 2, \ldots, \mu - 1$. Thus there exist at least two leaves, one corresponding to an incoming edge of w_{i+1} and one corresponding to an incoming edge of a vertex $u \in V^l$, $u \neq w_2$, $\phi(w_{i+1}) \leq l \leq k$.

Let Y_i, $i \in \{2, 3, \ldots, \mu - 1\}$, be the partial Q-node that has to be replaced by a Q-node Y_{i+1} in a w_{i+1} merge operation. Then the set $\{R_1, R_2, \ldots, R_i\}$ partitions into two subsets:

- $\{R_1^1, R_2^1, \ldots, R_\nu^1\}$, $1 \leq \nu \leq i$, the set of forms that are embedded on the same side as R_1 and
- $\{R_{\nu+1}^2, R_2^2, \ldots, R_i^2\}$, the set of forms that are embedded on the opposite side of R_1.

Since Y_i is a partial Q-node, there exists a level planar embedding of R_{Y_i} and two paths

$$\begin{array}{ll}
P = (u_1, u_2, \ldots, u_\sigma) & \sigma \geq 2 \\
u_1 \in R_1^1 \cup R_2^1 \cup \cdots \cup R_\nu^1 & \\
u_j \notin R_{X_\lambda} \cup R_{\nu+1}^2 \cup R_2^2 \cup \cdots \cup R_i^2 - \{w_i\} & j = 1, 2, \ldots, \sigma \\
\phi(u_j) < \phi(w_{i+1}) & j = 1, 2, \ldots, \sigma - 1 \\
\phi(u_\sigma) \geq \phi(w_{i+1}) &
\end{array}$$

and

$$\begin{array}{ll}
P' = (u'_1, u'_2, \ldots, u'_\xi) & \xi \geq 2 \\
u'_1 \in R_{X_\lambda} \cup R_{\nu+1}^2 \cup R_2^2 \cup \cdots \cup R_i^2 & \\
u_j \notin R_1^1 \cup R_2^1 \cup \cdots \cup R_\nu^1 & j = 1, 2, \ldots, \xi \\
\phi(u'_j) < \phi(w_{i+1}) & j = 1, 2, \ldots, \xi - 1 \\
\phi(u'_\xi) \geq \phi(w_{i+1}) &
\end{array}$$

such that

(a) P and P' are disjoint,

(b) both P and P' are on the boundary of the outer face of the embedding of R_{Y_i}, and

(c) either $u_\sigma = w_{i+1}$ or $u'_\xi = w_{i+1}$, but not both.

See Fig. 23 where we have illustrated the case $i = 1$.

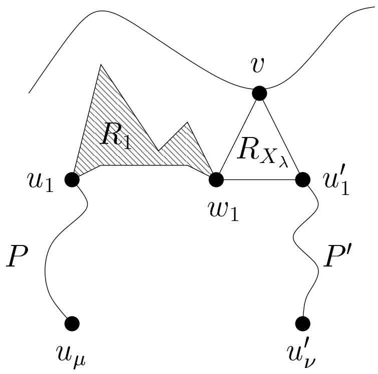


Figure 23: Illustration of the proof of Lemma 8.8.

First, case (i) is proven. Let R_{i+1} and R_1 be embedded on the same side of R_{X_λ}. It follows that $w_{i+1} \in P$, otherwise R_{i+1} and P would cross each other. Let Z be the child of Y_i that is an ancestor of the leaf labeled w_{i+1}. Since the path P is on the outer face of R_{Y_i} on the side where R_1 is embedded, Z must be an endmost nonignored child of Y_i on the side where $\mathrm{c}(R_1)$ is an endmost child. Since Y_i is partial, $\mathrm{c}(R_1)$ will appear within the pertinent sequence of leaves labeled w_{i+1} after the reduction with respect to w_{i+1} is complete. Therefore Rule II' is applied and $\mathrm{c}(R_1)$ and $\mathrm{c}(R_{i+1})$ are both children at the same end of Y_{i+1}.

Now let $\mathrm{c}(R_1)$ and $\mathrm{c}(R_{i+1})$ be children on the same side, and assume that R_1 and R_{i+1} have to be embedded on opposite sides of R_{X_λ} with respect to R_X. It follows that $w_{i+1} \in P'$, otherwise R_{i+1} and P' would cross each other. By construction, $\mathrm{c}(R_1)$ was found within in the pertinent sequence with respect to the vertex w_{i+1} after R_{i+1} was w_{i+1}-merged into R. So there exists a path P'' on the boundary of R_{Y_i}, and P'' connects a vertex $u \in R_1^1 \cup R_2^1 \cup \cdots \cup R_\nu^1$ and w_{i+1}, not using any vertices $u' \in \bigcup_{l=\phi(w_{i+1})}^{k} V^l$. However, P'' must cross P, a contradiction.

The case (ii) is proven analogously to the case (i). $\square$

If two contacts c_1 and c_2 influence each other, and therefore $\mathrm{ref}(\mathrm{c}_1) \cap \mathrm{ref}(\mathrm{c}_2) \neq \emptyset$ holds, we need to redefine their reference sets such that no conflicts appear when sink indicators for edge augmentation are considered. A situation, where we can chose for a sink indicator to which reference set it belongs has to be avoided.

Again let R, R_1, R_2, X, X_λ, Y_1, Y_2, $c(R_1)$, and $c(R_2)$ be defined as in Lemma 8.8. The idea is to leave the reference set of $c(R_1)$ (the contact associated to the "first" merge operation) unchanged, and adapt the reference set of $c(R_2)$ (the contact associated to the "second" merge operation). Let $I_1, I_2, \ldots, I_\mu$, $\mu \geq 0$, be the sequence of ignored nodes on the left side of X_λ with X_λ and I_μ being direct siblings, and let $J_1, J_2, \ldots, J_\rho$, $\rho \geq 0$, be the sequence of ignored nodes on the right side of X_λ with X_λ and J_1, being direct siblings. Let

$$\begin{aligned} \mathrm{ref}(c(R_1)) &= \left(\bigcup_{i=\nu_1}^{\mu} \mathrm{frontier}(I_i)\right) \cup \left(\bigcup_{i=1}^{\sigma_1} \mathrm{frontier}(J_i)\right) \\ & \quad \text{for } 1 \leq \nu_1 \leq \mu+1,\ 0 \leq \sigma_1 \leq \rho \end{aligned}$$

be the reference set of $c(R_1)$, where we assume without loss of generality that none of the two subsets is empty. The reference points of $c(R_1)$ are $I_{\nu_1}, I_\mu, J_1, J_{\sigma_1}$. Assume further that

$$\begin{aligned} \mathrm{ref}(c(R_2)) &= \left(\bigcup_{i=\nu_2}^{\mu} \mathrm{frontier}(I_i)\right) \cup \left(\bigcup_{i=1}^{\sigma_2} \mathrm{frontier}(J_i)\right) \\ & \quad \text{for } 1 \leq \nu_2 \leq \mu+1,\ 0 \leq \sigma_2 \leq \rho\ . \end{aligned}$$

After performing the second merge operation including the reduction of the leaves labeled w_2, the contacts $c(R_1)$ and $c(R_2)$ occupy two relative positions at the their parent Y_2.

(i) $c(R_1)$ and $c(R_2)$ are endmost children on different ends of Y_2. Due to Lemma 8.8, R_1 and R_2 are embedded on opposite sides of R_{X_λ} with respect to R_X. Thus $c(R_1)$ and $c(R_2)$ do not interfere when finally determining the sets of sink indicators that are considered for edge augmentation. We determine the reference points $I_{\nu_2}, I_\mu, J_1, J_{\sigma_2}$ and store them at $c(R_2)$.

(ii) $c(R_1)$ and $c(R_2)$ are at the same end of Y_2 with $c(R_1)$ being (by construction) an endmost child. Due to Lemma 8.8, R_1 and R_2 are embedded at the same side of R_{X_λ}. Thus $c(R_1)$ and $c(R_2)$ interfere when we finally determine the sets of sink indicators that are considered for edge augmentation.

The new reference set of $c(R_2)$ is determined as follows.

$$\mathrm{ref}(c(R_2))^{\mathit{left}} = \begin{cases} \bigcup_{i=\nu_2}^{\nu_1-1} \mathrm{frontier}(I_i) & \text{if } \nu_2 < \nu_1 \\ \emptyset & \text{otherwise} \end{cases} \tag{1}$$

$$\mathrm{ref}(c(R_2))^{\mathit{right}} = \begin{cases} \bigcup_{i=\sigma_1+1}^{\sigma_2} \mathrm{frontier}(J_i) & \text{if } \sigma_2 > \sigma_1 \\ \emptyset & \text{otherwise} \end{cases} \tag{2}$$

Then the reference set of $c(R_2)$ is

$$\mathrm{ref}(\mathrm{c}(R_2)) \quad = \quad \mathrm{ref}(\mathrm{c}(R_2))^{right} \cup \mathrm{ref}(\mathrm{c}(R_2))^{left} \ .$$

Hence, the ignored nodes $I_{\nu_2}, I_{\nu_1-1}, J_{\sigma_1+1}, J_{\sigma_2}$ are stored as reference points at $\mathrm{c}(R_2)$.

The application to the more general case of three or more contacts is straightforward.

Remark 8.9. *In order to achieve linear running time, the reference sequence of a newly introduced contact* $\mathrm{c}(R_i)$ *and its associated form* R_i *is never determined by scanning the sequence of ignored siblings. Instead we proceed as follows. At every consecutive sequence of contacts, we keep a pointer at the endmost contact* c_α *towards the innermost contact* c_ω*. The contact* c_ω *is obviously the contact that has been introduced last in this sequence of contacts. When introducing a new contact, we only need to consider the contacts on the same side as the new contact. We check the reference sequence of the innermost contact* c_ω *and determine the* ML-*values* $\mathrm{ML}(I_{\nu_{\omega-1}}, I_{\nu_\omega})$ *and* $\mathrm{ML}(J_{\sigma_\omega}, J_{\sigma_{\omega+1}})$ *between the reference sequence and their direct siblings. If* $\mathrm{ML}(I_{\nu_{\omega-1}}, I_{\nu_\omega}) < \mathrm{LL}(R_i)$ *or* $\mathrm{ML}(J_{\sigma_\omega}, J_{\sigma_{\omega+1}}) < \mathrm{LL}(R_i)$*, the left or right reference set of* $\mathrm{c}(R_i)$*, respectively, is empty. If* $\mathrm{ML}(I_{\nu_{\omega-1}}, I_{\nu_\omega}) \geq \mathrm{LL}(R_i)$ *or* $\mathrm{ML}(J_{\sigma_\omega}, J_{\sigma_{\omega+1}}) \geq \mathrm{LL}(R_i)$*, the left or right reference sequence of* $\mathrm{c}(R_i)$*, respectively, is determined in constant time using the pointers that we have installed as described in Remark 6.6.*

Consider the case were $\mathrm{c}(R_1)$ and $\mathrm{c}(R_2)$ are at the same end of Y_2. when removing the Q-node Y_2 during the application of the template Q2 or Q3. The contact $\mathrm{c}(R_1)$ is an endmost child of Y_2. Thus, after the application of the template Q2 or Q3 the contact $\mathrm{c}(R_1)$ is a direct sibling of either I_μ or J_1. Therefore, the identification of sink indicators that have to be considered for edge augmentation joining the vertex w_1 is a straightforward matter. After this identification is finished, the contact $\mathrm{c}(R_1)$ and the set of ignored siblings that were considered for augmentation are removed from the PQ-tree, leaving the contact $\mathrm{c}(R_2)$ as a direct sibling of either I_{ν_1-1} or J_{σ_1+1}. Again, the identification of the sink indicators that have to be considered for edge augmentation joining the vertex w_2 is straightforward.

Lemma 8.10. *Let* c_i*,* $i = 1, 2, \ldots, \mu$*,* $\mu \geq 1$*, be contacts that are endmost children of a Q-node* Y_1 *in a PQ-tree* T*. Let contact* c_i *be related to vertex* $w_i \in V$*, such that* $\phi(w_i) \leq \phi(w_{i+1})$*,* $i = 1, 2, \ldots, \mu - 1$*. In case that* $\phi(w_i) = \phi(w_{i+1})$ *holds, let the PQ-tree* T_i *corresponding to* c_i *be* w_i*-merged into* T *before the tree* T_{i+1} *corresponding to* w_{i+1} *is* w_{i+1}*-merged into* T*. Let* $\mathrm{ref}(\mathrm{c}_i)^{left}$ *be the left reference set of* c_i *with reference points* I_i^b, I_i^e *and* $\mathrm{ref}(\mathrm{c}_i)^{right}$ *be the right reference set of* c_i *with reference points* J_i^b, J_i^e*. For every* c_i*, the nodes* I_i^b *and* J_i^b *denote the first ignored node in the reference sequence* $\mathrm{rseq}(\mathrm{c}_i)^{left}$ *and* $\mathrm{rseq}(\mathrm{c}_i)^{right}$*, respectively, and* I_i^e *and* J_i^e *denote the last ignored node in the reference sequence* $\mathrm{rseq}(\mathrm{c}_i)^{left}$ *and* $\mathrm{rseq}(\mathrm{c}_i)^{right}$*, respectively. Then the following statements hold true.*

(i) If c_i *is adjacent to* I_i^b*, then augmenting* G_{st} *by an edge* (u, w) *for every* $\mathrm{si}(u) \in \mathrm{ref}(c_i)^{left}$ *does not destroy level planarity.*

(ii) If c_i *is adjacent to* J_i^b*, then augmenting* G_{st} *by an edge* (u, w) *for every* $\mathrm{si}(u) \in \mathrm{ref}(c_i)^{right}$ *does not destroy level planarity.*

Proof. By construction, the sequence of children of Y_1 is partitioned into the three sets: C_1, N and C_2 where

- $C_1 = c_1^1, c_2^1, \ldots, c_\nu^1$, $0 \leq \nu \leq \mu$, is the sequence of contacts on one end of Y_1 with c_1^1 being an endmost child of Y_1.
- N is a sequence of ignored and nonignored nodes.
- $C_2 = c_1^2, c_2^2, \ldots, c_\varphi^2$, $\varphi = \mu - \nu$, is the sequence of contacts on the opposite side of C_1 with c_1^2 as an endmost child of Y_1.

By construction, we have for C_ξ, $\xi = 1, 2$,

$$\phi(\omega(c_i^\xi)) \quad \leq \quad \phi(\omega(c_{i+1}^\xi)) \quad i = 1, 2, \ldots, |C_\xi| - 1$$

where $\omega(c_i^\xi)$ denotes the vertex related to c_i^ξ. For any $i \in \{1, 2, \ldots, |C_\xi| - 1\}$ with $\phi(\omega(c_i^\xi)) = \phi(\omega(c_{i+1}^\xi))$ the PQ-tree corresponding to c_i^ξ has been $\omega(c_i^\xi)$-merged into T before the tree corresponding to c_{i+1}^ξ has been $\omega(c_{i+1}^\xi)$-merged.
It follows that either $c_1^1 = c_1$ or $c_1^2 = c_1$ and c_1^1 and c_1^2 do no interfere, since their corresponding forms are placed on opposite sides with respect to R_{X_λ} and R_X. We may assume that $c_1^1 = c_1$. Due to Observation 7.8, c_1^1 is either adjacent to I_1^b or to J_1^b. Assume without loss of generality that c_1^1 is adjacent to I_1^b. It follows from Lemma 7.9 that considering $\mathrm{ref}(c_1)^{left}$ for edge augmentation does not destroy level planarity. Removing c_1^1 and $\mathrm{ref}(c_1)^{left}$ from the tree, the correctness of the lemma follows by a simple inductive argument. □

The function AUGMENT now combines all the described strategies with in the level planarity test of Jünger et al. [1999]. It is almost identical to the function LEVEL-PLANARITY-TEST, except that it does not call the function CHECK-LEVEL but a function EMBED-LEVEL. However, the function EMBED-LEVEL is almost identical to the function CHECK-LEVEL. We only need the following modifications.

(i) If v is a sink in V^j, $1 < j < k$, replace the corresponding leaf by a sink indicator $\mathrm{si}(v)$ before processing G^{j+1}. If this replacement constructs a node X having only sink indicators in its frontier, mark X as ignored and update the ML-values as described in Section 6.

(ii) When reducing a set of leaves with respect to a vertex w in a PQ-tree, ignore all sink indicators and ignored nodes during the application of the template matching algorithm. After the reduction is complete, the pertinent subtree is removed from the tree and replaced by a single representative. During the removal of the pertinent subtree with respect to w, we check for sink indicators in the pertinent subtree. For every $\mathrm{si}(v)$ that is found in the pertinent subtree, we add an edge (v, w) to G_{st}, unless $\mathrm{si}(v)$ is affected by the existence of a contact c in the pertinent subtree. If the latter applies, add an edge (v, w'), with w' being the vertex related to c.

(iii) When w-merging a PQ-tree T' into a PQ-tree T, necessary adjustments as described above have to be applied to the merge operations B or C. If necessary, a contact is introduced as a third child of the new Q-node. Furthermore, if the new contact mutually influences existing ones, the Rules I or II (see 8) have to be applied after reducing T with respect to w.

(iv) After processing the level k, an edge (v, t) is added for every vertex $v \in V^k$. Furthermore we scan the final PQ-tree T for remaining sink indicators, and add for every indicator $\mathrm{si}(v)$ an edge (v, t) to G_{st}, unless $\mathrm{si}(v)$ is affected by the existence of a contact c in the pertinent subtree. If the latter applies we add an edge (v, w'), with w' being the vertex related to c.

Theorem 8.11. *The algorithm LEVEL-PLANAR-EMBED computes a level planar embedding of a level planar graph $G = (V, E, \phi)$ in $\mathcal{O}(n)$.*

Proof. From Lemmas 5.1, 5.2, 5.3 and 8.10 it follows that augmenting G_{st} to a hierarchy using the function AUGMENT does not destroy level planarity. Consequently, the augmentation of G_{st} to a single source, single sink graph does not destroy level planarity either. The vertices s and t are on the outer face of a level planar embedding of G_{st}. By the discussion of Section 4 we can compute a topological sorting of G_{st} that induces an st-numbering and applying the planar embedding algorithm of Chiba et al. [1985] to G_{st} a level planar embedding of G can be constructed. Since the number of edges added to G to construct G_{st} is bounded by n, the level planar embedding is computed in $\mathcal{O}(n)$ time.

It remains to show that augmenting G_{st} to a hierarchy can be done in $\mathcal{O}(n)$ time. The function AUGMENT performs as the function LEVEL-PLANARITY-TEST, with certain modifications. It is sufficient to show that the amount of extra work performed by these modifications consumes $\mathcal{O}(n)$ time. Clearly, the maintenance of the ignored nodes during all template reductions, and all merge operations A, D, and E is bounded by the number of ignored nodes in the PQ-trees. The number of ingnored nodes is in $\mathcal{O}(n)$, thus it remains to show that the number of operations needed to perform merge operations B and C is as well bounded by the number of ignored nodes.

As described in Remark 8.9 the installation of a contact and the identification of its corresponding reference sequence is bounded by a constant number of operations. Clearly, the number of operations for deinstalling all contacts and their reference sequences is bounded by the number of ignored nodes. Hence, the amount of time needed to handle ignored nodes during the application of the merge operations is in $\mathcal{O}(n)$. □

9 Remarks

Once a level graph has been level planar embedded, we want to visualize it by producing a level planar drawing. This is very simple for proper graphs. Assign the vertices of every level integer x-coordinates according to the permutation that has been computed by CONSTRUCT-LEVEL-EMBED, and draw the edges as straight line segments. This produces a level planar drawing and after applying some readjustments such a drawing can be aesthetically pleasing.

For level graphs that are not necessarily proper, this approach is not applicable. It would be necessary to expand the level graph in horizontal direction for drawing the edges as straight line segments. If many long edges exist in the graph, the area that is needed will be rather large, and the drawings are not aesthetically pleasing.

However, 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, we draw the st-graph G_{st}, and remove afterwards all edges and the vertices s and t that are not contained in G.

Drawing st-graphs has been extensively studied recently (see, e.g, Kant [1993], Luccio, Mazzone, and Wong [1987], Rosenstiehl and Tarjan [1986], Tamassia and Tollis [1986], and Tamassia and Tollis [1989]). Suitable approaches for drawing the st-graph G_{st} have been presented by Di Battista and Tamassia [1988] and Di Battista, Tamassia, and Tollis [1992]. 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 [1988] produces in $\mathcal{O}(n)$ 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 $\mathcal{O}(n)$ time a level planar polyline grid drawing of G_{st} such that the drawing of G_{st} has $\mathcal{O}(n^2)$ 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 $\mathcal{O}(n)$ time.

References

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.

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.

G. Di Battista and R. Tamassia. Algorithms for plane representations of acyclic digraphs. *Theoretical Computer Science*, 61:175–198, 1988.

G. Di Battista, R. Tamassia, and I. G. Tollis. Constrained visibility representations of graphs. *Information Processing Letters*, 41:1–7, 1992.

S. Even and R. E. Tarjan. Computing an st-numbering. *Theoretical Computer Science*, 2:339–344, 1976.

D. R. Fulkerson and O. A. Gross. Incidence matrices and interval graphs. *Pacific J. Mathematics*, 15:835–855, 1965.

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.

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, july 1996.

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.

M. Jünger, S. Leipert, and P. Mutzel. Level planarity testing in linear time (full version). Technical report, Institut für Informatik, Universität zu Köln, 1999.

G. Kant. A more compact visibillity representation. In J. van Leeuwen, editor, *Proc. 19th International Workshop on Graph-Theoretical Concepts in Computer Science*, Lecture Notes in Computer Science. Springer Verlag, 1993.

S. Leipert. *Level Planarity Testing and Embedding in Linear Time.* PhD thesis, Universität zu Köln, 1998.

F. Luccio, S. Mazzone, and C. Wong. A note on visibility graphs. *Discrete Mathematics*, 64:209–219, 1987.

P. Rosenstiehl and R. E. Tarjan. Rectilinear planar layouts and bipolar orientations of planar graphs. *Discrete and Computational Geometry*, 1:343–353, 1986.

R. Tamassia and I. G. Tollis. A unified approach to visibillity representations of planar graphs. *Discrete and Computational Geometry*, 1:321–341, 1986.

R. Tamassia and I. G. Tollis. Tessellation representations of planar graphs. In *Proc. 27th Allerton Conf. Communication Control Computing*, pages 48–57, 1989.