UNIVERSITÄT DORTMUND

Planarity Testing for C-Connected Clustered Graphs

Elias Dahlhaus
Karsten Klein
Petra Mutzel

Algorithm Engineering Report
TR06-1-001
July 2006

ISSN 1864-4503

University of Dortmund
Department of Computer Science
Algorithm Engineering (LS 11)
44221 Dortmund / Germany
http://ls11-www.cs.uni-dortmund.de/

PLANARITY TESTING FOR C-CONNECTED CLUSTERED GRAPHS

ELIAS DAHLHAUS, KARSTEN KLEIN, AND PETRA MUTZEL

Abstract. We present a linear time algorithm for testing clustered planarity of c-connected clustered graphs and for computing a clustered planar embedding for such graphs[1]. Our algorithm uses a decomposition of the input graph based on SPQR-trees and is the first linear time algorithm for clustered planarity testing. We define a normal form of clustered embeddings and show that a clustered graph is clustered planar if and only if any of its normal form embeddings is clustered planar. We also give a combinatorial characterization of clustered planar embeddings and show how to test clustered planarity of a given embedding of a clustered graph.

1. Introduction

In several application domains it is necessary to draw graphs in a way that groups vertices into *clusters*. Code packages in Software Engineering tools may for example be represented as clusters, large computer networks are often partitioned into smaller subsets representing local or structural parts of the network, in VLSI design the building blocks of an electronic circuit should be placed near each other in the layout, and the members of different departments in a business process model can be separated by modelling the departments as clusters. Such a clustering structure may be defined by a rooted tree T representing a hierarchy where the leaves of T are the vertices of G and each inner node ν of T corresponds to the cluster $V(\nu)$ whose vertices are the leaves of the subtree rooted at ν. In a drawing of a graph with a given clustering structure, the clusters should be drawn as simple regions, e.g., bounded by a rectangle. In many applications it is reasonable to assume that the clusters induce connected subgraphs, we only consider graphs with this property in this paper. The complexity of the clustered planarity testing problem for arbitrary clustered graphs is still open. This paper is organized as follows. Section 2 recalls known results on clustered graphs, planar embeddings and graph decomposition. A new combinatorial characterization of clustered planar embeddings is given in Section 3, where the main components of this characterization are also extended to be used within skeleton graphs of SPQR-trees. A normal form of clustered planar embeddings is presented in Section 4. The clustered planarity test for biconnected graphs is described in Sections 5 and 6, the computation of a clustered planar embedding is given in Section 6, Section 6.3 deals with the time complexity of our approach, and Section 6.4 presents the extension to general connected graphs.

1.1. **Previous Work and Our Approach.** Feng, Cohen and Eades introduced the concept of planarity for clustered graphs. They showed, that a connected clustered graph $C = (G, T)$ is c-planar if and only if there is a planar drawing

[1]A preliminary version of this paper appeared in [5].

of G such that for each cluster ν in the cluster tree T and its induced subgraph $G(\nu)$, all the vertices and edges of $G - G(\nu)$ are in the outer face of the drawing of $G(\nu)$, and gave a quadratic time algorithm to recognize c-connected clustered planar graphs based on PQ-trees [8]. A related problem, where the input is given by a replacement system, was studied by Lengauer already in 1986 [13]. Lengauer gave a linear time algorithm, but the size of the input might exceed the order of the number of vertices and therefore the time bound is linear in the order of the input size, but not linear in the order of the number of vertices. Di Battista et al. [1] gave an approach to planarize non c-planar but c-connected clustered graphs and described a drawing algorithm based on the topology-shape-metrics approach.

We give an modified version of the work in [5] based on the usage of SPQR-trees. After the development of the linear time c-planarity testing algorithm in [5], several characterizations of graph classes were given for which polynomial planarity testing algorithms are possible without having the condition of c-connectedness. Cortese et al. [4] consider a class of clustered graphs whose underlying graph is a cycle, Cornelsen and Wagner [3] study completely connected clustered graphs, i.e., clustered graphs where for each cluster not only the induced subgraph but also the induced subgraph of the complement is connected. They show that such a clustered graph is c-planar if and only if the underlying graph is planar. An algorithm for testing clustered planarity of so called extrovert clustered graphs, where disconnected clusters need to fulfill a special connectivity property, is given by Goodrich et al. [9]. Another interesting class of clustered graphs is studied by Gutwenger et al. [10] al. They consider clustered graphs where all nodes in the cluster tree T that correspond to non c-connected clusters lie on the same path in T starting at the root of T, or where alternatively for each non c-connected cluster its super-cluster and all its siblings in T are connected. None of these characterizations have yet led to a promising approach for the general case.

We use an approach based on graph decomposition that uses SPQR-trees together with a criterion for testing clustered planarity of a c-connected clustered graph with a fixed embedding. We develop a normal form for clustered planar embeddings, show how to compute such an embedding for the input graph in linear time and show that clustered planarity of the input graph is equivalent to clustered planarity of any of the possible normal form embeddings for that graph. The final step then is to apply the testing criterion to the computed normal form embedding in order to test clustered planarity of the input graph. The graph decomposition only works for biconnected graphs, we show how to extend the approach to general connected graphs.

2. Preliminaries

For graph terminology and the basic concepts of graph connectivity and planarity, we refer the reader to [6]. Our basic definitions concerning clustered graphs follow the work of Cohen, Eades and Feng [7] and the definitions in [5].

A *block* of a graph is the induced subgraph of a maximal subset of vertices that is biconnected. A *cut vertex* is a vertex whose removal disconnects the graph. The block tree B_T of G consists of the blocks of G and the vertices of G as vertices. A block b is adjacent with a vertex x of G iff x is in b. Two blocks that have a cut vertex in common are therefore also connected in the block tree by a path over the cut vertex.

2.1. **Clustered Graphs.** A *clustered graph* $C = (G, T)$ consists of an undirected graph G and a rooted tree T where the leaves of T are the vertices of G. Each node ν of T represents a *cluster* $V(\nu)$ of the vertices of G that are leaves of the subtree rooted at ν. The tree T therefore describes an inclusion relation between clusters. If ν' is a descendant of ν in T, then we call $V(\nu')$ a *sub-cluster* of $V(\nu)$. The subgraph of G induced by $V(\nu)$ is denoted by $G(\nu)$. For simplicity, we will identify a node ν in T with the cluster $V(\nu)$ in the following. A clustered graph is *c-connected*, if each cluster induces a connected subgraph of G. In the following we assume that the given clustered graph is always c-connected.

2.2. **Embeddings and Clustered Planarity.** A *combinatorial embedding* of a planar graph G is defined as a clockwise ordering of the incident edges for each vertex of G with respect to a crossing-free drawing of G in the plane. A *planar embedding* is a combinatorial embedding together with a specified *external face*.

A *drawing of a clustered graph* $C = (G, T)$ is a representation of C in the plane, where the underlying graph G is drawn as usual and for each node ν in T the cluster is drawn as a simple closed region R that contains the drawing of $G(\nu)$ such that: (i) the regions for all sub-clusters of ν are completely contained in the interior of R; (ii) the regions for all other clusters are completely contained in the exterior of R; (iii) the drawing of each edge between two vertices of ν is completely contained in R. The drawing of an edge e and a region R have an *edge-region crossing* if the drawing of e crosses the boundary of R more than once.

A clustered graph C is *clustered-planar* (*c-planar* for short), if there is an embedding of C into the plane that allows a drawing without edge-edge or edge-region crossings. Such an embedding is called a *clustered planar embedding*.

Theorem 2.1. [8] *A connected clustered graph $C = (G, T)$ is c-planar if and only if G is planar and there exists a planar drawing of G such that for each node ν of T, all the vertices and edges of $G - G(\nu)$ are in the outer face of the drawing of $G(\nu)$.*

2.3. **Graph Decomposition and SPQR-trees.** We shortly describe a graph decomposition based on SPQR-trees [2]. They comprise a decomposition tree of a biconnected graph according to its *split pairs*. A split pair is a pair of nodes that is either connected by an edge or has the property that its removal increases the number of connected components. SPQR trees can be efficiently implemented in linear time [11]. The decomposition allows us to compute a normal form embedding that will be used for the clustered planarity test within our algorithm. Our description follows the one given in [14].

Construction of SPQR-Trees. Let G be a biconnected graph. The construction of an SPQR-tree $\mathcal{T}$ for G works recursively starting at an arbitrary edge e of G that is called the *reference* edge of $\mathcal{T}$. The end-nodes of e are used as the split pair associated with the first node (root) of $\mathcal{T}$. At every node μ of $\mathcal{T}$, the graph is split into the *split components* of the split pair associated with that node, i.e., the maximal subgraphs of the original graph, for which p is not a split pair. To make sure that the split components are biconnected, we add an edge to them and continue by computing their SPQR-tree. The resulting trees are made subtrees of the node used for splitting.

Concepts and Elements of SPQR-trees. The two vertices of the split pair p associated with a node μ are called the *poles* of μ. Each node μ of T has an associated

planar *st*-graph, called the *skeleton* of μ. The skeleton associated with a split pair p is a simplified version of the whole graph where some split components are replaced by single edges.

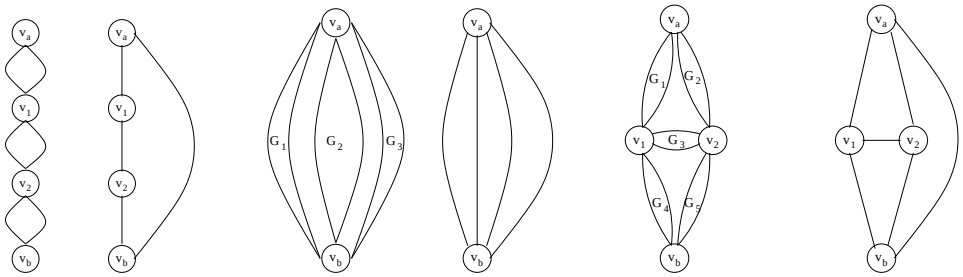
The *pertinent graph* of a node μ is the subgraph of the original graph that is represented by the subtree rooted at μ. Also, each node μ that is not the root of the tree is associated with an edge e of the skeleton of the parent ν of μ, called the *virtual edge* of μ in $skeleton(\nu)$. We call node μ the *pertinent* node of skeleton edge e, denoted by $pert(e)$. Each edge e in a skeleton represents a subgraph of the original graph. This graph together with e is called the expansion graph of e.

The decomposition tree with respect to reference edge e is a rooted ordered tree whose nodes are of four types that are defined by the structure and number of the split components of its poles:

Q-nodes are the leaves of the tree, there is one Q-node for each edge e in the graph. The skeleton consists of the two poles that are connected by two edges, where one edge represents the edge e and the other edge the rest of the graph.
S-nodes represent a serial structure, the pertinent graph has at least one cut vertex. The cut vertices $v_1, \ldots, v_k$ split the pertinent graph into the components $G_1, \ldots, G_{k+1}$. In the skeleton, the G_i are replaced by single edges, and the edge between the poles is added. The decomposition continues with the subgraphs G_i, where the poles are v_i and v_{i+1}.
P-nodes represent a parallel structure where the poles v_a, v_b have at least two split components $G_1, \ldots, G_k$ in the pertinent graph. In the skeleton, each G_i is replaced by a single edge and the edge between the poles is added. The decomposition continues with the subgraphs G_i, where the poles are again v_a and v_b.
R-node None of the other cases is applicable, so the pertinent graph is biconnected. The poles v_a and v_b are not a split pair of the pertinent graph. The decomposition now depends on the *maximal split pairs* of the pertinent graph with respect to the pair v_a, v_b. A split pair v_1, v_2 is maximal with respect to v_a, v_b, if for every other split pair there is a split component that includes the vertices v_a, v_b, v_1, v_2. For each maximal split pair p with respect to v_a, v_b, we define a subgraph G_p of the original graph as the union of all the split components of p that do not include v_a and v_b. In the skeleton, each subgraph is replaced by a single edge and the edge between the poles is added. The decomposition proceeds with the subgraphs defined by the maximal split pairs.

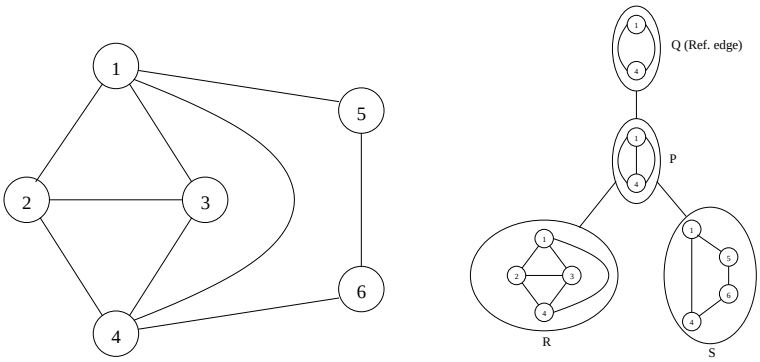
Figure 1 shows the pertinent graph together with the corresponding skeleton for an S-, P-, and R-node. We will omit the discussion of Q nodes in the rest of the paper because they are not needed to code the embedding of the graph G.

If G is 2-connected and planar, its SPQR-tree $\mathcal{T}$ represents all combinatorial embeddings of G. In particular, a combinatorial embedding of G uniquely defines a combinatorial embedding of each skeleton in $\mathcal{T}$, and fixing the combinatorial embedding of each skeleton uniquely defines a combinatorial embedding of G.



(a) S-node (b) P-node (c) R-node

Figure 1: Pertinent graphs and skeletons of the different node types of an SPQR-tree



(a)

Figure 2: A graph G and its SPQR-tree (Q-nodes of R- and S-node omitted)

3. Clustered Planar Embeddings

In this section we give a combinatorial characterization of c-planar embeddings based on a weighting criterion on the dual graph. The combinatorial characterization allows us to judge if a fixed embedding is a clustered planar embedding. But in order to check cluster planarity efficiently we have to consider all possible embeddings without enumerating them. We will use the fact that SPQR-trees can be used to represent all possible combinatorial embeddings of a planar biconnected graph. Therefore we also discuss how to extend the notion of an edge and face weight to the substructures occurring in SPQR-trees.

3.1. **Combinatorial Characterization.** Observe that in a c-planar embedding Γ of a c-connected clustered graph $C = (G, T)$, clusters appear as connected areas without holes. That is, for a cluster ν there may not be any parts of a cluster ν', that

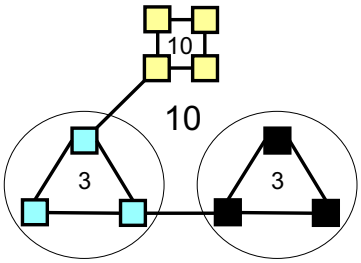
is not a descendant of ν in the inclusion tree, enclosed by a cycle K consisting only of edges in ν. See Figure 4(a). If such an inclusion occurs in a planar embedding Γ' of G, i.e., if a hole exists and therefore Γ' is not c-planar, there has to be some path connecting nodes in ν' to nodes on K since G is connected. Due to the c-connectivity of the clustered graph, at least one of the edges on the path has to pass through the least common ancestor of ν and ν' in the cluster tree. We exploit this property to characterize correct embeddings of c-planar graphs.

Let $weight(c)$ of a cluster c be the number of nodes in c, i.e., the weight may only increase on a path from a leaf to the root of the cluster tree T. For any edge $e = (v, w)$, let $weight(e)$ be the weight of the smallest cluster $c \in C$ that contains v and w, which is the least common ancestor of v and w in T. The weight of a face f, *weight(f)*, is the maximum weight of an edge that belongs to the face cycle of f. A face f therefore has as weight the maximum weight of the clusters whose regions share some part with f, which belongs to the least common ancestor of the vertices on the face boundary in T. We interpret the weights of the faces and edges of G as weights of the nodes and edges of the dual graph G' of Γ and G. A hole in a cluster then corresponds to a cyle of edges with weight at most $i-1$ that encloses a subgraph with weight at least i.

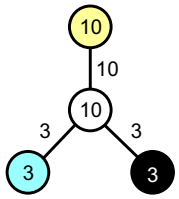
A clustered planar embedding now can be characterized as follows: For each i, let F_i be the set of faces f of Γ with $weight(f) \geq i$ and E_i be the set of edges e of G with $weight(e) \geq i$. Let E'_i be the set of edges in G' corresponding to the edges in E_i. Then (F_i, E'_i) represents a subgraph of G'. See Figure 3 for an example. Note that there cannot be any edges $e = (f_1, f_2)$ with weight k, where $k > weight(f_1)$ or $k > weight(f_2)$.

Theorem 3.1. (Clustered Planar Embeddings) *A planar embedding Γ of $G = (V, E)$ is a clustered planar embedding of G and $C = (G, T)$ if and only if for each i with $F_i \neq \emptyset$, (F_i, E'_i) is a connected subgraph of the dual graph of Γ and a face of maximum weight is taken as outer face.*

Proof. Suppose (F_i, E_i) is not connected. Then at most one of its components C_i can contain the vertex representing the outer face. Let us w.l.o.g. assume that C_1 does not contain the outer face. The faces in F_i have an outer cycle Z in G. The edges of Z are represented in G' by edges that leave C_1. Because C_1 is a connected component of the subgraph consisting of edges with weight at least i, the edges on Z are of weight at most $i-1$. Therefore the vertices that appear on Z are in a common cluster c of weight at most $i-1$. On the other hand each $f \in C_1$ is of weight at least i and therefore contains at least one edge e with weight $\geq i$. But then the vertices incident to e cannot be in c, i.e., Z surrounds vertices that are not in c and c has a hole. Using the same reasoning, we can show that a face of maximum weight needs to be chosen as outer face. If a face that has not maximum weight is chosen as the outer face, then the outer cycle is in a common cluster that is not of maximum weight and therefore the outer cycle surrounds faces of larger weight and c has therefore a hole. Vice versa, suppose the cluster c of weight $i-1$ has a hole with vertex set H and let Z be the innermost cycle of c that surrounds H. Then all inner faces f that share an edge with Z are of weight at least i. All these faces f can reach the outer face (of maximum weight) only through edges of Z. Therefore the outer face and the inner faces of Z sharing an edge with Z are in different connected components of (F_i, E_i). □

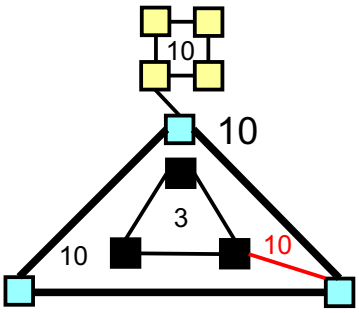


(a) A clustered planar embedding

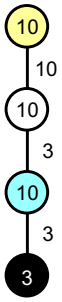


(b) and its dual graph

Figure 3: A c-connected clustered graph with face weights, circles denote clusters



(a) Non c-planar embedding, thick edges denote a cluster enclosing cycle



(b) and its dual graph

Figure 4: The same graph as in Fig. 3 with an embedding that is not clustered planar, the dual graph is not connected for weights ≥ 10

3.2. **Extension of the Weight Property.** In this section we give an extension of the weight property defined in Section 3.1 to the faces and edges of the skeletons in the SPQR-tree. The extension will then be used in Section 4 to compute a normal form embedding. We start our graph decomposition by selecting an edge with maximum weight as the reference edge; this edge will later define the outer face.

3.2.1. *Correlation of Faces in Skeletons and Pertinent Graphs.* Faces in an embedding of the skeleton graph of a tree node μ correspond to faces in the corresponding embedding of the pertinent graph of μ in the following way. Let G be the given graph and pg_μ denote the pertinent graph associated with μ, that is, pg_μ is a subgraph of G. If μ is for example the Q node associated with the reference edge, then

pg_μ is the whole graph G. We say that f is a face of pg_μ if all its boundary edges belong to pg_μ. We call f a *proper face* of pg_μ if it is a face of pg_μ but not of any $pg_{\mu'}$, such that μ' is a child of μ in the SPQR-tree. This means that there has to be a face in the embedded skeleton of μ that can be identified with f up to replacement of border edges as follows. Let f be a proper face of pg_μ, μ' a child node of μ, such that $pg_{\mu'}$ contains boundary edges of f, and v and w be the poles of μ'. Then the boundary edges of f that belong to $pg_{\mu'}$ form a subpath $p_{\mu'}$ of the face cycle of f starting at v and ending at w (see Fig. 5). Replacing each $p_{\mu'}$ by the (virtual) edge between the split pair of μ' defines a face f' of the embedded skeleton of μ. We call such a face a *proper face* of a skeleton to show the correlation to proper faces of the pertinent graphs. Let sg_μ denote the skeleton graph of node μ in the following. The proper faces of sg_μ are exactly the faces that are different from the outer face, i.e., the faces that are not adjacent to the virtual edge connecting the poles of μ.

(a) Skeleton (b) Pertinent graph

Figure 5: A proper face of the skeleton in a P node, and the corresponding face in the pertinent graph after expansion. The dashed and the dotted line denote the two subpaths belonging to child nodes.

3.2.2. *Weights in the SPQR-tree.* We would like to associate weights of faces of the embedding of G with weights of corresponding faces (as described in Section 3.2.1) in the skeletons of the SPQR-tree. Recall that the weight of a face f of G (and thus also of a proper face of some pg_μ) is the maximum weight of a boundary edge of f. Therefore we would also like to give skeleton edges in the SPQR-tree weights such that the weight of a face in a skeleton can be defined too as the maximum weight of the boundary (skeleton) edges and is equal to the desired face weight of the corresponding proper face in the pertinent graph.

Let $g_1, \ldots, g_k$ be the boundary edges of a proper skeleton face f_μ of a node μ and let f be the corresponding proper face of pg_μ. For each $g_i = (v_i, w_i)$ we consider a path p_i in pg_μ from v_i to w_i such that the maximum weight of an edge on p_i is minimized (i.e., a path whose vertices have the smallest least common ancestor in the cluster tree T). We call this skeleton edge weights *axis weights*, denoted by aw_{g_i}, and the paths p_i with maximum edge weight equal to the axis weight *axis*

in the following. The concatenation of the paths p_i forms a cycle C of G that separates f from the outer face of G.

Lemma 3.2. *The weight of any proper face f of pg_μ in a clustered planar embedding Γ is the maximum axis weight of a boundary edge of the corresponding proper face f_μ of sg_μ.*

Proof. If Γ is a clustered planar embedding and C is the cycle formed by the concatenation of the axes p_i of the boundary edges around f_μ, then by Theorem 3.1, the maximum weight of an edge of C and therefore the maximum axis weight aw_{max} of a boundary edge cannot be less than the weight of f, otherwise C would be included in a cluster c of size aw_{max} and surround vertices that are not in c. On the other hand, the maximum weight of an edge e of C, contained in a path p_i, cannot exceed the weight of f, because then there would be a path on the boundary of f connecting v_i and w_i with maximum weight less then the weight of e, contradicting the definition of the axis weight. □

We associate the axis weights of the boundary edges of face f_μ with the corresponding pertinent nodes adjacent to μ in the SPQR-tree. Let aw_μ denote the axis weight of a node μ in the following, let max_μ be the maximum weight of an edge in pg_μ, and for a face f_μ in the skeleton let aw_{f_μ} denote the maximum axis weight of a boundary edge of f_μ.

The boundary edges of the outer face of the graph pg_μ can be split into two paths p_1^μ and p_2^μ connecting the poles of μ. We call these paths the *outer paths* of pg_μ. The weight max_μ cannot appear exclusively in the interior of the pertinent graphs of the nodes in the SPQR-tree, there has to be an edge of weight max_μ on one of the outer paths, otherwise a face of weight max_μ would be separated by the outer paths from the outer face. This fact will be used to obtain a unique normal form embedding in Section 4.

An outer path of pg_μ containing an edge of weight max_μ is called a *dominating outer path*. Figure 6 illustrates some of the terms and definitions from this section.

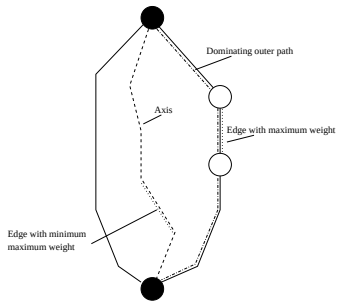


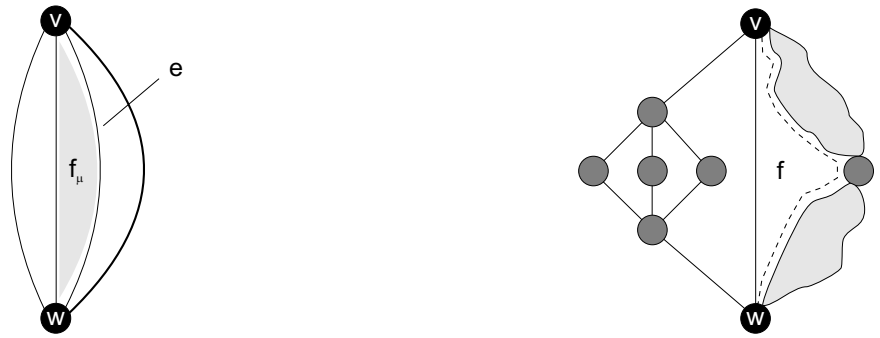
Figure 6: Illustration of an axis and an dominating outer path in a pertinent graph pg_μ.

4. Normal Forms of Clustered Planar Embeddings

In this section we develop a normal form of clustered planar embeddings. We use the idea to swap subgraphs at poles to transfer given clustered planar embeddings

into a corresponding normal form. We then show how to convert a clustered planar embedding into a normal form that is unique if we have determined a dominating outer path for each pertinent graph pg_μ. If both outer paths of a pertinent graph are dominating, we select one of them arbitrarily as *the* dominating outer path in the following. The main result will be that a clustered graph has a clustered planar embedding iff any of its normal form embeddings is clustered planar.

4.1. **Characterization of Normal Forms.** First, we discuss properties of clustered planar embeddings regarding the dominating outer paths of subgraphs, then we give a characterization of a normal form of clustered planar embeddings based on these properties.



(a) Skeleton of node μ (b) Pertinent graph of μ

Figure 7: The outer path (dashed line) of the pertinent graph of a skeleton edge e that lies on the boundary of a proper face f_μ of node μ is a part of the boundary of the corresponding face f in the pertinent graph of μ.

4.1.1. *Characteristics of clustered planar embeddings regarding the weight property.* Suppose f_μ is a proper face of the skeleton of a node μ in the SPQR-tree and f is the corresponding proper face of pg_μ. If e is a boundary edge of f_μ and ν is the pertinent node of e, then one of the outer paths of pg_ν is a subpath of the cycle of the boundary edges of f, see Figure 7. The dominating outer path of pg_ν can obviously only belong to f in a clustered planar embedding, if the weight of f, and therefore $aw_{f'_\mu}$, is at least as large as max_ν, which is a weight on the dominating outer path. Otherwise the dominating path would be separated and there would be a hole in the cluster. The reverse conclusion allows a swapping of pertinent graphs in the embedding:

Lemma 4.1. *Let Γ be a clustered planar embedding and let μ and ν be two nodes in the SPQR-tree, where μ is the parent of ν. Suppose an outer path of the pertinent graph pg_ν belongs to the boundary of a proper face f of pg_μ and the weight of f is at least max_ν. Then Γ can be converted into a clustered planar embedding Γ_r, such that the dominating outer path of pg_ν belongs to the boundary of f.*

Proof. The result is trivial if the dominating outer path of pg_ν already belongs to the boundary of f. Otherwise we want to swap pg_ν and show that the resulting embedding is still clustered planar. We assume that in Γ, the dominating outer path of pg_ν does not belong to the boundary of f but to the boundary of another face f' of G and Γ. Then f' also has a weight of at least max_ν. All faces that belong to pg_ν have a weight of at most max_ν. Let the sets F_i of faces be defined as in Theorem 1. Recall that by Theorem 1, each F_i is connected in the dual graph of G and Γ. Denote the set of proper faces in pg_ν that are in F_i by $F_{i,\nu}$. Then $F_{i,\nu}$ can only be non-empty if $i \leq max_\nu$. Since F_i is connected and f and f' are the only faces not belonging to pg_ν that share boundary edges with pg_ν , we can leave $F_{i,\nu}$ only through f or f', i.e., the connected components of $F_{i,\nu}$ are adjacent with f or f' (i.e., share boundary edges). When we swap pg_ν the connected components of $F_{i,\nu}$ adjacent with f become adjacent with f' and vice versa. Therefore F_i remains connected and the embedding Γ_r we obtain by swapping pg_ν is also a clustered planar embedding by Theorem 1 and the dominating outer path of pg_ν belongs to the boundary of f. □

We can use the swapping to construct an embedding that takes into account the dominating outer paths. For the P- and S-nodes of the SPQR-tree, we can even give a more specific characterization. First we discuss the S-node case.

Corollary 4.2. *Suppose μ is a S-node with children $\nu_1, \ldots, \nu_k$ in the SPQR-tree representing the clustered planar embedding Γ. Then Γ can be converted into a clustered planar embedding Γ_r such that the dominating outer path of pg_μ is the concatenation of the dominating outer paths of pg_{ν_i}.*

Proof. Let p be the dominating outer path of pg_μ and let p_i be the subpath of p in pg_{ν_i}. Let f be the face of G and Γ having p on its boundary. Note that f does not belong to pg_μ. The weight of f is at least max_μ. Each p_i belongs to the boundary of f and max_{ν_i} is at most max_μ and therefore at most the weight of f. Due to Lemma 4.1, either p_i ist the dominating path of pg_{ν_i} or we can swap pg_{ν_i}, replacing p_i by the dominating outer path of pg_{ν_i}. □

Next we consider the case of a P-node, which is slightly more difficult than the serial case. First we show that in a clustered planar embedding, there has to be a particular order of the parallel edges with respect to their axis and maximum weights, because there must not be any cycles of smaller weight around edges of higher weight. Therefore there may not be any local maximum of the axis weights besides the weights of the first and/or last edge in the edge sequence.

Lemma 4.3. *Let μ be a P-node with children $\nu_1, \ldots, \nu_k$ in the SPQR-tree representing the clustered embedding Γ, and let the skeleton edges e_i corresponding to the child nodes appear in the sequence $e_1, \ldots, e_k$ in Γ. Then the sequence of the axis weights of the e_i splits into a decreasing sequence $aw_{e_1}, \ldots, aw_{e_l}$ and an increasing sequence $aw_{e_{l+1}}, \ldots, aw_{e_k}$, where one of the sequences might be empty. Moreover, for $i \leq l$, $max_{e_i} \leq aw_{e_{i-1}}$ and, for $i > l$, $max_{e_i} \leq aw_{e_{i+1}}$.*

Proof. If the axis weights would not split into a decreasing and an increasing sequence, there would be positions $i < j < l$ such that $aw_{e_j} > aw_{e_i}$, aw_{e_l}. The concatenation of the axes of $pg_{pert(e_i)}$ and $pg_{pert(e_l)}$ would form a cycle of maximum edge weight $< aw_{e_j}$ that separates an edge and therefore also a face of weight

aw_{e_j} from the outer face, contradicting Theorem 3.1. To prove the second statement, we first consider the case $i < l$, i.e., e_i is not the last edge in the decreasing subsequence. If $max_{e_i} > aw_{e_{i-1}}$ then the axes of $pg_{pert(e_{i-1})}$ and $pg_{pert(e_{i+1})}$ which both have a maximum weight $< max_{e_i}$ separate an edge and therefore a face of weight $\geq max_{e_i}$ from the outer face contradicting Theorem 3.1. For symmetry reasons, we also have $max_{e_i} \leq aw_{e_{i+1}}$ for $i > l+1$. We consider the position l and observe that $max_{e_l} \leq aw_{e_{l+1}}$ or $max_{e_l} \leq aw_{e_{l-1}}$, for the same reasons as above. In the first case, we consider $aw_{e_1}, \ldots, aw_{e_{l-1}}$ as the decreasing and $aw_{e_l}, \ldots, aw_{e_k}$ as the increasing sequence. In the second case, we check whether $max_{e_{l+1}} \leq aw_{e_{l+1}}$. If this is not the case, for symmetry reasons, we put e_{l+1} into the decreasing sequence. □

Let $e_1, \ldots, e_k$ and l be as in the previous Lemma. Then $e_1, \ldots, e_l$ is called ***the*** *decreasing sequence* and $e_{l+1}, \ldots, e_k$ is called ***the*** *increasing sequence* of $e_1, \ldots, e_k$.

We now provide a method to convert the clustered planar embedding Γ into a clustered planar embedding Γ_r such that the maximum weight of the non-dominating outer path of pg_μ is minimized.

Assume that e_i and e_j belong both to the decreasing subsequence or both to the increasing subsequence. Then either $aw_{e_i} \leq max_{e_i} \leq aw_{e_j}$ or vice versa.

Now assume that e_i and e_j are any parallel edges in sg_μ. We say that e_i and e_j *overlap* if neither $aw_{e_i} \leq max_{e_i} \leq aw_{e_j}$ nor $aw_{e_j} \leq max_{e_j} \leq aw_{e_i}$. That means if e_i and e_j overlap then they cannot both belong to the increasing or the decreasing subsequence. The *overlap graph of* μ, denoted by O_μ, consists of the vertex set $\{e_1, \ldots, e_k\}$ and the edges (e_i, e_j) such that e_i and e_j overlap. Connected components of O_μ are called *overlap components* of μ. Observe that e_i and e_j belong to the same subsequence if they can be joined by a path of the overlap graph of even length and that e_i belongs to the decreasing and e_j belongs to the increasing subsequence (or vice versa) if they can be joined by an odd length path of the overlap graph, just because subsequence membership alternates along a path in an overlap component.

From our observations follows that if we fix for one edge e_i in a overlap component C of μ that it belongs to the increasing subsequence then we also know for each e_j of C whether it belongs to the increasing or to the decreasing subsequence of $e_1, \ldots, e_k$ independent of the particular clustered planar embedding. We extend the notions of maximum weight and axis weight to overlap components. Let max_C denote the maximum max_{e_i} with $e_i \in C$ and aw_C the minimum aw_{e_i} with $e_i \in C$.

Lemma 4.4. *If C and C' are different overlap components of μ then either $aw_C \leq max_C \leq aw_{C'}$ or $aw_{C'} \leq max_{C'} \leq aw_C$.*

Proof. Two edges e_i and e_j overlap if and only if the open intervals (aw_{e_i}, max_{e_i}) and (aw_{e_j}, max_{e_j}) intersect or if for one of them, say e_i, $aw_{e_i} = max_{e_i}, aw_{e_i} \in (aw_{e_j}, max_{e_j})$. For any overlap component C, the union of the intervals (aw_{e_i}, max_{e_i}) with $e_i \in C$ is (aw_C, max_C): The union of the intervals (aw_{e_i}, max_{e_i}) with $e_i \in C$ is a union of open intervals (a, b) with $a > aw_C$ and $b < max_C$. We only have to show that each $x \in (aw_C, max_C)$ belongs to some (aw_{e_i}, max_{e_i}) with $e_i \in C$. Assume this is not the case. Then we can partition the $e_i \in C$ into *small* e_i where $(aw_{e_i}, max_{e_i}) \subset (aw_C, x)$ and *large* e_i where $(aw_{e_i}, max_{e_i}) \subset (x, max_{e_i})$. No small e_i overlaps with a large e_j and there are small and large e_i. Therefore C is not

connected in the overlap graph which is a contradiction. As a consequence, the intervals (aw_C, max_C) and $(aw_{C'}, max_{C'})$ must be disjoint. If for one of the overlap components, say for C, $aw_C = max_C$, then $aw(C) \notin (aw_{C'}, max_{C'})$. This proves the Lemma. □

Lemma 4.5. *A clustered planar embedding Γ can be converted into a clustered planar embedding Γ' with the following property: If μ is a P-node with children $\nu_1, \ldots, \nu_k$ in the SPQR-tree representing Γ, and the subgraphs pg_{ν_i} appear in this sequence in Γ', then for each overlap component C of μ, the $e_i \in C$ with $max_{e_i} = max_C$ appears in the increasing subsequence of $e_1, \ldots, e_k$, where e_i is the skeleton edge corresponding to node ν_i.*

Proof. Let C be an overlap component. Assume that the e_i with $max_{e_i} = max_C$ belongs to the decreasing subsequence of $e_1, \ldots, e_k$. Let $C_{\leq}$ be the union of overlap components C' with $max_{C'} \leq max_C$ and let g_j be the largest index, such that $g_j \in C_{\leq}$. Note that $aw_{e_{i-1}}$ and $aw_{e_{j+1}}$ are at least max_C. Therefore also the weights of the face f_j consisting of an outer path of $pg_{pert(e_j)}$ and an outer path of $pg_{pert(e_{j+1})}$ and of the face f_{i-1} consisting of an outer path of $pg_{pert(e_{i-1})}$ and an outer path of $pg_{pert(e_i)}$ are at least max_C because the (axis) weight of the corresponding faces f_μ^j and f_μ^{i-1} in sg_μ are at least $aw_{e_{j+1}}$. We can view $C_{\leq}$ as a component G_μ and set $S_C := \cup_{e_i \in C_{\leq}} pg_{pert(e_i)}$ (the edges in the pertinent graphs of the skeleton edges in $C_{\leq}$). The outer paths of S_C are the outer paths of $pg_{pert(e_i)}$ and $pg_{pert(e_j)}$ that belong to f_{i-1} and f_j. The dominating outer path of C is the outer path belonging to $pg_{pert(e_i)}$. We can swap H_C and the resulting planar embedding is still clustered planar by Lemma 4.1. The maximum weight edge e_i of C now belongs to the increasing subsequence. We have to do this with every overlap component.

While swapping $C_{\leq}$, also all C' with $max(C') < max(C)$ are swapped. Therefore we start with the overlap component C with max_C maximal. If max_C belongs to the decreasing sequence we only reverse the enumeration $e_1, \ldots, e_k$. Then we put the overlap component with the second largest maximum weight into the right position as explained above, and we continue with the overlap component with the third largest maximum weight and so on. □

4.1.2. *The Normal Form Embedding.* Using the results from Section 4.1.1, we can state the following Theorem:

Theorem 4.6. *Each clustered planar embedding Γ can be converted into a clustered planar embedding Γ_{nf} that satifies the following conditions:*

Let μ be a node in the SPQR graph that represents the embedding Γ_{nf}.

1. *If μ is an S-node with children $\nu_1, \ldots, \nu_k$, then the dominating path of pg_μ is the concatenation of the dominating paths of the subgraphs pg_{ν_i}.*
2. *If μ is a P-node with children $\nu_1, \ldots, \nu_k$, where in Γ_{nf} the pg_{ν_i} appear in the given sequence from 1 to k, then either the maximum weight max_{ν_i} of each overlap component appears in the decreasing subsequence of $e_1, \ldots, e_k$ where ν_i is the pertinent node of skeleton edge e_i or the maximum weight weight of each overlap component appears in the increasing subsequence of $e_1, \ldots, e_k$.*
3. *If e is an skeleton edge of μ then one of the following two conditions is satisfied:*

- *e is an inner edge of sg_μ (i.e., the outer paths of $pg_{pert(e)}$ are not on an outer path of pg_μ), and for the proper faces f_1 and f_2 of pg_μ that contain the outer paths of $pg_{pert(e)}$ (i.e., the corresponding proper faces f_1' and f_2' in sg_μ share e as their boundary edge), the dominating outer path of $pg_{pert(e)}$ is at the boundary of the f_j that has the larger weight.*
- *e is not an inner edge of sg_μ (i.e., the outer path of pg_μ contains an outer path of $pg_{pert(e)}$), f is the proper face of pg_μ that shares an outer path with $pg_{pert(e)}$ (i.e., the corresponding proper face f' of sg_μ is the only proper face of sg_μ that has e as boundary edge), and the dominating outer path of $pg_{pert(e)}$ is on the boundary of f if and only if $max_e \leq weight(f)$.*

We call the embedding Γ_{nf} a *normal form embedding.*

4.2. **Construction of a Unique Normal Form Embedding.** In order to have a *unique* normal form embedding that is independent from the given particular embedding Γ, regardless whether it is clustered planar or not, we introduce the following rules that make our decisions well-defined:

(1) If both outer paths of pg_μ are dominating, we select one of them as *the* dominating outer path.
(2) We sort the faces of G with respect to their axis weight of the corresponding face in some skeleton graph to a sequence $f_1, \ldots, f_k$ and if we have to select one of the faces f_i and f_j of maximum weight, we select the one with maximum index i or j.
(3) We sort the nodes of the SPQR-tree in the first priority by their axis weight and in the second priority by their maximum weight to a sequence $\mu_1, \ldots, \mu_k$. Note that if μ is a P-node with skeleton edges $e_1, \ldots, e_k$, then we can construct a normal form embedding such that the decreasing/increasing subsequence is also decreasing/increasing with respect to the index in the sorted list.

We construct Γ_{nf} bottom up. The embedding of each Q-node is uniquely determined. For S- and R-nodes, the embeddings are uniquely determined up to reversal. For P-nodes, there is only one sequence $e_1, \ldots, e_k$ such that for each overlap component C, $max(C)$ appears in the increasing subsequence when the decision rules are applied. If the embedding of the skeleton of each child node ν of a node μ in the SPQR-tree is uniquely determined up to reversal, then the dominating outer path of each pg_ν is uniquely determined by our decision rules. Therefore also the embedding of pg_μ is uniquely determined up to reversal when we apply the decision rules.

Note that the construction of Γ_{nf} is independent of any particular embedding Γ. We can convert any embedding Γ to Γ_{nf} and if Γ is a clustered planar embedding, we can convert it to the clustered planar embedding Γ_{nf}. Vice versa, we can reverse the conversion, and if Γ is also a normal form embedding then clustered planarity is preserved. We therefore get the following result:

Theorem 4.7. *A graph G and a clustering C have a clustered planar embedding if and only if any normal form embedding of G and C is clustered planar.*

5. Clustered Planarity Testing Algorithm

In Sections 5 and 6 we describe the clustered planarity testing algorithm that is based on the concepts introduced in the previous sections. Section 5 states how the axis and maximum weights as well as the dominating outer paths and also the skeleton embeddings can be determined. Section 6 shows how to compute the normal form embedding and apply Theorem 3.1 efficiently. We first consider the case where the whole graph is biconnected and then give a sketch on how to derive an algorithm for the general case of connected graphs.

5.1. **Overview.** The main steps of the clustered planarity testing algorithm are as follows:

(1) First we compute an SPQR-tree for the given graph and determine the axis weights and the maximum weights of the skeleton edges, which are independent of the embeddings of the skeleton graphs.
(2) We determine embeddings for the skeleton graphs sg_μ of the SPQR-tree according to the rules of our normal form embedding.
(3) We determine the (axis) weights of the proper faces of each sg_μ depending on the computed embedding of the skeleton graphs.
(4) We determine the dominating outer paths of the skeleton graphs, which are projections of the dominating outer paths of the subgraphs pg_μ to sg_μ. Note that each path of pg_μ joining two vertices of sg_μ projects to a path of sg_μ.
(5) For each node μ with child node ν in the SPQR-tree, we determine how pg_ν has to be embedded relative to pg_μ to get a normal form embedding. This *relative embedding of μ and ν* depends only on the skeletons of μ and ν and the axis and maximum weights of their edges.
(6) We merge the relative embeddings to one embedding Γ that is a normal form embedding.
(7) We check that Γ is a clustered planar embedding using Theorem 3.1.

5.2. **Assignment of Axis and Maximum Weights for the Skeleton Edges.** The maximum weights can be assigned as follows: For each leaf-node (Q-node) μ with edge e, we just assign $max_\mu := weight(e)$. If μ is a node with children $\nu_1, \ldots, \nu_k$ in the SPQR-tree (not a leaf), then $max_\mu := max(max_{\nu_1}, \ldots, max_{\nu_k})$. This can be done in linear time with respect to the size of the derivation tree and therefore also in linear time with respect to the size of the graph G.

The axis weights are more difficult to compute. We use a minimum spanning tree with respect to the weights of the edges and take care that we do not lose our linear time bound. The edges of G can be divided into two classes depending on their position in the spanning tree and the class types can then be used to derive the axis weights.

First we compute a minimum spanning tree T_S of the whole graph G with respect to the edge weights in linear time using the following result:

Lemma 5.1. *Given a clustered graph $C = (G, T)$, a spanning tree T_S of G is a minimum spanning tree of G with respect to the edge weights if and only if for each cluster c of C, T_S restricted to c is a single tree (not a forest).*

Proof. First we show that any spanning tree where some cluster of C does not induce a subtree is not a minimum spanning tree, i.e., for every minimum spanning

tree, each cluster induces a subtree. Let S be T_S restricted to c. We assume that S is not a tree, i.e., S is not connected. Since G restricted to c is connected, we can find two connected components S_1 and S_2 of S, such that there is an edge (u, v) of G that joins a vertex u of S_1 with a vertex v of S_2. There is also a path p from a vertex u' in S_1 to a vertex v' in S_2 such that all inner vertices of p do belong to T_S but not to S_1 or S_2. Let (u', x) be the first edge of p. x is also not in c, otherwise x would belong to S_1. Therefore the weight of (u', x) is greater than the size of c. On the other hand, the weight of (u, v) is at most the size of c. Therefore, if we replace in T_S the edge (u', x) by (u, v), we still get a tree, and the sum of the weights decreases. Vice versa, we show that all spanning trees with the property that each cluster induces a subtree have the same weight sum. Note that T_S remains a tree if we shrink any cluster to a vertex, because any cluster induces a subtree. Let c be a cluster and $c_1, \ldots, c_k$ be its child clusters. Then the number of edges (u, v) of T_S , such that c is the smallest cluster containing u and v is $k-1$. The sum of the edge weights of these edges (u, v) is therefore $(k-1)|c|$. Let k_c be the number of child clusters of the cluster c. Then the weight sum of T_S is therefore $\sum_{c \in C} (k_c - 1)|c|$. This is true for every T_S such that each cluster $c \in C$ induces a subtree. □

Using Lemma 5.1, we determine T_S in linear time as follows:

For each cluster c, let E_c be the set of edges $e \in E$ such that c is the smallest cluster containing both end vertices of e. We contract the child cluster of c, i.e., we replace each edge $e = uv$ of E_c by $c_u c_v$ where c_u and c_v are the child clusters of c that contain u and v respectively. For each cluster c, we compute a spanning tree T_c of E_c. Note that all edges in E_c have the same weight. Now the tree T'_c arises from T_c by replacing each edge $c_u c_v$ of T_c by one edge uv. Then the union of all T'_c is a minimum spanning tree T_S, i.e., a spanning tree, such that each cluster induces a subtree.

Next we root T_S at the vertex x where x is one of the two vertices of the reference edge. Now consider any inner node μ of the SPQR-tree with poles u and v and pertinent graph pg_μ. We divide the inner nodes into two classes depending on the position of their poles within the spanning tree.

Introverted nodes are nodes whose poles are on a single path from the root to a leaf of the spanning tree, i.e., the parent of u or the parent of v in T_S is in pg_μ, see Figure 8.

Social nodes are nodes whose poles are in two different paths from the root to a leaf of T_S , i.e., the parents of u and of v in T_S are not in pg_μ.

Observe that for a social node μ the spanning tree T_S restricted to pg_μ splits into two trees T^u_μ with root u and T^v_μ with root v. Any path of pg_μ from u to v contains an edge e of G with one vertex in T^u_μ and the other vertex in T^v_μ. We call such an edge a *connecting edge* of pg_μ.

No we can easily derive the axis weights from the type of the node as follows:

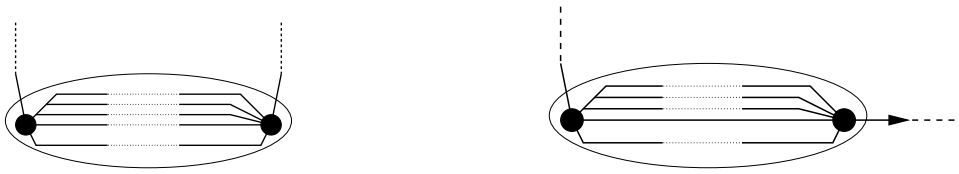
Lemma 5.2. *If μ is an introverted node of the SPQR-tree with poles u and v, then the axis weight aw_μ of μ is the maximum weight of an edge on the unique path from u to v in T_S. If μ is a social node, then aw_μ is the minimum weight of a connecting edge of pg_μ.*

Proof. Consider the smallest cluster c that contains u and v and that remains connected if we restrict it to pg_μ. We use the fact that aw_μ is the size of c. This follows from the following points:

- If there is a path from u to v in pg_μ with maximum weight w then u and v are in a connected component of a cluster c of size w restricted to pg_μ. Since clusters are connected in G and u and v separate pg_μ from the rest of the graph, G restricted to $pg_\mu \cup c$ is connected.
- If $pg_\mu \cup c$ is connected and contains u and v, then there is a path from u to v using only vertices in $pg_\mu \cup c$. Such a path has maximum weight $\leq |c|$.

From Lemma 5.1 we know that the unique path from u to v in T_S solely consists of vertices of c if μ is introverted. Therefore the maximum weight on this path is at most $|c|$, i.e., the axis weight aw_μ. If the maximum weight would be less, this path would contradict the definition of the axis weight.

If μ is social then the weight of any connecting edge of pg_μ cannot be smaller than the axis weight of μ. But then there has to be a connecting edge with weight aw_μ because aw_μ is the size of c. □



(a) Social node

(b) Introverted node

Figure 8: Location of the poles of introverted (right) and social (left) nodes in the spanning tree T_S, thick edges denote tree edges, dotted edges are not in the spanning tree.

To compute the axis weights efficiently, we proceed recursively, i.e., determine the axis weights of a node μ from the axis weights of the child nodes using Lemma 5.2. If we know the axis weights of all skeleton edges of μ, then we can derive the axis weight of μ in the same way as we determine the axis weight of μ from the weights of the edges in the pertinent graph pg_μ.

Lemma 5.3. *Let μ be a non-leaf node in the SPQR-tree with poles u and v.*

(1) *If μ is introverted then the skeleton edges of μ whose pertinent node is introverted form a tree T_μ with the same root as T_S restricted to pg_μ. The axis weight of μ, aw_μ, is the maximum axis weight of an edge on the unique path from u to v in T_μ.*

(2) *If μ is social then the introverted skeleton edges of μ form two trees T_μ^u and T_μ^v with u and v respectively as root. The axis weight of μ, aw_μ, is the minimum axis weight of a skeleton edge joining a vertex of T_μ^u and a vertex of T_μ^v (we call these edges also the* connecting edges *of the skeleton of μ).*

Proof. Let T_μ be the set of introverted edges of sg_μ and let x be a vertex of sg_μ. There is exactly one edge (x, y) of T_S, such that y is the T_S-parent of x. There is at most one edge e_x of sg_μ, such that (x, y) is in $pg_{pert(e_x)}$. Moreover, if x is not u or v, there exists such an edge e_x and this edge is introverted. Furthermore, each introverted edge of sg_μ is such an e_x for some x. Let x and p_x be the vertices incident with e_x, i.e., nodes in sg_μ. Then we can consider p_x as the parent of x in T_μ and p_x is an ancestor of x in T_S. Therefore T_μ has no cycle and is a forest. If μ is social then T_μ splits into two trees, otherwise μ is introverted and T_μ is a tree. Suppose μ is introverted. Then each edge on the unique path from u to v in T_S is in some $pg_{pert(e')}$, such that e' is on the unique path from u to v in sg_μ. This is in particular true for the maximum weight edge e on the unique path from u to v in T_S. By Lemma 5.2, the edge e' of sg_μ such that $pg_{pert(e')}$ contains e has the same axis weight as the weight of e. Therefore the axis weight of μ is the maximum axis weight of an edge of T_μ on the unique path from u to v. Now suppose μ is social and let e be a connecting edge of pg_μ. Note that also the edge e' of sg_μ, such that $pg_{pert(e')}$ contains e is social. Moreover, all connecting edges of $pg_{pert(e')}$ are also connecting edges of pg_μ. If e is a connecting edge of pg_μ of minimum weight, then e is also a connecting edge of $pg_{pert(e')}$ of minimum weight. On the other hand, if e' joins a vertex of T^u_μ and of T^v_μ, then also e' is social, and all connecting edges of $pg_{pert(e')}$ are connecting edges of pg_μ. Each connecting edge of such $pg_{pert(e')}$ therefore has a weight that is at least the axis weight of μ. Therefore the axis weight of μ is the minimum over the axis weights of edges e' of sg_μ connecting a vertex of T^u_μ with a vertex of T^v_μ. □

We can efficiently check if a node μ with poles u and v is introverted, by checking that one of the Q-nodes, that correspond to edges connecting u and v with its respective parent in T_S , is a descendant of μ in the SPQR-tree. This can be done in linear time with a simple preprocessing of the SPQR-tree. Let μ be introverted. Note that for the components $pg_{pert(e)}$ that correspond to skeleton edges e that are passed by the unique path from u to v the nodes $pert(e)$ are also introverted. We determine a subforest T_I of the SPQR-tree consisting of the introverted nodes. The children of μ in T_I are pertinent nodes of the edges in T_μ that are on the unique path from u to v in T_μ. The axis weight of μ is the maximum axis weight of a child of μ in T_I.

If μ is social, then we determine the subforest T_{Soc} of the SPQR-tree consisting of the social nodes. The children of μ in T_{Soc} are the nodes with one pole vertex in T^u_μ and the other one in T^v_μ, i.e., the connecting edges of μ. The axis weight of μ is the minimum axis weight of a child of μ in T_{Soc}.

Following Lemma 5.3, we get the axis weight of μ by determining the axis weight of the children of μ in T_I or T_{Soc} and taking the maximum or minimum, respectively. This can be done in linear time. We therefore have

Proposition 5.4. *The axis weights of the skeleton edges can be computed in linear time.*

5.3. **Computation of Embeddings for the Skeleton Graphs.** We need to compute embeddings for the skeleton graphs of the S, P, and R-nodes of the SPQR-tree. For the rigid case in R-nodes there is a unique combinatorial embedding up to reversal. The combinatorial embedding of the path for the sequential case in S-nodes is unique, too. For the skeleton graphs of P-nodes we have to find a normal

form embedding, i.e., a sequence $e_1, \ldots, e_k$ of the parallel edges that splits into a decreasing and an increasing sequence that satisfy the conditions of Theorem 2. The strategy to compute an embedding for the skeleton of a P-node μ is as follows:

(1) Instead of computing the whole overlap graph of μ we only compute a spanning forest O'_μ. We then also know the overlap components.
(2) For each edge in sg_μ we determine whether it belongs to the decreasing or the increasing subsequence. For each overlap component C, we select a e_C with maximum maximum weight and put it into the increasing sequence.
(3) We know for each overlap component the elements of the decreasing subsequence and of the increasing subsequence and can therefore derive the sequence $e_1, \ldots, e_k$ which gives us an embedding of the skeleton graph.

5.3.1. *A Spanning Forest of the Overlap Graph.* Remember that the union of the intervals (aw_{e_i}, max_{e_i}) of an overlap component C forms an open interval (aw_C, max_C). Therefore if $aw_C < a < max_C$ then there is an $e \in C$ with $aw_e < a < max_e$. If some $e' \in C$ has smaller axis weight than e, then $aw_C < aw_{e'}$. On the other hand, if aw_e overlaps with some e'' then $aw_e < max_{e''}$ and therefore $aw_e < max_C$. It follows that if there is a e' with smaller axis weight than e in the same overlap component C, then $aw_C < aw_e < max_C$. We then can conclude that there is a $e''' \in C$ with $aw_{e'''} < aw_e < max_{e'''}$, i.e., e''' overlaps with e and has smaller axis weight than e. Hence we have the following:

Lemma 5.5. *If the overlap component containing edge e also contains an edge e' with smaller axis weight than e, then e overlaps with an e' with smaller axis weight than e.*

We know now, that if an e overlaps with an e' of smaller axis weight, we can use such an e' as parent of e in the spanning forest of the overlap graph. But this does not determine the whole spanning forest. We only know that if e has no parent, then $aw_e = aw_C$ where C is the overlap component containing e. There might be several edges e_i in C with $aw_{e_i} = aw_C$. Observe that if $aw_C < max_C$ such an e_i can only belong to C iff $aw_{e_i} < max_{e_i}$. If $aw_{e_i} = max_{e_i}$ its maximum weight is less than or equal to any axis weight of edges on C. For the backward direction, the intersection of the intervals (aw_{e_i}, max_{e_i}) and (aw_C, max_C) is non-empty and we can pick any x of the intersection and any $e' \in C$ with $aw_{e'} < x < max_{e'}$ so that e and e' overlap.

Observe also in general that if $aw_e = aw_{e'} < max_{e'} \leq max_e$ then e and e' overlap. If there is a e' with $aw_{e'} = aw_e < max_{e'} < max_e$ we can take such a e' as parent of e.

We now have the following situation: If e and e' belong to the same overlap component and have no parent then they coincide in the weight and in the maximum weight and the axis weight is smaller than the maximum weight. Therefore if the parent of e is not defined but $max_e > aw_e$ then we have to check whether there is another e' with $aw_{e'} = aw_e$ and $max_{e'} = max_e$. We consider e and e' as equivalent if they conincide in max and aw and the axis and the maximum value are different.

We observe that we can order the equivalent edges e in any way, and just the predecessor of edge e can be taken as parent of e. We use the observations above and proceed as follows:

(1) We sort the skeleton edges of μ in the first priority by their axis weight and in the second priority by their maximum weights to a sequence $e_1, \ldots, e_k$. This can be done in linear time by bucket sort.
(2) If there is a e_j with $j < i$ and $max_{e_j} > aw_{e_i}$ then we can take such e_j as parent $par(e_i)$ of e_i. To do this efficiently, we proceed as follows:
   (a) We determine for each $i > 1$, $max'(e_i) := max_{j<i}(max_{e_j})$, i.e., the maximum max value of a e_j with index smaller then i.
   (b) For each maximum weight m, we determine the smallest index $i(m)$, such that $max_{e_{i(m)}} = m$. This can be done in linear time by bucket sort.
   (c) For each e_i with $max'(e_i) > aw_{e_i}$ we assign $par(e_i) := e_{i(max'(e_i))}$, i.e., the parent of e_i is an e_j with $max(e_j) = max'(e_i)$ of smallest index j.

Our main result is now

Proposition 5.6. *Spanning forests of the overlap graphs O_μ of μ can be determined in linear time.*

5.3.2. *The partition of overlap components into decreasing and increasing subsequence.* First we check whether an edge e_i of sg_μ belongs to the increasing or the decreasing subsequence of the skeleton edges $e_1, \ldots, e_k$. Note that overlap components and nodes of O'_μwithout a parent are in a 1-1-correspondence. We proceed as follows:

(1) We determine for each overlap component C of μ, max_C and a $e_C \in C$ with $max_{e_C} = max_C$. That means, we determine for each e of sg_μ that has no parent in O'_μ, the number max_C of that C with $e \in C$, i.e., the maximum $max_{e'}$ of a descendent e' of e in O'_μ (including e). For this C, we determine e_C. If e has proper descendants, i.e., C has more than one element, this e_C is the e_i of minimum index such that $max_{e_i} = max_C$, which means $e_C = i(max_C)$. If e has no proper descendants then e_C is just e.
(2) To check whether e belongs to the increasing or the decreasing sequence, we check if it has an odd or even distance from some e_C in the spanning forest O'_μ.

We know now whether an edge e in sg_μ belongs to the decreasing or the increasing subsequence of $e_1, \ldots, e_k$, but we do not know the position of e in $e_1, \ldots, e_k$ yet.

We use our sorting $e_1, \ldots, e_k$ with respect to the axis and maximum weights from Section 5.3.1 as follows: Let $F_1(e_i) = 1$ and $F_2(e_i) = 0$ if e_i belongs to the decreasing subsequence and $F_1(e_i) = 0$ and $F_2(e_i) = 1$ if e_i belongs to the increasing subsequence. This e_i is the l^{th} element of the decreasing subsequence if $\sum_{j\geq i}^{F1(e_j)} = l$ and e_i belongs to the decreasing subsequence, and it it the l^{th} element of the increasing subsequence if $\sum_{j\leq i}^{F2(e_j)} = l$ and it belongs to the increasing subsequence.

The final sequence $e_1, \ldots, e_k$ is then just the concatenation of the decreasing and the increasing subsequence.

All steps can be done in linear time. We can now state the main result of this subsection:

Proposition 5.7. *For all P-nodes, the embeddings of the skeleton edges represented by the sequence $e_1, \ldots, e_k$ can be computed in linear time.*

5.4. **Determining the Dominating Outer Paths of the Skeleton Graphs.** We assume that μ is a P- or R-node in the SPQR-tree, so that the embedding of the skeleton is now fixed up to reversal (Remember that in an S-node, the skeleton consists simply of a path between the poles representing the serial structure, and an additional edge connecting the poles). Let u and v be the poles of μ. Note that any path p from u to v in pg_μ projects to a path p' from u to v in sg_μ. This path p' contains just those edges e_i in the skeleton such that $pg_{pert(e_i)}$ contains at least one edge of p. The outer paths of pg_μ project to *outer paths of the skeleton of* μ, i.e., the two paths from u to v in the skeleton that form the outer cycle of the skeleton. We would like to find the outer path of the skeleton that is the projection of the dominating outer path of pg_μ to the skeleton.

Lemma 5.8. *Let a clustered planar embedding Γ of the graph G and a corresponding SPQR-tree be given. If there is a proper face f_μ in sg_μ with axis weight $aw_{f_\mu} = max_\mu$ then there is an edge e on an outer path of the skeleton with $aw_e = max_\mu$.*

Proof. The proper face f in pg_μ that is the corresponding face to f_μ also has weight max_μ. Let C be the concatenation of the axes of the edges e' that belong to the outer cycle of sg_μ. One of the edges of C must have weight max_μ , because otherwise a face of this weight is separated by a cycle of smaller weight from the outer face. Therefore one of the edges on the outer cycle of sg_μ must have axis weight max_μ. □

The *dominating outer path* of a skeleton graph sg_μ is the projection of the dominating outer path of pg_μ into sg_μ, i.e., an edge e of sg_μ belongs to the dominating outer path of sg_μ if and only if $pg_{pert(e)}$ contains edges of the dominating outer path of pg_μ.

Lemma 5.9. *A dominating outer path of sg_μ can be determined as follows:*

1. *If sg_μ contains a proper face f_μ with weight $aw_{f_\mu} = max_\mu$ then an outer path of sg_μ containing an edge e with $aw_e = max_\mu$ can be selected as the dominating outer path.*
2. *If sg_μ does not contain a proper face f with weight $aw_{f_\mu} = max_\mu$ then an outer path of sg_μ containing an edge e with $max_{pert(e)} = max_\mu$ can be selected as the dominating outer path.*

Proof. We have to show that the selected outer path of sg_μ is the projection of a dominating outer path of pg_μ into sg_μ.

If sg_μ contains a proper face of weight max_μ then Lemma 5.8 applies, i.e., there is an edge e on one of the outer paths p' of sg_μ, such that $aw_e = max_\mu$. This path p' is the projection of an outer path p of pg_μ that also passes $pg_{pert(e)}$. If u and v are the endnodes of e then p restricted to $pg_{pert(e)}$ is a path from u to v in $pg_{pert(e)}$. Since $aw_e = max_\mu$, p restricted to $pg_{pert(e)}$ passes an edge of weight $\geq max_\mu$. Since $max_{pert(e)} \leq max_\mu$, such an edge must have weight max_μ.

If sg_μ does not contain a proper face of weight max_μ then we consider any edge e of pg_μ on an outer path of pg_μ with weight max_μ. This edge e belongs to some $pg_{pert(e')}$ with e' belonging to sg_μ. Such a e' belongs to one of the outer paths of sg_μ, and $max_{pert(e')} = max_\mu$. Now consider any e' in sg_μ that belongs to anouter path with $max_{pert(e')} = max_\mu$. Let $f_\mu$be the proper face of sg_μthat contains e' and let f be the corresponding face of pg_μ. Since the weight of $f_\mu$and therefore the weight of f are smaller than max_μ, the dominating outer path of $pg_{pert(e')}$ (which contains

an edge of $max_\mu = max_{pert(e')}$) is not a subpath of the boundary of f. Therefore the dominating path of $pg_{pert(e')}$ is a subpath of a dominating path of pg_μ. This means that e' is an edge of the projection of a dominating path of pg_μ into sg_μ. Therefore the outer path of sg_μ containing e' can be taken as dominating outer path of sg_μ. □

The dominating outer path of sg_μ now can be determined as follows:

(1) We determine the outer paths of sg_μ. Note that a planar embedding of sg_μ together with the edge connecting the poles is known. The two faces containing this edge determine the outer paths of sg_μ.
(2) We check whether there is a proper face of sg_μ with weight max_μ. Note that each face is determined by the sequence of its boundary edges. The maximum axis weight of a boundary edge of any face can hence be determined in linear time.
(3) We proceed as in Lemma 5.9 to determine the dominating outer paths. Since we know the face weights and the maximum and axis weights of the edges in sg_μ, we can determine the maximum axis or maximum weight of each outer path in linear time and hence perform this step also in linear time.

As a result of this subsection, we get the following:

Proposition 5.10. *If μ is a P- or R-node then a dominating path of sg_μ can be determined in linear time.*

6. Computation of the Clustered Planar Embedding

In this section we show how the final clustered planar embedding can be computed. Our final embedding has to satisfy the requirements of a normal form embedding. We can assume from the results of the previous sections that the embedding of each sg_μ in the SPQR-tree is known. If μ is an S- or an R-node, we are done, in the case of a P-node we compute the decreasing and increasing subsequence of the parallel edges and sort both sets with respect to the axis weight. We also know the dominating paths, i.e., we know if the left or right outer path of sg_μ is dominating.

Now we have to find the relative orientation of a child component sg_ν of sg_μ. Note that the axis in the pertinent graph of a skeleton edge e splits the pertinent graph into two parts. We have to guarantee that for each face f incident with e, for the part that in the final embedding will be turned in direction of f, its maximum weight is at most the axis weight of f. Otherwise the part P_{max} with the higher maximum weight would be enclosed by a circle with edge weights bounded by the axis weight of f. We therefore have to check if we have to swap the embedding of $sg_{pert(e)}$. This can be done by first computing the relative orientation of a child component within its parent component by a bottom-up traversal of the SPQR-tree, followed by a top-down traversal that uses the relative orientations to compute the absolute orientation as follows:

For each node ν in the SPQR-tree, we store an orientation value $or(\nu)$ that specifies the relative embedding of a child component sg_ν within its parent component sg_μ. If e is an inner edge of sg_μ then we have to swap $sg_{pert(e)}$ if the part P_{max} is directed to the face of smaller axis weight. If e is an outer edge, and μ is not an S-node, then $sg_{pert(e)}$ has to be swapped if the incident face f different from the

outer face has smaller axis weight than the weight of P_{max} and P_{max} is directed to this face or f has an axis weight that is at least the maximum weight in P_{max} and P_{max} is directed to the outer face of sg_μ. If μ is an S-node, the parts with the larger weight are directed in the same direction.

Now we know for each child component its relative embedding within its parent component. We can derive the absolute orientation $abs(\mu)$ just by checking how often a component would be swapped either directly or by swapping an ancestor component. If there is an odd number of swaps, the component needs to be swapped in order to achieve the correct final orientation. We therefore only need to compute

$$abs(\mu) := \Pi_{(\nu \text{ ancestor of } \mu \text{ in } SPQR\text{-tree}) \vee (\nu=\mu)} \; or(\nu)$$

which can be done in linear time by a top-down traversal.

Let emb_μ be the original embedding of sg_μ. Then

$$emb_\mu^{final} := \begin{cases} emb_\mu, & \text{if } abs(\mu) = 1 \\ \text{reversal of } emb_\mu, & \text{if } abs(\mu) = -1 \end{cases}$$

The final embedding emb^{final} of G can be defined as follows: We determine the embedding emb_μ^{pre} of pg_μ recursively knowing the embeddings $emb_{pert(e_i)}^{pre}$ of the $pg_{pert(e_i)}$ for all e_i in sg_μ and the embedding emb_μ^{final} of sg_μ.

emb_μ^{pre} is determined as follows: For a skeleton edge e in sg_μ with incident vertices u and v in emb_μ^{final}, in the clockwise enumeration of the incident edges of u and v in sg_μ, e is replaced by the edges of $pg_{pert(e)}$ that are incident with u, and with v, respectively. They appear in emb_μ^{pre} in the same order as in $pg_{pert(e)}$. After processing all nodes in the SPQR-tree, we know the embedding of the root node and therefore we can derive the embedding for the graph G. We finally insert the root edge connecting the poles u and v of the root node between the first and last edge of u and v, respectively, and get an embedding for the input graph G that satisfies the requirements of a normal form.

6.1. **How to Derive the Normal Form Efficiently.** If v is a vertex in G, a node in the SPQR-tree $\mathcal{T}$ whose skeleton contains v is called an *allocation node* of v. The allocation nodes of v form a subtree T_v of T and if v is not a pole of the root of the SPQR-tree, then the root of T_v is the only node μ of T_v such that v is not a pole of μ, otherwise the root of T_v is the root of $\mathcal{T}$. Note the the number of roots of the trees T_v is equal to the number of vertices in G. The number of non-root nodes is also linear in the size of G, because each node μ appears as non-root node in at most two trees T_u and T_v where u and v are the poles of μ, and the size of the SPQR-tree of a planar graph is linear in the number of nodes of the graph. We can therefore determine the trees T_v by a traversal of the SPQR-tree in linear time.

We use the trees T_v to efficiently determine the final embedding emb^{final} of G. We may assume that the embeddings emb_μ^{pre} are known, where emb_μ^{pre} is the projection of a normal form embedding emb^{final} of G into sg_μ.

We call a vertex v an *inner vertex* of sg_μ, if v belongs to sg_μ, but is not a pole of μ. For each inner vertex v of sg_μ, emb_μ^{pre} determines the cyclic enumeration of the edges incident with v in sg_μ. For each pole of μ, we have a cyclic enumeration of the edges incident with v where the first and last edge are fixed. Therefore we

have for each node μ of T_v an enumeration of the children of μ that corresponds to the clockwise enumeration of the edges of sg_μ incident with v.

The leaves of T_v are just the Q-nodes representing edges that are incident with v (except for the special case if v is a pole of the root of the SPQR-tree). We determine a postorder of the nodes in T_v and restrict this order to the leaves of T_v (plus the root if necessary). This gives us exactly the enumeration of edges incident with v as in emb^{final}, because if $e_1, \ldots, e_k$ is the enumeration of edges incident with v in sg_μ that corresponds to the embedding emb_μ^{pre} then in emb^{final} the edges of $pg_{pert(e_1)}$ are enumerated first, the descendants of $pg_{pert(e_2)}$ second and so on, which is also true for the post order enumeration of T_v.

Therefore we get the following result:

Theorem 6.1. *The normal form embedding emb^{final} of G can be determined in linear time if the axis and maximum edge weights and the SPQR-tree are known.*

Proof. A postorder traversal can be done in linear time. Selecting and renumbering the leaves can be done in the same time bounds (for each leaf of T_v, determine the number of leaves of T_v with a smaller postorder number).

□

6.2. **Checking Clustered Planarity of the Final Embedding.** At this point we have an embedding emb^{final} of G that is a normal form embedding. We know that this embedding is a clustered planar embedding if and only if G has a clustered planar embedding. We check this using Theorem 3.1 as follows:

We determine a spanning tree T_F of the dual graph of G such that T_F restricted to the sets F_i (see Section 3) is a spanning tree of (F_i, E_i) for each i. The existence of such a spanning tree is equivalent to the statement that any (F_i, E_i) is connected. Therefore, by Theorem 3.1, the fact that T_F restricted to F_i is a spanning tree of (F_i, T_i) for each i, is equivalent to the statement that emb^{final} is a clustered planar embedding.

Let $F_{=i}$ be the set of faces of weight i and $E_{=i}$ be the set of edges of weight i. Clearly, if i is the maximum weight of an edge of G, then $F_{=i} = F_i$ and $E_{=i} = E_i$ and, in a clustered planar embedding, such a $(F_{=i}, E_{=i})$ therefore has to be connected. If i is not the maximum weight, then in a clustered planar embedding for each connected component C of $(F_{=i}, E_{=i})$ there is an $f \in F_{=i}$ and an $f_1 \in F_{i+1}$, such that f and f_1 share an edge of weight i. If there is no such edge of weight $\geq i$, then (F_i, E_i) would not be connected. As the maximum weight of an edge at the boundary of f is i, the weight must be exactly i. We compute the spanning tree T_F as follows:

(1) Determine the connected components of $(F_{=i}, E_{=i})$ and the spanning trees T_C for each of these components.
(2) For each connected component C of some $(F_{=i}, E_{=i})$ with i not maximum edge weight, try to determine an $f_C \in C$ and an $f_{1,C}$ of weight $> i$, such that f_C and $f_{1,C}$ share an edge e_C of weight i.
(3) T_F consists of the edges of the trees T_C and the edges e_C.

The computation steps obviously can be done in linear time. Regarding the correctness, we show the following:

Proposition 6.2. *T_F is a tree if and only if the given embedding emb^{final} is a clustered planar embedding.*

Proof. Using Theorem 3.1, we show that T_F is a tree iff for each i, (F_i, E_i) is connected. We use the following auxiliary result.

Lemma 6.3. *T_F is a tree if and only if for each nonmaximum weight i and each connected component C of $(F_{=i}, E_{=i})$, the tree edge e_C of weight i and with incident faces f_C in C and $f_{1,C}$ with weight greater than i exists and for the maximum edge weight i in G, $(F_{=i}, E_{=i})$ is connected.*

Proof. T_F restricted to each component C of some $(F_{=i}, E_{=i})$ is a tree. therefore T_F is a tree if and only if T_F remains a tree if we contract each such component C to a node. After contraction of these connected components, T_F is a forest and only the edges e_C remain. Therefore T_F is a tree iff for all but one component C, the edge e_C is defined. For components C of $(F_{=i}, E_{=i})$ with maximum weight, e_C is not defined. Therefore T_F is a tree iff there is exactly one connected component of the $(F_{=i}, E_{=i})$ of maximum weight and for each nonmaximum weight i and each connected component C of $(F_{=i}, E_{=i})$, e_C is defined. □

Now we assume that T_F is a tree. We root T_F as follows: For each nonmaximum weight i in G and each connected component C of $(F_{=i}, E_{=i})$, we take f_C as root of T_F restricted to C. The parent of f_C is $f_{1,C}$. For the maximum weight i we take any $f \in F_{=i}$ as root. The weight of the parent of f is at least the weight of f and the weight of the edge joining f and its parent is the weight of f. Therefore T_F restricted to F_i is a spanning tree of (F_i, E_i) and therefore (F_i, E_i) is connected for each i. Vice versa, if for each i, (F_i, E_i) is connected then we know from the discussion above that for each connected component C of $(F_{=i}, E_{=i})$, there is an edge e_C with weight i at the boundary of some face $f \in C$ and of another face f' of weight $> i$. We chose one such edge as edge of T_F and obtain a tree. □

6.3. **Complexity Analysis.** An SPQR-tree for a given biconnected graph can be constructed in linear time [11]. The recursive procedure to compute the maximum weights needs linear time with respect to the size of the SPQR-tree and therefore also linear time with respect to the size of the graph G. From Proposition 5.4 we know that the axis weights can be determined in linear time.

The computation of the embeddings of the skeleton graphs can be done in linear time. For the P-nodes we need to sort the subsequences of the parallel edges, but as they are sorted by their axis weights, this can be done in linear time, too. The face weights are determined by the computed axis weights and the computation of the dominating paths can be done in linear time by Proposition 5.10. After obtaining a normal form embedding, we can check in linear time, if it is clustered planar, which is true if and only if the clustered graph is clustered planar.

We get the following main result:

Theorem 6.4. *If G is a biconnected graph then we can check in linear time whether $C = (G, T)$ has a clustered planar embedding and we can construct such a clustered planar embedding within the same time bound.*

6.4. **Extension to Connected Graphs.** We give a sketch of how to extend the algorithm to the case of connected graphs. A connected, but not biconnected graph G can be decomposed into its biconnected components in linear time [12, 15].

We select in G an edge e of maximum weight and consider the block b_r containing e as root of B_T. This defines, for every other block and each vertex b, a parent $P(b)$ of b in B_T.

The algorithmic idea now is as follows:

(1) We determine clustered planar embeddings emb_b of each of the blocks b, such that the parent $P(b)$, which has to be a vertex, appears at the outer face.
(2) Let v be the parent of a block b and b' the parent of v, i.e., v separates b and b'. We embed b into a face f of b' that contains v as boundary vertex and has a weight that is at least as large as the maximum weight of an edge in b or a descendant of b.

To guarantee that the weight of a face into which a block is embedded is large enough, we redefine the maximum weights.

(1) We define weights of vertices. Note that the decision to embed a block b into a certain face f means that all descendants of b in the block tree are enclosed by the face boundary of f, too. The weight of a vertex v is therefore defined as the maximum weight of an edge that appears in a block that is a descendant of v in B_T.
(2) Let μ be a node in the SPQR-tree of block b. The axis weight is defined as before. The maximum weight of μ is the maximum weight of an edge of pg_μ or an *inner vertex* of pg_μ (i.e., a vertex different from the poles of μ).
(3) If max_μ is the weight of an edge of pg_μ then the dominating path of pg_μ is the outer path containing such an edge. If max_μ is instead the weight of an inner vertex of pg_μ , then the dominating outer path is one that contains an inner vertex of pg_μ of maximum weight.

We can determine max_μ in the same recursive manner as in the biconnected case. We determine the weights of the vertices and the maximum weights of the skeleton edges and select the maximum of these weights as max_μ.

The next step is to determine the normal form embeddings of all the blocks b. It is obvious that each vertex v appears as boundary vertex of some face of b that has a weight that is at least as large as the weight of v. In particular, this is true for the parent $P(b)$ of block b. The weight of $P(b)$ is at least the maximum over all vertex and edge weights of b. Therefore $P(b)$ appears at a face of maximum weight of b, this face can then be taken as outer face of b.

After determining the embeddings of all blocks, we embed each block b into the face of the parent of $P(b)$ of largest weight that has the vertex $P(b)$ at its boundary. This defines the final embedding. The time bound is the same as in the case of biconnected components.

7. Future work

The question how to decide c-planarity of general, possibly non-c-connected clustered graphs is still open. If the underlying graph is connected, we can still use the concept of faceweights, but we have to take care about edge-region crossings.

References

1. G. Di Battista, W. Didimo, and A. Marcandalli, *Planarization of clustered graphs*, vol. 2265, 2002, pp. 60–74.
2. G. Di Battista and R. Tamassia, *On-line planarity testing*, SIAM J. Comput. **25** (1996), no. 5, 956–997.
3. S. Cornelsen and D. Wagner, *Completely connected clustered graphs*, WG: Graph-Theoretic Concepts in Computer Science, International Workshop WG, 2003.

4. P. F. Cortese, G. Di Battista, M. Patrignani, and M. Pizzonia, *Clustering cycles into cycles of clusters (extended abstract)*, Proc. of The 12th Int. Symposium on Graph Drawing (GD 2004), 2004.
5. E. Dahlhaus, *A linear time algorithm to recognize clustered planar graphs and its parallelization*, LATIN: Latin American Symposium on Theoretical Informatics, 1998.
6. R. Diestel, *Graph theory*, third ed., Graduate Texts in Mathematics, vol. 173, Springer, 2005.
7. Peter Eades, Qing-Wen Feng, and Robert F. Cohen, *Clustered graphs and C-planarity*, Tech. report, March 22 1995.
8. Q. W. Feng, R. F. Cohen, and P. Eades, *Planarity for clustered graphs*, ESA: Annual European Symposium on Algorithms, 1995.
9. M. T. Goodrich, G. S. Lueker, and J. Z. Sun, *C-planarity of extrovert clustered graphs*, Proc. of The 13th Int. Symposium on Graph Drawing (GD 2005) (Limerick, Ireland), vol. 3843, September 2005.
10. Gutwenger, Junger, Leipert, Mutzel, Percan, and Weiskircher, *Advances in C-planarity testing of clustered graphs*, Graph Drawing: Proc. Graph Drawing (GD), 2002.
11. C. Gutwenger and P. Mutzel, *A linear time implementation of SPQR-trees*, Proc. of The 8th Int. Symposium on Graph Drawing (GD 2000), 2000.
12. J. E. Hopcroft and R. E. Tarjan, *Efficient algorithms for graph manipulation*, Commun. ACM **1** (1973), no. 6, 372–378.
13. T. Lengauer, *Hierarchical planary testing algorithms*, ICALP, 1986, pp. 215–225.
14. P. Mutzel and R. Weiskircher, *Optimizing over all combinatorial embeddings of a planar graph*, IPCO: 7th Integer Programming and Combinatorial Optimization Conference, 1999.
15. R. E. Tarjan, *Depth first search and linear graph algorithms*, SIAM J. Comput. **1** (1972), no. 2, 146–160.

(E. Dahlhaus) Technische Universität Darmstadt
E-mail address, E. Dahlhaus: elias.dahlhaus@bahn.de

(K. Klein, P. Mutzel) University of Dortmund, Germany
E-mail address, K. Klein: karsten.klein@cs.uni-dortmund.de

E-mail address, P. Mutzel: petra.mutzel@cs.uni-dortmund.de