Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 12, no. 2, pp. 225–262 (2008)

C-Planarity of C-Connected Clustered Graphs

Pier Francesco Cortese[1] *Giuseppe Di Battista*[1] *Fabrizio Frati*[1]
Maurizio Patrignani[1] *Maurizio Pizzonia*[1]

[1]Dipartimento di Informatica e Automazione
Roma Tre University, Italy

Abstract

We present the first characterization of c-planarity for c-connected clustered graphs. The characterization is based on the interplay between the hierarchy of the clusters and the hierarchies of the triconnected and biconnected components of the underlying graph. Based on such a characterization, we provide a linear-time c-planarity testing and embedding algorithm for c-connected clustered graphs. The algorithm is reasonably easy to implement, since it exploits as building blocks simple algorithmic tools like the computation of lowest common ancestors, minimum and maximum spanning trees, and counting sorts. It also makes use of well-known data structures as SPQR-trees and BC-trees. If the test fails, the algorithm identifies a structural element responsible for the non-c-planarity of the input clustered graph.

Submitted:	Accepted:	Final:	Published:
August 2006	August 2008	August 2008	October 2008

Article type:	Communicated by:
Regular Paper	M. Goodrich

Work partially supported by MUR under Project "MAINSTREAM: Algoritmi per strutture informative di grandi dimensioni e data streams."

E-mail addresses: cortese@dia.uniroma3.it (Pier Francesco Cortese) gdb@dia.uniroma3.it (Giuseppe Di Battista) frati@dia.uniroma3.it (Fabrizio Frati) patrigna@dia.uniroma3.it (Maurizio Patrignani) pizzonia@dia.uniroma3.it (Maurizio Pizzonia)

1 Introduction

In most networks it is possible to find semantic relationships among components that allow to group them into clusters. For example, in a social network representing employees of a company and their work relationships it could be desirable to group the people of each department together. As another example, the routers of the Internet are often grouped into areas, that in turn can be grouped into Autonomous Systems.

When representing a network containing clusters, it is quite common to draw the elements of each cluster inside the same region of the plane. Also, disjoint clusters typically lie into disjoint regions. The clustered planarity field studies the interplay between the classical planarity of graphs and the presence of clusters. For its practical interest and because of its theoretical appeal, clustered planarity is attracting increasing attention (see, e.g., [17, 10, 15, 16, 11, 3]).

More formally, a *clustered graph* $C(G, T)$ consists of a graph G and of a rooted tree T such that the leaves of T are the vertices of G (see Fig. 1.a and 1.b). Each internal node ν of T corresponds to the subset $V(\nu)$ of the vertices of G (called *cluster*) that are the leaves of the subtree rooted at ν. Each non-leaf node of T has at least two children. The subgraph of G induced by $V(\nu)$ is denoted as $G(\nu)$. An edge e between a vertex of $V(\nu)$ and a vertex of $V - V(\nu)$ is said to be *incident* on ν. Graph G and tree T are called *underlying graph* and *inclusion tree*, respectively. A clustered graph is *c-connected* if for each node ν of T we have that $G(\nu)$ is connected (e.g., the clustered graph in Fig. 1.a and 1.b is c-connected).

In a *drawing* of a clustered graph $C(G, T)$ vertices and edges of G are drawn as points and curves [12], and each node ν of T is a simple closed region $R(\nu)$ such that:

- $R(\nu)$ contains the drawing of $G(\nu)$;
- $R(\nu)$ contains a region $R(\mu)$ if and only if μ is a descendant of ν in T; and
- any two regions $R(\nu_1)$ and $R(\nu_2)$ do not intersect if ν_1 is not an ancestor or a descendant of ν_2.

See Fig. 1.a for an example of a drawing of a clustered graph. Consider an edge e and a node ν of T. If e crosses the boundary of $R(\nu)$ more than once, we say that edge e and region $R(\nu)$ have an *edge-region crossing*. A drawing of a clustered graph is *c-planar* if it does not have edge crossings or edge-region crossings. A clustered graph is *c-planar* if it admits a c-planar drawing (e.g., the clustered graph in Fig. 1.a and 1.b is c-planar).

The problem of characterizing the c-planar clustered graphs has been tackled from many points of view and from many authors. Namely, Feng, Cohen, and Eades [20] have shown that a clustered graph $C(G, T)$ is c-planar iff G has a planar embedding such that for each node ν of T all the vertices and edges of $G - G(\nu)$ are on the external face of the embedding of $G(\nu)$. However, such a characterization is more a characterization of the c-planarity of the embeddings rather than a structural characterization of the c-planarity of the graphs.

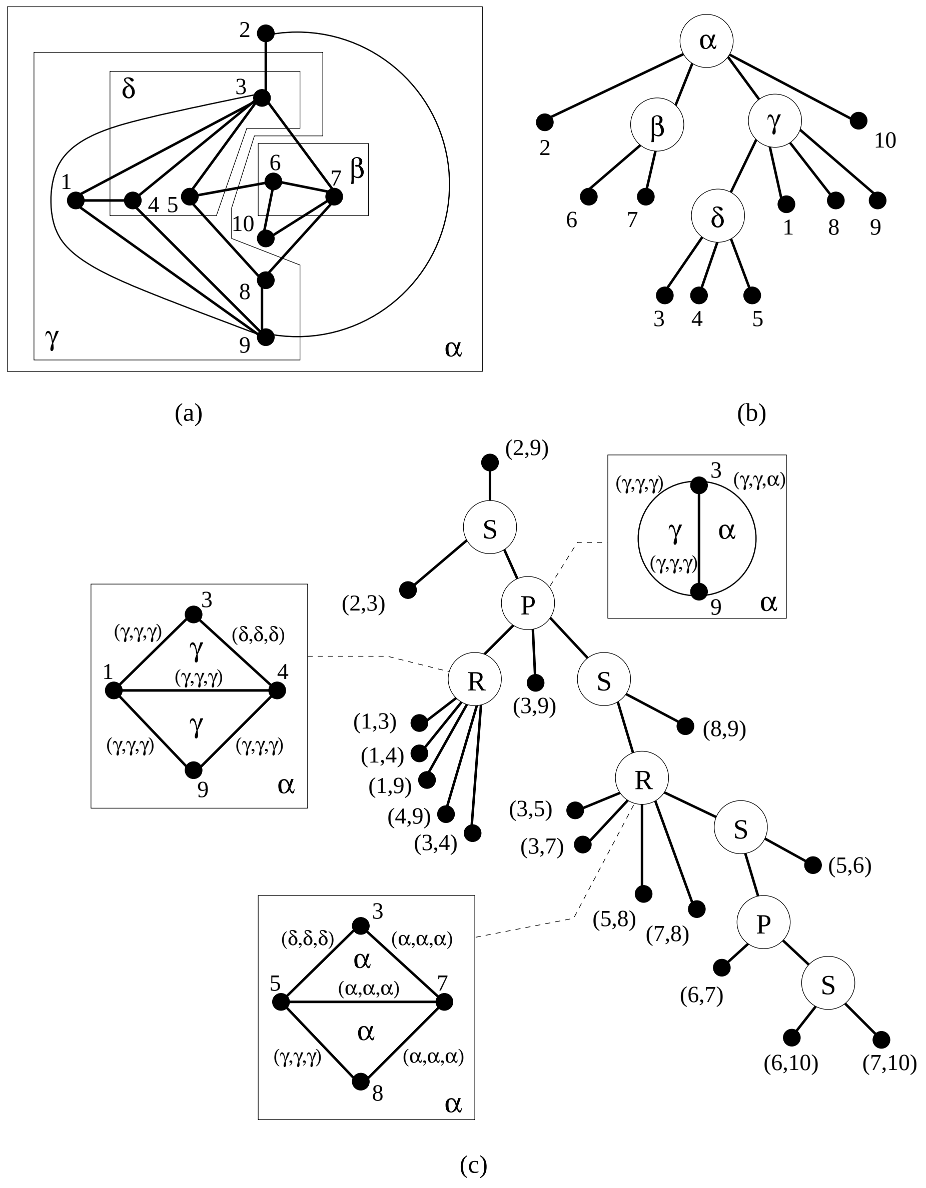

Figure 1: (a) A drawing of a clustered graph $C(G,T)$. The underlying graph G of C is drawn with thick lines. (b) The inclusion tree T of C. (c) The SPQR-tree of G. The boxes contain the skeletons of selected nodes. The triple on each virtual edge represents $lcc(e)$, $lsc(e)$, and $hsc(e)$, respectively. The faces of the skeletons are labelled with their lcc.

Another characterization of the c-planar embeddings, based on the connectivity properties of their dual graphs, is due to Dahlhaus [9]. A further elegant characterization has been given by Cornelsen and Wagner [5] for a subclass of clustered graphs. Namely, a completely connected clustered graph is c-planar iff its underlying graph is planar, where completely connected means that for each node ν of T, $G(\nu)$ and $G - G(\nu)$ are connected (e.g., the clustered graph in Fig. 1.a and 1.b is not completely connected).

Testing a clustered graph for c-planarity is a problem of unknown time complexity in the general case [6]. There are several families of non-c-connected clustered graphs for which the c-planarity testing problem has been shown to be polynomial-time solvable.

- The results [2, 1] by Biedl, Kaufmann, and Mutzel on "partitioned drawings" of graphs can be interpreted as linear-time c-planarity tests for non-connected "flat" clustered graphs with exactly two clusters. A clustered graph $C(G, T)$ is *flat* if all the leaves of T have distance two from the root.

- Gutwenger et al. presented a polynomial-time algorithm for c-planarity testing for *almost connected* clustered graphs [22], i.e., graphs for which all nodes corresponding to the non-connected clusters lie on the same path in T starting at the root of T, or graphs in which for each non-connected cluster its parent cluster and all its siblings in T are connected.

- Cortese et al. studied the class of non-connected clustered graphs such that the underlying graph is a cycle and the clusters at the same level of T also form a cycle, where two clusters are considered adjacent if they are incident to the same edge [7]. The c-planarity testing and embedding problem is linear for this class of graphs.

- Goodrich et al. introduced a polynomial-time algorithm for producing planar drawings of *extrovert* clustered graphs [21], i.e., graphs for which all clusters are connected or extrovert. A cluster μ with parent ν is extrovert if and only if ν is connected and each connected component of μ has a vertex with an edge that is incident to a cluster which is external to ν.

- Cortese et al. showed that the c-planarity testing and embedding problem is polynomial-time solvable for *rigid clustered cycles*, that is, for flat clustered graphs where the underlying graph is a cycle and the graph of the clusters' adjacencies is planar and has a fixed embedding [8].

- Jelínková et al. presented a polynomial-time algorithm for testing the c-planarity of "k-rib-Eulerian" graphs [24]. A graph is *k-rib-Eulerian* if it is Eulerian and it can be obtained from a 3-connected planar graph with k vertices, for some constant k, by replacing some edges with one or more paths in parallel.

- Di Battista and Frati showed how to test in linear time the c-planarity of embedded flat clustered graphs with at most five vertices per face [13].

Concerning c-connected clustered graphs, there exist three polynomial-time algorithms, discussed below, to test their c-planarity.

- Feng, Cohen, and Eades presented in [20, 19] a quadratic-time algorithm. Their algorithm visits the inclusion tree of the clusters bottom-up, starting from the leaves. The subgraph induced by each cluster is tested for planarity with the constraint that the vertices adjacent to other clusters must be incident to the external face. If the test is positive the cluster is replaced in its parent by a "gadget" representing all its possible embeddings. All such planarity tests are performed using PQ-trees, whose structure is similar to the one of the adopted gadgets.

- Lengauer [25] found a result analogous to the one in [20, 19], but in a different context. Namely, in that case the clustered graph is specified in terms of a set of graph patterns and in terms of their composition. The time complexity of the algorithm is linear in the size of the input. However, the input size of Lengauer's algorithm can be quadratic in the size of the represented clustered graph.

- Dahlhaus [9] proposed a linear-time algorithm based on the following main ingredients: a decomposition of G into its biconnected and triconnected components, a weight of each cluster proportional to its size, and the above mentioned characterization of c-planar embeddings. The testing algorithm is based on the incremental construction of a certain planar embedding and on a final test that checks whether this embedding is c-planar. The work in [9] contains many interesting ideas and profound intuitions. However, it has also some weak points: it is hard to find in the paper a complete algorithmic description, there is no complete proof of correctness, and it is not clear how to perform in linear time some of the algorithmic steps.

In this paper we present the first (as far as we know) structural characterization of c-planarity for c-connected clustered graphs whose underlying graph is biconnected. The characterization is based on the interplay between the hierarchy of the clusters and the hierarchy of the triconnected components of the underlying graph G. It is given in terms of properties of the skeletons of the nodes of the SPQR-tree of G. Notice that in at least two other papers [9, 25] the relationship between triconnectivity and c-planarity has already been studied. We also easily extend the characterization to general clustered graphs exploiting the decomposition of G into its biconnected components.

Further, based on our c-planarity characterization, we present a new linear-time c-planarity testing and embedding algorithm for c-connected clustered graphs. Such an algorithm is reasonably easy to implement, since it is based on simple algorithmic tools as the computation of lowest common ancestors, minimum and maximum spanning trees, and counting sorts. It also makes use of well-established data structures as SPQR-trees and BC-trees [14, 23] (both in their simple static version). If the test fails, our algorithm identifies a structural element responsible for the non-c-planarity of the input clustered graph.

The paper is organized as follows. In Section 2 we give basic definitions. In Section 3 we present the characterization for c-connected clustered graphs whose underlying graph is biconnected. Section 4 extends the characterization to c-connected clustered graphs whose underlying graph is simply connected. In Section 5 we describe a linear-time algorithm for testing the c-planarity of c-connected clustered graphs whose underlying graph is biconnected, and in Section 6 we extend such an algorithm to handle clustered graphs whose underlying graph is simply connected. Finally, in Section 7, we compare our results with previous ones. Section 8 contains our conclusions.

2 Background

We assume familiarity with planarity and connectivity of graphs [18]. We also assume some familiarity with graph drawing [12].

A *drawing* D of a graph $G(V, E)$ is a mapping of each vertex $v \in V$ to a distinct point $D(v)$ of the plane and of each edge $(u, v) \in E$ to a Jordan curve joining points $D(u)$ and $D(v)$. Given a drawing D of a graph G, two edges *intersect* if they share a point which is not a common end-vertex. A drawing with no intersection is *planar* and a graph is *planar* if it admits a planar drawing. A planar drawing partitions the plane into topologically connected regions called *faces*. The unbounded face is called the *external face*. Let G be a planar graph. An *embedded planar graph* G_Γ is an equivalence class of planar drawings of G with the same circular ordering for the adjacency lists of the vertices and with the same external face. Such a choice Γ of a circular ordering of the adjacency lists and of the external face is called *planar embedding* of G.

A graph $G(V, E)$ is *connected* if every pair of vertices of G is connected by a path. A *separating k-set* of a graph G is a set of k vertices whose removal increases the number of connected components of G. Separating 1-sets and 2-sets are called *cutvertices* and *separation pairs*, respectively. A connected graph is said to be *biconnected* if it has no cutvertex. The maximal biconnected subgraphs of a graph are its *blocks*. Observe that each edge of G falls into a single block of G, while cutvertices are shared by different blocks. The *block cutvertex tree*, or BC-tree, of a connected graph G has a B-node for each block of G and a C-node for each cutvertex of G. Edges in the BC-tree connect each B-node μ to the C-nodes associated with the cutvertices in the block of μ. The BC-tree of G may be thought as rooted at a specific block ν. Consider a cutvertex v of a BC-tree rooted at ν. The graph obtained by merging the blocks of which v is an ancestor is called the *pertinent graph* of v, and is denoted by pertinent(v).

A *split pair* $\{u, v\}$ of a graph G is either a separation pair or a pair of adjacent vertices. A *maximal split component* of G with respect to a split pair $\{u, v\}$ (or, simpler, a maximal split component of $\{u, v\}$) is either an edge (u, v) or a maximal subgraph G' of G such that G' contains u and v and $\{u, v\}$ is not a split pair of G'. A vertex w distinct from u and v belongs to exactly one maximal split component of $\{u, v\}$. We call the *split component* of $\{u, v\}$ a subgraph of G that is the union of any number of maximal split components of $\{u, v\}$.

In the following, we summarize SPQR-trees. For more details, see [14]. SPQR-trees are closely related to the classical decomposition of biconnected graphs into triconnected components. Let $\{s, t\}$ be a split pair of G. A *maximal split pair* $\{u, v\}$ of G with respect to $\{s, t\}$ is a split pair of G distinct from $\{s, t\}$ such that, for any other split pair $\{u', v'\}$ of G, there exists a split component of $\{u', v'\}$ containing vertices u, v, s, and t. Let $e = (s, t)$ be an edge of G, called *reference edge*. The SPQR-tree $\mathcal{T}$ of G with respect to e describes a recursive decomposition of G induced by its split pairs. Tree $\mathcal{T}$ is a rooted ordered tree whose nodes are of four types: S, P, Q, and R. Denote by G' the st-biconnectible graph obtained from G by removing e. Each node μ of $\mathcal{T}$ has an associated st-biconnectible multigraph, called the *skeleton* of μ and denoted by skeleton(μ). Also, it is associated with an edge of the skeleton of the parent ν of μ, called the *virtual edge* of μ in *skeleton*(ν). Tree $\mathcal{T}$ is recursively defined as follows.

Trivial Case: If G consists of exactly one edge between s and t, then $\mathcal{T}$ consists of a single Q-node whose skeleton is G itself.

Parallel Case: If the split pair $\{s, t\}$ has at least two maximal split components $G_1, \dots, G_k$ ($k \geq 2$), the root of $\mathcal{T}$ is a P-node μ. Graph *skeleton*(μ) consists of k parallel edges between s and t, denoted $e_1, \dots, e_k$.

Series Case: If the split pair $\{s, t\}$ has exactly one maximal split component G' which is not a single edge and if G' has cutvertices $c_1, \dots, c_{k-1}$ ($k \geq 2$) in this order on a path from s to t, the root of $\mathcal{T}$ is an S-node μ. Graph *skeleton*(μ) is the path $e_1, \dots, e_k$, where e_i connects c_{i-1} with c_i ($i = 2 \dots k-1$), e_1 connects s with c_1, and e_k connects c_{k-1} with t.

Rigid Case: If none of the above cases applies, let $\{s_1, t_1\}, \dots, \{s_k, t_k\}$ be the maximal split pairs of G with respect to $\{s, t\}$ ($k \geq 1$) and, for $i = 1, \dots, k$, let G_i be the union of all the maximal split components of $\{s_i, t_i\}$. The root of $\mathcal{T}$ is an R-node μ. Graph *skeleton*(μ) is the triconnected graph obtained from G by replacing each subgraph G_i with the edge e_i between s_i and t_i.

Except for the trivial case, μ has children $\mu_1, \dots, \mu_k$, in this order, such that μ_i is the root of the SPQR-tree of graph $G_i \cup (u_i, v_i)$ with respect to reference edge (u_i, v_i) ($i = 1, \dots, k$). Edge (u_i, v_i) is said to be the *virtual edge* of node μ_i in *skeleton*(μ) and of node μ in *skeleton*(μ_i). Graph G_i is called the *pertinent graph* of node μ_i, and of edge (u_i, v_i) and it is denoted by pertinent(u_i, v_i). Vertices u and v are the *poles* of G_i.

The tree $\mathcal{T}$ so obtained has a Q-node associated with each edge of G, except the reference edge e. We complete the SPQR-tree $\mathcal{T}$ by adding another Q-node, representing the reference edge e, and making it the parent of μ so that it becomes the root of $\mathcal{T}$ (see Fig. 1.c for an example).

The SPQR-tree $\mathcal{T}$ of a graph G with n vertices and m edges has m Q-nodes and $O(n)$ S-, P-, and R-nodes. Also, the total number of vertices of the skeletons stored at the nodes of $\mathcal{T}$ is $O(n)$.

A biconnected graph G is planar if and only if the skeletons of all the nodes of the SPQR-tree $\mathcal{T}$ of G are planar. An SPQR-tree $\mathcal{T}$ rooted at a given Q-node can be used to represent all the planar embeddings of G having the reference edge (associated with the Q-node at the root) on the external face. Namely, any embedding can be obtained by selecting one of the two possible flips of each skeleton around its poles and selecting a permutation of the children of each P-node with respect to their common poles.

3 A Characterization of the C-Connected C-Planar Clustered Graphs whose Underlying Graph is Biconnected

In this section we characterize the c-planarity of a c-connected clustered graph $C(G,T)$ when G is planar and biconnected. First, we introduce some definitions on the cluster hierarchy. Given a connected non-empty subgraph G' of G, the *allocation cluster* of G', denoted by $ac(G')$ is the lowest common ancestor in T of the vertices of G'. The allocation cluster represents the lowest cluster containing all the vertices of G'. In the following we refer to the allocation clusters of many special subgraphs of G like edges, paths, and cycles. For example in Fig. 1 the allocation cluster of path $3,5,8$ is γ. Two clusters α and β of T are *comparable* when they are on the same path from a leaf to the root of T. If α and β are comparable, the operators $\prec$, $\preceq$, and max are defined, where $\alpha \preceq \beta$ ($\alpha \prec \beta$) means that α is an ancestor (proper ancestor) of β and $\max(\alpha,\beta)$ is the farthest cluster from the root between α and β. The following properties are easy to prove.

Property 1 *Given two connected subgraphs G' and G'' of G sharing a vertex, $ac(G')$ and $ac(G'')$ are comparable. Also, if $G'' \subseteq G'$ then $ac(G') \preceq ac(G'')$.*

Property 2 *Let G' be a connected subgraph of G. There is at least one edge $e \in G'$ such that $ac(e) = ac(G')$.*

For example, in Fig. 1 the subgraph induced by vertices $3,5,8,7$ has allocation cluster equal to α and $ac((3,7)) = \alpha$.

Property 3 *There is at least one edge $e \in G$ such that $ac(e)$ is the root of T.*

Now we relate the concept of allocation cluster, typical of the clusters hierarchy, to the hierarchy of the triconnected components of G, represented by the SPQR-tree $\mathcal{T}$. A *lowest connecting path* of a virtual edge $e = (u,v)$ of the skeleton of a node of $\mathcal{T}$ is a path between u and v in *pertinent*(e) with maximum allocation cluster. Observe that by Property 1 all the paths connecting

u and v are pairwise comparable. The *lowest connecting cluster* of e, denoted by $lcc(e)$ is the allocation cluster of the lowest connecting path of e. Fig. 2 shows an example of a virtual edge e and its lowest connecting path. The allocation cluster of path $p_1 = (u, x, w, y, v)$ is β, while the allocation cluster of path $p_2 = (u, x, z, y, v)$ is α. Only p_1 is a lowest connecting path of e, and $lcc(e) = \beta$. Observe that if $e \in G$ the lowest connecting path of e is the edge itself.

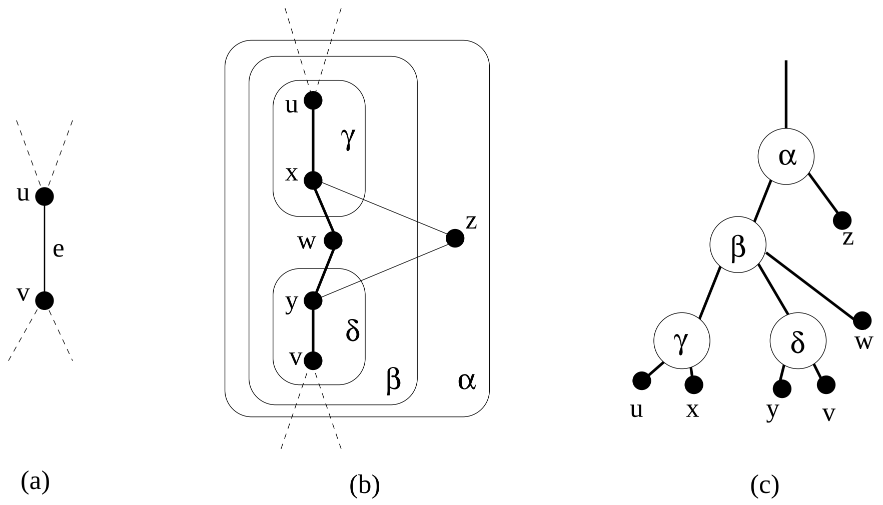

Figure 2: Lowest connecting paths and clusters. (a) A virtual edge $e = (u, v)$. (b) Graph *pertinent*(e); the thick lines show the lowest connecting path p_1. (c) Tree T restricted to the clusters in *pertinent*(e).

Consider a skeleton of a node μ of $\mathcal{T}$ and a path p composed by virtual edges of the skeleton. The *lowest connecting cluster* of p is the lowest common ancestor of the lowest connecting clusters of the edges of p. To have some intuition on this definition, consider that each edge of p corresponds to a pertinent graph that has a lowest connecting path. Hence, we can see p as "representative" of the concatenation of the lowest connecting paths traversing such pertinent graphs. Each of such paths has an allocation cluster; the lowest connecting cluster is the lowest common ancestor of such allocation clusters. We shall adopt the same definition of lowest connecting cluster also for cycles and faces of *skeleton*(μ). Also, for technical reasons we define the lowest connecting cluster of an external face as the root of the inclusion tree T. In Fig. 1.c the faces of the skeletons are labelled with the corresponding lowest connecting clusters.

Now we relate the above definitions to the c-planar embeddings. In the following when we refer to a planar embedding of a pertinent graph we always suppose that it has its poles on the external face.

Theorem 1 *A planar embedding of a c-connected clustered graph is c-planar iff it does not exist a cycle c that encloses an edge e such that $ac(e) \prec ac(c)$.*

Proof: The idea of the proof of necessity is that $ac(e) \prec ac(c)$ implies that the region of the drawing representing $ac(c)$ would enclose the region representing a proper ancestor of $ac(c)$. The sufficiency is proved by observing that if c and e do not exist, then there exists a drawing such that the enclosure relationships of the regions representing the clusters respect the inclusion tree.

Let $C(G,T)$ be a c-connected clustered graph. First, we prove that if a planar embedding Γ of G is c-planar then every cycle c of Γ does not enclose any edge e such that $ac(e) \prec ac(c)$. Suppose that there exist in Γ a cycle c and an edge e such that e is enclosed in c and that $ac(e) \prec ac(c)$. It follows that the region $R(\alpha)$ representing the allocation cluster α of c encloses an edge whose allocation cluster β is a proper ancestor of α. Since $G(\alpha)$ is connected and since Γ is planar, then $R(\beta)$ is enclosed in $R(\alpha)$ and so Γ is not c-planar.

Now, in the hypothesis that Γ is a planar embedding of G such that every cycle c of Γ does not enclose any edge e with $ac(e) \prec ac(c)$, we prove that Γ is c-planar. Construct any planar drawing of G with embedding Γ and draw any region $R(\gamma)$ representing a cluster γ of T so that $R(\gamma)$ "surrounds" $G(\gamma)$, i.e., $R(\gamma)$ contains any vertex and edge of $G(\gamma)$ and does not contain a vertex or an edge of $G - G(\gamma)$. The obtained drawing D of C is a c-planar drawing. Namely, since D is a planar drawing then it has no edge crossings; further, the construction of the regions and the c-connectivity of C implies that D has no edge-region crossings. It remains to show that D does not contain two regions $R(\alpha)$ and $R(\beta)$, respectively associated to clusters α and β of T, such that: (i) $R(\alpha)$ encloses $R(\beta)$ and (ii) α is not an ancestor of β. This is done by showing that if there exist in D regions $R(\alpha)$ and $R(\beta)$ that verify conditions (i) and (ii), then there exists a cycle c of Γ that encloses an edge e such that $ac(e) \prec ac(c)$.

By the construction of the regions, there exists in D a cycle c belonging to $G(\alpha)$ that encloses the subgraph $G^* = G(\beta)$. If β is a proper ancestor of α then trivially $ac(e) \prec ac(c)$, for some edge $e \in G^*$. Otherwise, consider the subgraph G' of G that is inside c in D, where $c \in G'$. By the c-connectivity of C, G' is connected. Since α is not an ancestor of β and since β is not a proper ancestor of α then α and β are not comparable. This implies that the allocation cluster of G' is a proper ancestor of both c and G^*, that is $ac(G') \prec ac(c)$ and $ac(G') \prec ac(G^*)$. By Property 2 there exists at least one edge $e \in G'$ such that $ac(e) = ac(G')$. Edge e can not be part of c and so it is enclosed in c. Hence, we obtain $ac(e) = ac(G') \prec ac(c)$, so we derived a contradiction and this concludes the proof. $\square$

Given a node μ of the SPQR-tree $\mathcal{T}$ of the underlying graph G of the clustered graph $C(G,T)$, an embedding of *skeleton*(μ) is *c-planar* if every cycle c of *skeleton*(μ) does not enclose an edge e of *skeleton*(μ) with $lcc(e) \prec lcc(c)$. Intuitively, the c-planarity of a skeleton is the one of the embedded graph obtained by substituting each edge of the skeleton with its lowest connecting path.

Lemma 1 *If an embedding Γ of skeleton(μ) is c-planar then for any cycle c in Γ and for any face f inside c we have that $lcc(c) \preceq lcc(f)$.*

The proof of Lemma 1 is analogous to the one of Theorem 1.

Given a virtual edge $e = (u, v)$ and a c-planar embedding Γ of $pertinent(e)$, a lowest connecting path s of e separates $pertinent(e)$ into two embedded subgraphs each containing s. We call *highest side* $hs(\Gamma, s)$ and *lowest side* $ls(\Gamma, s)$ such subgraphs, where $ac(hs(\Gamma, s)) \preceq ac(ls(\Gamma, s))$. By Properties 1 and 2 we have:

Property 4 *$ac(hs(\Gamma, s)) = ac(pertinent(e))$.*

Hence, the value of $ac(hs(\Gamma, s))$ does not depend on the choice of the c-planar embedding Γ and of s and we can define the *highest side cluster* of e, $hsc(e) = ac(pertinent(e))$.

Lemma 2 *The value of $ac(ls(\Gamma, s))$ does not depend on the choice of s.*

Proof: Suppose, by contradiction, that there exist two different lowest connecting paths s_1 and s_2, with $lcc(s_1) = lcc(s_2) = \gamma$, such that $\alpha = ac(ls(\Gamma, s_1))$, $\beta = ac(ls(\Gamma, s_2))$, and $\alpha \neq \beta$. By Property 1, we have that α and β are comparable. Assume, w.l.o.g, that $\alpha \prec \beta \preceq \gamma$. Then, there exists an edge e_1 such that $ac(e_1) = \alpha$. Edge e_1 belongs to $ls(\Gamma, s_1)$ and does not belong to $ls(\Gamma, s_2)$, then e_1 is necessarily enclosed in a simple cycle c composed by subpath $\overline{s}_1$ of s_1 and subpath $\overline{s}_2$ of s_2 (see Fig. 3). By Property 1, $lcc(\overline{s}_1)$ and $lcc(\overline{s}_2)$ are comparable. By construction, $lcc(c) = \min\{lcc(\overline{s}_1), lcc(\overline{s}_2)\} \succeq \gamma \succ \alpha$. Hence, by Theorem 1, Γ would be a non-c-planar embedding, contradicting the hypothesis. □

Due to Lemma 2 we can define the *lowest side cluster* of Γ $lsc(\Gamma) = ac(ls(\Gamma, s))$ and the *lowest side cluster* of e, $lsc(e) = \max_{\Gamma}\{lsc(\Gamma)\}$. Observe that the definitions of $hsc(e)$ and of $lsc(e)$ hold only if $pertinent(e)$ is c-planar. For technical reasons if $pertinent(e)$ is not c-planar we define $hsc(e) = lsc(e) = \perp$, where $\perp$ is by convention a proper ancestor of any cluster. See Fig. 4 for an example. As another example, Fig. 1 contains, for each virtual edge e of the represented skeletons, a triple describing $lcc(e)$, $lsc(e)$, and $hsc(e)$, respectively.

Property 5 *For each edge e of the skeleton of a node of $\mathcal{T}$, $hsc(e) \preceq lsc(e) \preceq lcc(e)$.*

Property 6 *Let c be a cycle of virtual edges in a skeleton of a node of $\mathcal{T}$ and let e be an edge of c. We have that $lcc(c)$ is comparable with $lcc(e)$, with $lsc(e)$, and with $hsc(e)$.*

Property 7 *Let $e = (u, v)$ be a virtual edge. Suppose that $pertinent(e)$ is c-planar. Then in any c-planar embedding Γ of $pertinent(e)$ and for any lowest connecting path s of e there exist two edges $e_1 \in hs(\Gamma, s)$ and $e_2 \in ls(\Gamma, s)$ such that $ac(e_1) = hsc(e)$ and $ac(e_2) \preceq lsc(e)$. Also, if $hsc(e) \prec lcc(e)$ then there exists an edge $e_1 \in hs(\Gamma, s)$ such that $ac(e_1) = hsc(e)$ and $e_1 \notin s$. Similarly, if $lsc(e) \prec lcc(e)$ then there exists $e_2 \in ls(\Gamma, s)$ such that $ac(e_2) \preceq lsc(e)$ and $e_2 \notin s$.*

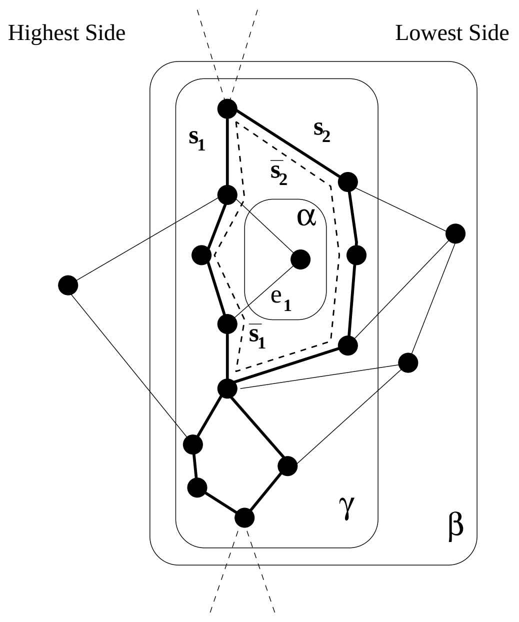

Figure 3: Illustration for the proof of Lemma 2. The paths s_1 and s_2 are drawn in thick lines, while the subpaths $\overline{s}_1$ and $\overline{s}_2$ are drawn in dashed lines.

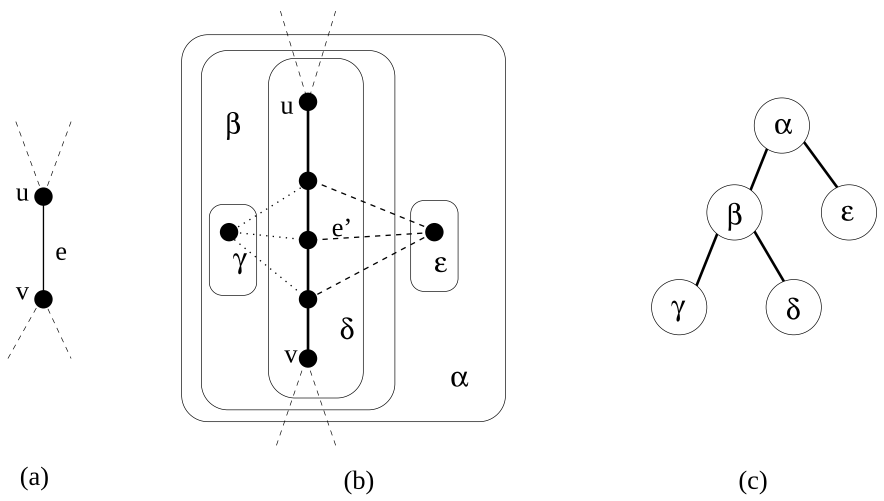

Figure 4: (a) A virtual edge $e = (u, v)$. (b) Graph $pertinent(e)$. (c) Tree T restricted to the clusters in $pertinent(e)$. We have that $lcc(e) = \delta$, $lsc(e) = \beta$, and $hsc(e) = \alpha$. If the edge e' is removed from $pertinent(e)$ then $lsc(e)$ becomes δ.

Two comparable virtual edges e_1 and e_2 of a skeleton of a node of $\mathcal{T}$ are *incompatible* when, assuming w.l.o.g. $lcc(e_1) \preceq lcc(e_2)$, one of the following conditions hold: (i) $lcc(e_1) \prec lcc(e_2)$ and $hsc(e_2) \prec lcc(e_1)$; (ii) $lcc(e_1) = lcc(e_2)$, $hsc(e_1) \prec lcc(e_1)$, and $hsc(e_2) \prec lcc(e_2)$.

For example, the skeleton of the P-node shown in Fig. 1.c has three virtual edges that are pairwise compatible. Now we can formulate the characterization.

Theorem 2 *Let $C(G,T)$ be a c-connected clustered graph where G is planar and biconnected, and let $\mathcal{T}$ be the SPQR tree of G rooted at an edge whose allocation cluster is the root of T. C is c-planar if and only if for each node μ of $\mathcal{T}$ the following conditions are true:*

1. *If μ is an R node then the embedding of skeleton(μ) is c-planar and each edge e of skeleton(μ) is incident to two faces f_1 and f_2 such that the lowest connecting cluster of f_1 is an ancestor of the highest side cluster of e and the lowest connecting cluster of f_2 is an ancestor of the lowest side cluster of e.*

2. *If μ is a P node then*

   (a) it does not exist a set of three edges of skeleton(μ) that are pairwise incompatible and

   (b) there exists at most one edge e^* of skeleton(μ) such that the lowest side cluster of e^* is a proper ancestor of the lowest connecting cluster of e^* and if there exists such e^* then for each edge $e \neq e^*$ of skeleton(μ) the lowest connecting cluster of e is an ancestor of the lowest side cluster of e^*.

3.1 Proof of the Sufficiency

The sufficiency of the conditions of Theorem 2 can be proved by structural induction starting from the leaves of $\mathcal{T}$. For each leaf μ (Q node) of $\mathcal{T}$, *pertinent*(μ) is trivially c-planar. We prove the inductive step by considering any non-leaf node μ with children $\mu_1, \mu_2, \ldots, \mu_k$. Namely, we suppose that their pertinent graphs are c-planar and argument that, if the conditions of the theorem are satisfied, then also *pertinent*(μ) is c-planar. This is done by means of two lemmas, one for the R nodes and the other for the P nodes. The lemma for the R nodes is valid for all nodes whose skeleton has a fixed embedding and so it is formulated in the most general setting. The case of the S nodes is trivial. A special lemma describes the situation of the root of $\mathcal{T}$ and completes the proof of sufficiency.

Lemma 3 *Let μ be a node of $\mathcal{T}$ and let Γ_μ be a c-planar embedding of skeleton(μ). If (i) the pertinent graphs of the children of μ are c-planar, and (ii) each edge e of skeleton(μ) is incident to two faces f_1 and f_2 of Γ_μ such that $lcc(f_1) \preceq hsc(e)$ and $lcc(f_2) \preceq lsc(e)$, then pertinent(μ) is c-planar.*

Proof: Let $e_1, e_2, \ldots, e_k$ be the edges of $skeleton(\mu)$ and consider for each e_i, $i = 1, \ldots, k$, a c-planar embedding Γ_i of $pertinent(e_i)$ such that $lsc(\Gamma_i) = lsc(e_i)$.

Denoting by f_{i1} and f_{i2} the faces of Γ_μ incident to e_i, construct an embedding Γ of $pertinent(\mu)$ by turning each Γ_i, $i = 1, \ldots, k$, so that $hs(\Gamma_i, s_i)$ is toward f_{i1} and $ls(\Gamma_i, s_i)$ is toward f_{i2} with $lcc(f_{i1}) \preceq hsc(e_i)$ and $lcc(f_{i2}) \preceq lsc(e)$. The second condition of the lemma makes this possible.

By supposing that Γ_μ is c-planar, we show that Γ is also c-planar, that is for any cycle c of Γ and any edge e that is enclosed in c we have $ac(c) \preceq ac(e)$ (see Theorem 1).

We denote by $e^* = (v_1, v_2)$ the edge of $skeleton(\mu)$ containing e in $pertinent(e^*)$ and we denote by f_1^* and f_2^* the faces incident to e^* in Γ_μ for which $lcc(f_1^*) \preceq hsc(e^*)$ and $lcc(f_2^*) \preceq lsc(e^*)$. Let Γ^* be the embedding of $pertinent(e^*)$ in Γ. We also denote by c^* the cycle in Γ_μ whose edges e_i contain in $\bigcup(pertinent(e_i))$ all the edges of c.

Edge e^* may be part of c^* or can be enclosed in c^*. In both cases at least one of f_1^* and f_2^* is enclosed by c^*. If c^* encloses f_1^*, we have $hsc(e^*) \preceq ac(e)$ by definition of $hsc(e^*)$, $lcc(f_1^*) \preceq hsc(e^*)$ by hypothesis and by the construction of Γ, $lcc(c^*) \preceq lcc(f_1^*)$ by Lemma 1, and $ac(c) \preceq lcc(c^*)$ by definition of lowest connecting cluster. Hence, $ac(c) \preceq lcc(c^*) \preceq lcc(f_1^*) \preceq hsc(e^*) \preceq ac(e)$.

If c^* encloses f_2^* (and does not enclose f_1^*), consider a lowest connecting path s of e^* and the path $p \subset c$ between v_1 and v_2 in $pertinent(e^*)$. If $e \in ls(\Gamma^*, s)$ then $lsc(e^*) \preceq ac(e)$ by definition of $lsc(e^*)$, $lcc(f_2^*) \preceq lsc(e^*)$ by hypothesis and by the construction of Γ, $lcc(c^*) \preceq lcc(f_2^*)$ by Lemma 1, and $ac(c) \preceq lcc(c^*)$ by definition of lowest connecting cluster, so $ac(c) \preceq lcc(c^*) \preceq lcc(f_2^*) \preceq lsc(e^*) \preceq ac(e)$. If $e \in hs(\Gamma^*, s)$ then there exists a simple cycle $\overline{c}$ formed by p and s, or by a part of them, such that e is enclosed by or is part of $\overline{c}$. In both cases the c-planarity of Γ^* implies that $ac(\overline{c}) \preceq ac(e)$. Also, $ac(p) \preceq ac(\overline{c})$ ($ac(p)$ is an ancestor of $ac(s)$ and of $ac(\overline{c})$) and since p is part of c $ac(c) \preceq ac(p)$. So $ac(c) \preceq ac(p) \preceq ac(\overline{c}) \preceq ac(e)$. □

Lemma 4 *Let μ be a P node such that the pertinent graphs of its children are c-planar. Suppose that: (i) it does not exist a set of three edges of $skeleton(\mu)$ that are pairwise incompatible, (ii) there exists at most one edge e^* of $skeleton(\mu)$ such that $lsc(e^*) \prec lcc(e^*)$, and (iii) if there exists e^* then each edge $e \neq e^*$ of $skeleton(\mu)$ is such that $lcc(e) \preceq lsc(e^*)$. Then $pertinent(\mu)$ is c-planar*

Proof: First, observe that all the edges of $skeleton(\mu)$ are pairwise comparable since they share the poles. Subdivide the edges of $skeleton(\mu)$ into two ordered sets $I_L = \{l_1, l_2, \ldots, l_p\}$ and $I_R = \{r_1, r_2, \ldots, r_q\}$ such that all the edges in I_L (in I_R) are pairwise compatible. This partition exists since the nature of the incompatibility relationship guarantees that if $skeleton(\mu)$ does not contain three pairwise incompatible edges then the incompatibility graph is bipartite. Also the edges in I_L (in I_R) are ordered so that $lcc(l_p) \preceq lcc(l_{p-1}) \preceq \ldots \preceq lcc(l_1)$ ($lcc(r_q) \preceq lcc(r_{q-1}) \preceq \ldots \preceq lcc(r_1)$). If two or more edges $l_i, l_{i+1}, \ldots, l_m$ ($r_j, r_{j+1}, \ldots, r_n$) in I_L (in I_R) have the same lowest connecting cluster, they are ordered so that $hsc(l_m) \preceq hsc(l_{m-1}) \preceq \ldots \preceq hsc(l_i)$ ($hsc(r_n) \preceq hsc(r_{n-1}) \preceq \ldots \preceq hsc(r_j)$). If two or more edges l_i (r_j) in I_L (in I_R) have the same lowest connecting cluster and the same highest side cluster, they are ordered in any way.

Consider the embedding Γ_μ of *skeleton*(μ) obtained by placing its edges in the order $I = \{l_p, \ldots, l_1, r_1, \ldots, r_q\}$. We show that *skeleton*(μ) is c-planar. This is done by considering three edges e_1, e_2, e_3 that appear in this order in Γ_μ and by proving that the lowest connecting cluster of the cycle $c = \{e_1, e_3\}$ is an ancestor of the lowest connecting cluster of e_2. If e_1 and e_3 are both in I_L (in I_R), then $lcc(c) = lcc(e_1) \preceq lcc(e_2)$ (resp. $lcc(c) = lcc(e_3) \preceq lcc(e_2)$). Otherwise $e_1 \in I_L$ and $e_3 \in I_R$. Suppose $e_2 \in I_L$ ($e_2 \in I_R$), $lcc(c) \preceq lcc(e_1) \preceq lcc(e_2)$ (resp. $lcc(c) \preceq lcc(e_3) \preceq lcc(e_2)$).

We now show that Γ_μ is such that each edge e of *skeleton*(μ) is incident to two faces f_1 and f_2 such that $lcc(f_1) \preceq hsc(e)$ and $lcc(f_2) \preceq lsc(e)$. The internal faces of Γ_μ consist of two edges that are consecutive in I. We denote by $\{e_1, e_2\}$ an internal face of Γ_μ between edges e_1 and e_2. We have that $lcc(\{l_i, l_{i+1}\}) = lcc(l_{i+1})$, for $1 \leq i < p$, and that $lcc(\{r_j, r_{j+1}\}) = lcc(r_{j+1})$, for $1 \leq j < q$. Since l_i and l_{i+1}, for $1 \leq i < p$ (r_j and r_{j+1}, for $1 \leq j < q$), are compatible, then $lcc(\{l_i, l_{i+1}\}) \preceq hsc(l_i)$ ($lcc(\{r_j, r_{j+1}\}) \preceq hsc(r_j)$). Also denoting by f_e the external face of Γ_μ we clearly have $lcc(f_e) \preceq hsc(l_p)$ and $lcc(f_e) \preceq hsc(r_q)$. By the second condition on the P nodes there exists at most one edge e^* such that $lsc(e^*) \prec lcc(e^*)$, and so $lcc(f_2) \preceq lsc(e)$ can be violated only for $e = e^*$. By the ordering of the edges in $\Gamma(\mu)$, we have that either $e^* = r_1$ or $e^* = l_1$. W.l.o.g. suppose that $e^* = r_1$. By the third condition on the P nodes $lcc(\{l_1, r_1\}) \preceq lcc(l_1) \preceq lsc(r_1)$. By Lemma 3 we can conclude that *pertinent*(μ) is c-planar. □

Lemma 5 *Let $C(G,T)$ be a c-connected clustered graph and let (u,v) be an edge of G such that $ac(u,v)$ is the root of T. If $C(G-(u,v),T)$ admits a c-planar embedding with u and v on the external face, then C is c-planar.*

Proof: Since $ac((u,v))$ is the root of T, it is easy to see that (u,v) cannot create cycles enclosing an edge whose allocation cluster is a proper ancestor of the allocation cluster of the cycle. Hence, by Theorem 1, C admits a c-planar embedding obtained by adding (u,v) to the one of $C(G-(u,v),T)$. □

The above lemma completes the proof of sufficiency.

3.2 Proof of the Necessity

The proof of the necessity of Theorem 2 is split into five lemmas, that, considered altogether, constitute a complete proof. Before giving such lemmas, we discuss the following issue: how limiting is the choice of rooting $\mathcal{T}$ to a specific edge whose allocation cluster is the root of T? The answer is in the following theorem.

Theorem 3 *Let $C(G,T)$ be a c-connected c-planar clustered graph and let Γ be any c-planar embedding of C. Let e be an edge whose allocation cluster is the* *root of T. Change the external face of Γ choosing as external any face containing e. The resulting planar embedding is still c-planar.*

Proof: Let f be the external face of Γ and let Γ' be the embedding of C derived from Γ with external face g containing e. By contradiction, suppose that Γ' is not c-planar. By Theorem 1, there exists in Γ' a cycle c that encloses an edge $\overline{e}$ such that $ac(\overline{e}) \prec ac(c)$. Note that, $ac(c)$ cannot be the root of T, hence we have also $ac(e) \prec ac(c)$. Consider c and f in Γ', either (i) c encloses f and separates f from g or (ii) c does not encloses f and hence f and g are not separated by c. In the first case, Γ cannot be c-planar since in this embedding c encloses e. In the second case, Γ cannot be c-planar since in this embedding c encloses $\overline{e}$. In both cases we have a contradiction. □

Now we start the proof of the necessity. The first two lemmas are aimed to prove the necessity of the conditions on the skeletons of the R nodes.

Lemma 6 *Let μ be an R node of $\mathcal{T}$ such that the conditions of Theorem 2 are satisfied for all nodes of the subtree rooted at μ but for μ itself. Namely, suppose that the embedding of skeleton(μ) with its poles on the external face is not c-planar. Then C is not c-planar.*

Proof: Because of the non-c-planarity, the unique embedding of *skeleton*(μ) with its poles u and v on the external face contains a cycle c^* that encloses an edge e^* such that $lcc(e^*) \prec lcc(c^*)$. It is possible to find in each embedding of *pertinent*(μ) a cycle c corresponding to c^* by substituting each virtual edge e^* of c^* with a lowest connecting path of e^*. By the definition of lowest connecting path, $ac(c) = lcc(c^*)$. Also, by the definition of $lcc(e^*)$ we have that *pertinent*(e^*) contains an edge e with $ac(e) = lcc(e^*)$. Since μ is an R node, in any embedding of *pertinent*(μ) e lies inside c. So, because of Theorem 1, *pertinent*(μ) is not c-planar with (u, v) on the external face and C is not c-planar with the root of $\mathcal{T}$ on the external face. By Theorem 3 it follows that C is not c-planar. □

Lemma 7 *Let μ be an R node of $\mathcal{T}$ such that the conditions of Theorem 2 are satisfied for all nodes of the subtree rooted at μ but for μ itself. Namely, suppose that skeleton(μ) contains an edge e incident to two faces f_1 and f_2 ($lcc(f_1) \preceq lcc(f_2)$) with $hsc(e) \prec lcc(f_1)$ or $lsc(e) \prec lcc(f_2)$, then C is not c-planar.*

Proof: Concatenating the lowest connecting paths of the edges composing f_1 (f_2), we can find in *pertinent*(μ) a cycle c_1 (c_2) corresponding to f_1 (f_2) of *skeleton*(μ) with $ac(c_1) = lcc(f_1)$ ($ac(c_2) = lcc(f_2)$). Due to Property 7 there exist two edges e_1 and e_2 of *pertinent*(e) such that $ac(e_1) = hsc(e)$, $ac(e_2) \preceq lsc(e)$ and that either e_1 lies inside c_1 and e_2 lies inside c_2 or e_1 lies inside c_2 and e_2 lies inside c_1. In both cases we have a cycle c that contains an edge e such that $ac(e) \prec ac(c)$, so *pertinent*(μ) is not c-planar with (u, v) on the external face and C is not c-planar with the root of $\mathcal{T}$ on the external face. By Theorem 3 it follows that C is not c-planar. □

The next three lemmas are to prove the necessity of the conditions of Theorem 2 on the skeletons of the P nodes.

Lemma 8 *Let μ be a P node of $\mathcal{T}$ such that the conditions of Theorem 2 are satisfied for all nodes of the subtree rooted at μ but for μ itself. Namely, suppose that $skeleton(\mu)$ contains three edges e_1, e_2, e_3 that are pairwise incompatible, then C is not c-planar.*

Proof: Consider any embedding of $skeleton(\mu)$ and suppose, w.l.o.g., that e_1, e_2, and e_3 are embedded in this order around the poles. Consider a cycle c of $pertinent(\mu)$ composed by a lowest connecting path of e_1 and by a lowest connecting path of e_3. We have that $lcc(c)$ is the lowest common ancestor of $lcc(e_1)$ and $lcc(e_3)$. By applying Property 7 it is possible to find an edge $e \in pertinent(e_2)$ such that $ac(e) = hsc(e_2)$. Since e_2 is incompatible with both e_1 and e_3 we have $ac(e) = hsc(e_2) \prec lcc(c)$. Hence, $pertinent(\mu)$ is not c-planar with (u, v) on the external face and C is not c-planar with the root of $\mathcal{T}$ on the external face. By Theorem 3 it follows that C is not c-planar. □

Lemma 9 *Let μ be a P node of $\mathcal{T}$ such that the conditions of Theorem 2 are satisfied for all nodes of the subtree rooted at μ but for μ itself. Namely, suppose that $skeleton(\mu)$ contains two edges e_1^* and e_2^* with $lsc(e_1^*) \prec lcc(e_1^*)$ and $lsc(e_2^*) \prec lcc(e_2^*)$, then C is not c-planar.*

Proof: Consider a lowest connecting path p_1 of e_1^* and a lowest connecting path p_2 of e_2^*. Let c be the cycle obtained by concatenating p_1 and p_2; we have that $lcc(c)$ is the lowest common ancestor of $lcc(e_1^*)$ and $lcc(e_2^*)$. W.l.o.g. we assume that $lcc(c) = lcc(e_1^*)$. By Property 7 in any embedding of $pertinent(e_1^*)$ there exist edges e_1 and e_2 that are separated by p_1 and are such that $ac(e_1) = hsc(e_1^*)$ and $ac(e_2) \preceq lsc(e_1^*)$. One between e_1 and e_2 is enclosed by c. By the above inequalities $ac(e_1) \prec lcc(c)$, $ac(e_2) \prec lcc(c)$, and hence $pertinent(\mu)$ is not c-planar with (u, v) on the external face and C is not c-planar with the root of $\mathcal{T}$ on the external face. By Theorem 3 it follows that C is not c-planar. □

Lemma 10 *Let μ be a P node of $\mathcal{T}$ such that the conditions of Theorem 2 are satisfied for all nodes of the subtree rooted at μ but for μ itself. Namely, suppose that $skeleton(\mu)$ contains an edge e^* with $lsc(e^*) \prec lcc(e^*)$ and an edge $e \neq e^*$ such that $lsc(e^*) \prec lcc(e)$, then C is not c-planar.*

Proof: By Property 5 we have that $hsc(e^*) \prec lcc(e)$. Consider a lowest connecting path p_{e^*} of e^*, a lowest connecting path p_e of e, and the cycle $c = p_{e^*} \cup p_e$; $lcc(c)$ is the lowest common ancestor of $lcc(e^*)$ and $lcc(e)$. By Property 7 in any embedding of $pertinent(e^*)$ there exist edges e_1 and e_2 that are separated by p_{e^*} and are such that $ac(e_1) = hsc(e^*)$ and $ac(e_2) \preceq lsc(e^*)$. One between e_1 and e_2 is enclosed by c. By the above inequalities $ac(e_1) \prec lcc(c)$, $ac(e_2) \prec lcc(c)$, and hence with (u, v) on the external face and C is not c-planar with the root of $\mathcal{T}$ on the external face. By Theorem 3 it follows that C is not c-planar. □

The above lemma completes the proof of necessity of Theorem 2.

4 Characterization of the C-Planarity of General C-connected Clustered Graphs

In this section we extend the characterization given in Section 3 to general c-connected clustered graphs.

Theorem 4 *Let $C(G,T)$ be a c-connected clustered graph and let $\mathcal{B}$ be the BC-tree of G rooted at a block ν that contains an edge e whose allocation cluster is the root of T. C is c-planar if and only if each block μ of $\mathcal{B}$ admits a c-planar embedding Γ_μ such that the parent cut-vertex of μ (if any) is on the external face of Γ_μ and each child cut-vertex ρ_i of μ is incident to a face f_i whose lowest connecting cluster is an ancestor of the allocation cluster of $pertinent(\rho_i)$.*

Proof: First, we show the sufficiency of the conditions. Suppose each block μ admits an embedding Γ_μ respecting the above conditions. We show how to build a c-planar embedding Γ_G of G, by suitably merging the embeddings Γ_μ. We traverse top-down $\mathcal{B}$ starting at its root ν. Let μ be the current block, and let $\mu_1, \mu_2, \dots, \mu_k$ be the blocks whose parent cutvertices $\rho_1, \rho_2, \dots, \rho_k$ (not necessarily distinct) are children of μ. We select a face f_i of Γ_μ incident to ρ_i and whose lowest connecting cluster is an ancestor of the allocation cluster of pertinent(ρ_i). We embed Γ_{μ_i} into f_i identifying the two instances of ρ_i in μ and μ_i. Distinct children of the same cutvertex are embedded in such a way that one does not enclose the other. Now we show that the obtained embedding Γ_G is c-planar. Suppose, by contradiction, that Γ_G is not c-planar. Let p be a simple cycle enclosing an edge e with $ac(e) \prec ac(p)$. Observe that, since p is simple, all edges of p are contained into the same block μ^*. If e also belongs to μ^*, then Γ_{μ^*} is not c-planar, contradicting the hypothesis. Otherwise, suppose e belongs to pertinent(ρ_i), where ρ_i is a child cutvertex of μ^*. Hence, $ac(pertinent(\rho_i)) \preceq ac(e)$. By construction, $ac(f_i) \preceq ac(pertinent(\rho_i))$. Since each edge of f_i belongs or is internal to p, Theorem 1 ensures that $ac(p) \preceq ac(f_i)$. Therefore, we have $ac(p) \preceq ac(e)$ contradicting the hypothesis that $ac(e) \prec ac(p)$.

It is trivial that the c-planarity of the blocks of the BC-tree is a necessary condition for the c-planarity of C. Also, by Theorem 3, the choice of rooting $\mathcal{B}$ to a node ν containing an edge e whose allocation cluster is the root of T, and the choice of rooting $\mathcal{T}_\nu$ to e are not limiting, that is if a c-planar embedding Γ_G of G exists, then there exists also a c-planar embedding of G with e on the external face. This leads to assume that Γ_G has e on the external face. In order to show the necessity of the other two conditions, suppose that a block $\mu^* \neq \nu$ does not admit a c-planar embedding with its parent cutvertex on the external face. Then the blocks that are ancestors of μ^* have to be embedded inside μ^* and so e cannot be on the external face of Γ_G, contradicting the hypothesis. Now suppose that a child cutvertex ρ_i of μ is such that $ac(pertinent(\rho_i)) \prec ac(f_j)$ for each face f_j of Γ_μ incident to ρ_i. By Property 2 $pertinent(\rho_i)$ contains an edge e_i such that $ac(e_i) = ac(pertinent(\rho_i))$. So embedding $pertinent(\rho_i)$ inside μ creates a cycle f_j containing an edge e_i such that $ac(e_i) \prec ac(f_j)$, while embedding μ inside any embedding of $pertinent(\rho_i)$ implies that e cannot be on the external face of Γ_G, contradicting the hypothesis. Hence, C is not c-planar with e on the external face and by Theorem 3 it follows that C is not c-planar. □

5 Testing and Embedding Algorithm: Biconnected Case

In this section we describe a linear-time algorithm for testing the c-planarity and computing a c-planar embedding for c-connected clustered graphs whose underlying graph is biconnected. First, we show in Section 5.1 how the c-planarity characterizations given in Sections 3 and 4 can be modified in such a way to produce conditions that are easy to check in linear time. Then, we provide an overview of the algorithm in Section 5.2. Sections 5.3 and 5.4 contain the description of the two main phases of the algorithm.

5.1 Encoding the Cluster Hierarchy

The characterizations provided by Theorems 1, 2 and 4 only require to test if a cluster is an ancestor or proper ancestor of another cluster. In fact, we only need to perform comparisons between clusters that are comparable, i.e., that lie on the same path from the root to a leaf of T.

Let ψ be a function associating each node μ of T to a value $\psi(\mu)$ such that $\psi(\mu) > \psi(\nu)$, if ν is the parent of μ. We can recast the c-planarity conditions by replacing each condition on T with comparisons between suitable values of ψ. In the following we adopt as function $\psi(\mu)$ the *depth*, denoted $d(\mu)$ where the depth of the root of T is zero and $d(\mu) = d(\nu) + 1$ if ν is the parent of μ.

Observe that the use of the depth instead of the allocation cluster allows to replace several definitions given on the tree T with depth values. Namely, the lowest connecting cluster $lcc(e)$ of a virtual edge e can be replaced by its depth. We denote the value of $d(lcc(e))$ by $d(e)$. Analogously, the lowest connecting cluster $lcc(f)$ of a face f can be replaced by its depth $d(lcc(f))$, denoted $d(f)$. In a similar way we define the *highest (lowest) side depth* of a virtual edge e as $hsd(e) = d(hsc(e))$ ($lsd(e) = d(lsc(e))$.

According to the above definitions, both the incompatibility of two edges and the conditions of Theorems 2 and 4, can be restated by replacing each occurrence of $\prec$ and $\preceq$ with $<$ and $\leq$, respectively, and by replacing each occurrence of $ac(\cdot)$, $lcc(\cdot)$, $hsc(\cdot)$, and $lsc(\cdot)$ with $d(\cdot)$, $d(\cdot)$, $hsd(\cdot)$, and $lsd(\cdot)$, respectively.

5.2 Overview of the Algorithm

The input of the algorithm is a c-connected clustered graph $C(G, T)$ such that G is biconnected and planar. The output of the algorithm is a c-planar embedding of C or a node of the SPQR-tree $\mathcal{T}$ for which the c-planarity conditions are not verified. The algorithm consists of two phases that are sketched below and fully described in the following sections.

Preprocessing. This phase consists of three steps.

SPQR-tree Decomposition. First, we compute the depth of each edge e of G. Second, we compute an SPQR-tree $\mathcal{T}$ of G rooted at any edge e_r of depth zero.

Skeleton-Labelling. We label each non-virtual edge e of the skeletons of $\mathcal{T}$ with the three labels $d(e) = hsd(e) = lsd(e)$, which are equal to the depth of the corresponding edge of G. Each virtual edge e is labeled with $d(e)$ and $hsd(e)$ only, by performing a suitable bottom-up traversal of $\mathcal{T}$.

Edges-Sorting. We sort the edges of each P node of $\mathcal{T}$ with respect to the value of their depth and, secondarily, with respect to their highest side depth. The rationale for this sort will be clear later.

Embedding-Construction. We perform a bottom-up traversal of $\mathcal{T}$. We check if a non-planarity condition is verified for the current node μ, and in this case we return μ, which is a node of $\mathcal{T}$, such that the pertinent graphs of its children are c-planar but *pertinent*(μ) is not. Otherwise, we compute a c-planar embedding of skeleton(μ), and compute the value $lsd(e)$ for the virtual edge e which represents μ in the skeleton of the parent μ' of μ. Finally, we construct the c-planar embedding of the whole graph by means of a top-down traversal of $\mathcal{T}$.

5.3 The Preprocessing Phase

The depth of each edge is computed in constant time with a lowest common ancestor query performed with the data structure in [27]. The **SPQR-tree Decomposition** step can be performed in linear time [23].

In the **Skeleton-Labelling** step, we perform a bottom-up traversal of $\mathcal{T}$. Let μ be the current node. Based on the values of $d(e)$ and $hsd(e)$ of the edges of skeleton(μ), we compute the values of $d(e')$ and $hsd(e')$ for the virtual edge e' which represents μ in the skeleton of its parent μ'. The value of $hsd(e')$ is the minimum of the highest side depth of the edges of μ. It is easy to see that if μ is an S-node (P-node), $d(e')$ is the minimum (maximum) of the depths of the edges of μ. If μ is an R-node, the computation of $d(e')$ requires a more detailed analysis of skeleton(μ).

Lemma 11 *Let μ be an R-node and let MST be a maximum spanning tree of skeleton(μ), where the edges are weighted with their depth. The depth of the path with maximum depth between the poles of μ is the minimum depth of the edges in the unique path p in MST between the poles of skeleton(μ).*

Proof: By definition the depth $d(p)$ is equal to $d(lcc(p))$, i.e., the minimum depth of its edges. Let e be an edge of p with depth $d(e) = d(p)$. Suppose, for contradiction, that there is a second path p' with $d(p') > d(e) = d(p)$. All edges in p' have depth greater of $d(e)$. When e is removed, MST splits into two trees T_u and T_v, one containing the pole u and the other containing the pole v. Each vertex of skeleton(μ) either falls into T_u or into T_v. Since p' connects u with v it necessarily contains an edge e' which joins a vertex in T_u with a vertex in T_v. If e' is chosen to replace e, T_u and T_v are joined into tree T, which has weight greater than MST, contradicting the hypothesis that MST is the maximum spanning tree. $\square$

Since skeleton(μ) is planar and weighted with integer values, a maximum spanning tree can be constructed in linear time (see for example [4, 26]) with respect to the size of skeleton(μ). Hence, because of Lemma 11 the whole **Skeleton-Labelling** step can be performed in linear time.

The **Edges-Sorting** step requires special care. In fact, if we performed a separate counting sort for each P node, since there are instances where the depth has $O(n)$ values, where n is the number of vertices of G, in the worst case we spent quadratic time. Hence, we do the following. First, we construct a unique set E_P of the virtual edges of all the P nodes, each e labelled with $d(e)$, $hsd(e)$, and with its P node. Second, we perform a counting sort of E_P with respect to $hsd(e)$. Third, we perform a second counting sort with respect to $d(e)$ considering the virtual edges in the order obtained by the first sort. At this point we have that the elements of E_P are sorted according to the value of their depth and, secondarily, with respect to their highest side depth. Finally, we scan E_P and distribute the edges in their proper skeletons. All this requires linear time.

5.4 The Embedding-Construction Phase

In the **Embedding-Construction** phase we first perform a bottom-up traversal of $\mathcal{T}$ in which the c-planarity conditions are verified for each node μ and $\mathcal{T}$ is decorated with suitable embedding descriptors. Secondly, we perform a top-down traversal of $\mathcal{T}$ producing a c-planar embedding for graph G taking into account the values computed for each node μ of $\mathcal{T}$.

Let μ be the current node in the bottom-up traversal of $\mathcal{T}$, let u and v be its poles (assumed arbitrarily ordered at the beginning of the computation), and let e' be the virtual edge which represents μ in the skeleton of its parent μ' and let $high(e')$ be a label associated to it. Suppose *skeleton*(μ) has been embedded and let Γ_μ be its c-planar embedding. We denote *right* (*left*) the side that remains on the right (left) hand when traversing clockwise the external face of Γ_μ from v to u. When computing Γ_μ we assign to $high(e')$ a value in $\{right, left\}$ which denotes which one between the right and left sides of Γ_μ corresponds in *pertinent*(μ) to a path containing an edge e with $d(e) = hsd(e')$. Hence, when processing node μ', we use $high(e')$ to compute the Boolean value of $flip(\mu)$, that specifies if Γ_μ has to be reversed when inserted into $\Gamma_{\mu'}$ in the final top-down traversal.

Provided that the conditions stated in Theorem 2 hold for node μ, we compute an embedding Γ_μ of $skeleton(\mu)$ (if more than one embedding is possible) and the values $flip(\mu_1), \ldots, flip(\mu_k)$ for its children nodes $\mu_1, \ldots, \mu_k$, in such a way to minimize $lsd(e')$. In the following it is specified how S, P and R nodes are processed.

5.4.1 Embedding Construction for S Nodes.

If μ is an S-node skeleton(μ) has a fixed embedding. We set $flip(\mu_1), \ldots, flip(\mu_k)$ so that the corresponding $high(e_1), \ldots, high(e_k)$ are turned towards the same side of Γ_μ, say *right*. Consequently, the left side has minimum depth $lsd(e') = \min_i lsd(e_i)$.

5.4.2 Embedding Construction for R Nodes.

Suppose μ is an R node, with children $\mu_1, \ldots, \mu_k$. Let Γ_μ be the (unique) embedding of $skeleton(\mu)$.

We have to test the c-planarity of Γ_μ, and to verify that for each edge e of $skeleton(\mu)$ incident to two faces f_1 and f_2 of Γ_μ, with $d(f_1) \leq d(f_2)$, if $d(f_1) \leq hsd(e)$ and $d(f_2) \leq lsd(e)$ (see Theorem 2).

Consider the plane graph G^* obtained from Γ_μ by splitting each edge e of Γ_μ with a vertex of depth $d(e)$. It is easy to see that the embedding of $skeleton(\mu)$ is c-planar if and only if G^* is c-planar.

In order to test the c-planarity of a c-connected clustered graph $C(G, T)$, where G has a fixed embedding Γ, we rely on Theorem 1. The statement of Theorem 1 requires to check every cycle of G in order to prove the c-planarity of Γ. This, of course, is not efficient, since we have an exponential number of cycles in a plane graph. Observe, however, that the possible values of $ac(c)$ are as many as the nodes of T. Hence, Theorem 1 can be reformulated as follows:

Lemma 12 *An embedding Γ of a c-connected clustered graph $C(G, T)$ is c-planar if and only if there is no node α of T such that $G(\alpha)$, induced by the vertices in α, contains a cycle c that encloses an edge that is not in $G(\alpha)$.*

Let $C(G, T)$ be a c-connected clustered graph where $G(V, E)$ is embedded, let d_{max} be the height of T, and let $D(V', E')$ be the dual graph of G. For each $e \in E'$, weight e with the depth of the corresponding primal edge. For each integer $i \in [0, d_{max}]$, we define the *i-restricted dual* D_i as the subgraph of D containing only edges with weight at most i and no isolated vertex.

Theorem 5 *Let $C(G, T)$ be a c-connected clustered graph and let d_{max} be the height of T. An embedding Γ of G is c-planar if and only if:*

1. *for each integer $i \in [0, d_{max}]$, graph D_i is connected and*
2. *an edge e_r of the root of T is on the external face.*

Proof: First, we prove the necessity of Conditions 1 and 2. Suppose that no edge of the root of T is on the external face of Γ. By Property 3 there is at least one edge e_r of the root of T in G. Hence, the lowest common cluster of the edges on the external face includes edge e_r, and Theorem 1 applies. Suppose that the graph D_k is not connected for a depth k in $[0, d_{max}]$. Since by definition D_k has no isolated vertex, each connected component of D_k contains at least one edge. Denote with C_r the connected component containing an edge e_r on the external face and denote with e' an edge contained into a connected component $C' \neq C_r$. Consider all edges of D attached to a vertex of C' which are not in C'. These edges are not in D_k and the corresponding edges of G form a cycle c. By Property 2, we have that edges in c can not be shared between two clusters of level k. Hence, there exists a cluster α of level k containing the cycle c which separates edges e_r and e', not belonging to T_α. Since e_r is on the external face, e' is enclosed by c and Lemma 12 applies.

On the contrary, suppose that the embedding Γ is not c-planar. We show that both Conditions 1 and 2 can not be verified. By Lemma 12 there exists a node α of T such that the subtree T_α contains a cycle c that encloses an edge e which is not in T_α. Consider a path p connecting e to c. By Property 2, p has an edge e', enclosed in c, that belongs to a proper ancestor of α. By Condition 2 and by the fact that e_r is not part of c, we have that e_r is not enclosed by c. Hence, each path of D connecting the two edges corresponding to e_r and e' uses at least one edge corresponding to an edge of c. It follows that D_k is not connected. □

A result similar to Theorem 5 has been presented in [9]. We have the following lemma.

Lemma 13 *Let G be an embedded planar graph, let D be its dual with edges weighted with the depths of the corresponding edges of G. Each i-restricted dual D_i, with $i \in [0, d_{max}]$, is connected if and only if the minimum spanning tree $mST(D)$ of D, rooted at any vertex v_r of D_0, is such that edges of non-decreasing weights are encountered when traversing each path p from v_r to a leaf.*

Proof: First observe that the i-restricted duals D_i, for $i \in [0, d_{max}]$, are the subgraphs of D restricted to the faces and the edges with weight less or equal than i, where each face is given the minimum weight of its incident edges. Also, observe that a weighted graph H is connected if and only if it admits a (minimum) spanning forest $mSF(H)$ which is a single (minimum) spanning tree $mST(H)$. Therefore, in order to check if each D_i is connected we could test whether it admits a minimum spanning tree $mST(D_i)$. Further, since we weighted the edges of D with the depth of the corresponding edges of G, we have that $mSF(D_i)$ is a subgraph of $mST(D_{i+1})$.

If $mSF(D_i)$ is not connected for some i then each path in D_k connecting two nodes on two different components of $mSF(D_i)$ uses at least one edge of weight greater than i. Hence, all paths connecting v_r to a node v that belongs to a different component (tree) of $mSF(D_i)$ have at least one edge with weight greater than i. It follows that the minimum weight path between v_r and v is not monotonically non-decreasing. Suppose now that $mST(D_k)$ has a path p from v_r to a leaf which is not monotonically non-decreasing, i.e., p contains at least a sequence of edges of weight j preceded by edge e_1 with weight $w_1 < j$ and followed by edge e_2 with weight $w_2 < j$. Let i be the maximum between w_1 and w_2. Since $mSF(D_i)$ is a subgraph of $mST(D_k)$, we have that $mSF(D_i)$ contains e_1 and e_2, but does not contain the path p, hence it is not connected. □

The conditions of Lemma 13 can be used to check the c-planarity of the embedding of the plane graph G^* in linear time. Let D^* be the dual of G^*. We compute a minimum spanning tree $mST(D^*)$ of D^*. As D^* is planar, $mST(D^*)$ can be constructed in $O(n^*)$, where n^* is the number of nodes of D^* [4, 26]. Then, we easily check in $O(n^*)$ time that the depths are monotonically non-increasing when traversing $mST(D^*)$ from the root to the leaves.

Consider each children μ_i corresponding to e_i. Edge e_i is incident to two faces, f_1 and f_2 for which we assume w.l.o.g. $d(f_1) \leq d(f_2)$. If $d(f_1) > hsd(e)$ or $d(f_2) > lsd(e)$ the algorithm fails since the graph is not c-planar. The value of $high(\mu_i)$ identifies one of the two faces of e_i, we call it f_{high}. We distinguish two cases: (i) f_{high} is an internal face of Γ_μ. If $f_1 = f_{high}$ then we set $flip(\mu_i) = false$, otherwise $flip(\mu_i) = true$. (ii) f_{high} is the external face. We preferentially embed the lowest side into an internal face. Namely, let f_{low} be the opposite face of f_{high} with respect of e_i. If $d(f_{low}) \leq hsd(e)$ then $flip(\mu_i) = true$ otherwise $flip(\mu_i) = false$. This can be done in linear time.

We compute $lsd(e')$ and $high(e')$ in the following way. We consider the ordered split pair $\{u, v\}$ of e' and we call b_r (b_l) the path on the external face of Γ_μ connecting u to v clockwise (counterclockwise). For each edge e_i on b_r (b_l), let $w_{r,i}$ ($w_{l,i}$) be the depth of the side of e_i to be turned towards the external face according to $flip(e_i)$ computed above and $d_r = \min_i w_{r,i}$ ($d_l = \min_i w_{l,i}$). If $d_l < d_r$, we set $lsd(e') = d_r$ and $high(e') = left$ otherwise we set $lsd(e') = d_l$ and $high(e') = right$. Observe that, the procedure according to which $flip(\mu_i)$ are computed assures that the embedding described is one with maximum value of $lsd(e')$ among the possible c-planar embeddings of $pertinent(e')$.

5.4.3 Embedding Construction for P Nodes

If μ is a P node, we have to test the conditions stated in Theorem 2 for P nodes. If all the conditions hold, we construct a c-planar embedding for $skeleton(\mu)$ which maximizes the value of $lsd(e')$, otherwise the graph is not c-planar. Thanks to the **Preprocessing** phase, we have a list $I(\mu)$ where all the virtual edges of $skeleton(\mu)$ appear ordered with respect to the $\preceq_e$ relationship defined as follows: an edge e_1 *precedes* e_2 ($e_1 \preceq_e e_2$) if $d(e_1) > d(e_2)$ or if $d(e_1) = d(e_2)$ and $hsd(e_1) \geq hsd(e_2)$.

Condition (a) of Theorem 2 asks to check that skeleton(μ) does not contain three pairwise incompatible edges. This can be done by checking the bipartiteness of the incompatibility graph in a simple constructive way based on Lemma 14.

Let e_1 be the first element of $I(\mu)$. Condition (b) of Theorem 2 asks to test for each edge $e \in I(\mu)$, with $e \neq e_1$, if $d(e) = lsd(e)$. Also, Condition (b) asks to test for each edge $e \in I(\mu)$, with $e \neq e_1$, if $d(e) \leq lsd(e_1)$. All these tests can be easily done in time linear in the size of skeleton(μ).

The construction of the embedding of $skeleton(\mu)$ consists of the computation of the order of the edges of μ. Namely, the proof of Theorem 2 ensures that a c-planar embedding of $skeleton(\mu)$ is such that edges are ordered into two sequences $I_L = \langle e_{l_1} \succeq_e e_{l_2} \succeq_e \ldots \succeq_e e_{l_p} \rangle$ and $I_R = \langle e_{r_1} \preceq_e e_{r_2} \preceq_e \ldots \preceq_e e_{r_q} \rangle$, each one composed by compatible edges. If the incompatibility graph is bipartite we will be able to construct I_L and I_R. Further, since we want to maximize the value of $lsd(e')$, we search for a particular pair I_L and I_R such that the difference between $\max_{e \in I_L} hsd(e)$ and $\max_{e \in I_R} hsd(e)$ is maximized.

The computation of I_L and I_R requires the use of the following lemma.

Lemma 14 *Let I be a sequence of virtual edges ordered with respect to the $\preceq_e$ relationship, such that edges in I are pairwise compatible. Suppose $e \notin I$ is an edge following all edges in I with respect to the $\preceq_e$ relationship. If e is compatible with the last edge in I then e is compatible with all edges in I.*

Proof: Let e_{last} be the last edge in I. Since e is compatible with e_{last} and $e_{last} \preceq_e e$, we have that $d(e) \leq hsd(e_{last})$. Since all the edges in I are pairwise compatible, we also have that $d(e_{last})$ is less or equal than the highest side depth of all edges in I. It follows that $d(e)$ is less or equal than the highest side depth of each edge in I, and therefore e is compatible with all edges in I. □

We build two sequences I_1 and I_2 by inserting one by one the edges of $I(\mu)$ into one of them. Namely, we start by inserting e_1 in I_1. Let e_i be the current edge and let $e_{1,last}$ and $e_{2,last}$ be the last inserted elements of I_1 and I_2, respectively. If e_i is incompatible with both $e_{1,last}$ and $e_{2,last}$ we conclude that the incompatibility graph is not bipartite. If e_i is incompatible with the last element of one of the two sequences we insert it into the other sequence. Otherwise, if e_i is compatible with both $e_{1,last}$ and $e_{2,last}$, then we insert it into the sequence containing $\min\{hsd(e_{1,last}), hsd(e_{2,last})\}$. We set I_L as the reverse of I_1 and $I_R = I_2$.

Since we insert an edge e_i into a sequence only if e_i is compatible with the last element of the sequence, and the sequences are ordered with respect to the $\preceq_e$ relationship, Lemma 14 ensures that both I_L and I_R contain pairwise compatible edges. If an edge e is compatible with both the sequences, inserting it into the sequence with smaller value of highest side depth on the last edge guarantees that the difference between $\max_{e \in I_L} hsd(e)$ and $\max_{e \in I_R} hsd(e)$ is maximized. In fact, the following property holds:

Property 8 *Let I be a sequence of edges ordered with respect to the $\preceq_e$ relationship, such that edges in I are pairwise compatible. The last edge e_{last} in I has $hsd(e_{last}) = \max_{e \in I}(hsd(e))$.*

According to the construction rules provided in the sufficiency proof of the characterization given in Subsection 3.1, for each edge $e_i \in I_L$, we set $flip(e_i) = true$ if $high(e_i) = right$, and $flip(e_i) = false$ otherwise. Conversely, for each edge $e_i \in I_R$, we set $flip(e_i) = true$ if $high(e_i) = left$, and

$flip(e_i) = false$ otherwise. Finally, the value of $lsd(e')$ is maximum between $hsd(e_{l_1})$ and $lsd(e_{r_q})$. All the operations performed on a P node can be clearly executed in linear time.

Finally, we compute the c-planar embedding of G. We start with the current embedding equal to the skeleton of the child of the root of $\mathcal{T}$ and proceed by means of a top-down traversal of $\mathcal{T}$. For each node μ of $\mathcal{T}$ with children $\mu_1, \ldots, \mu_k$, the embeddings of $skeletons(\mu_i)$ are merged into the current embedding. If $flip(\mu_i) = true$ the embedding is flipped before the merge operation. This computation is linear since each skeleton is flipped at most once.

The whole algorithm is summarized in Figures 6, 7, and 8. From the above discussion we can state the following theorem.

Theorem 6 *Given a c-connected clustered graph $C(G,T)$, such that G is biconnected, the above described algorithm tests the c-planarity of C, and, if C is c-planar, computes a c-planar embedding of C in linear time.*

6 Testing and Embedding Algorithm: General Case

In this section we extend the algorithm presented in Section 5 to the case of c-connected clustered graph whose underlying graph is planar and simply connected.

The following lemmas permit to state the correctness of the algorithm.

Lemma 15 *Let $C(G,T)$ be a c-planar clustered graph and let $\mathcal{B}$ be the block-cutvertex tree of G. Let α be a cutvertex of $\mathcal{B}$ with parent μ and let $\{u,\alpha\}$ be a split pair of μ. Suppose that in a c-planar embedding of C pertinent(α) appears in an internal face of the embedding of pertinent(u,α). There exists a c-planar embedding of C such that pertinent(α) is embedded in the external face of the embedding of pertinent(u,α).*

Proof: Suppose that there is no c-planar embedding of G unless $pertinent(\alpha)$ is inside $pertinent(u,\alpha)$. This implies that in any drawing of C with $pertinent(\alpha)$ embedded outside $pertinent(u,\alpha)$ at least one of the following two conditions is verified: (i) there is a cycle c of depth $d(c) > d(pertinent(u,\alpha))$ enclosing $pertinent(u,\alpha)$; (ii) there are two cycles c_1 and c_2 of depth greater than $d(pertinent(\alpha))$ passing through $pertinent(u,\alpha)$ and enclosing the two faces outside $pertinent(u,\alpha)$ (see the dotted and dashed cycles of Fig. 5). In case (i), since c encloses both the faces outside $pertinent(u,\alpha)$, there can not be a c-planar embedding with $pertinent(\alpha)$ inside $pertinent(u,\alpha)$. In case (ii), from Fig. 5 it is apparent that the parts of the two cycles c_1 and c_2 outside $pertinent(u,\alpha)$ form a cycle enclosing $pertinent(u,\alpha)$. Hence, there can not be a c-planar embedding with $pertinent(\alpha)$ inside $pertinent(u,\alpha)$. $pertinent(u,\alpha)$.
□

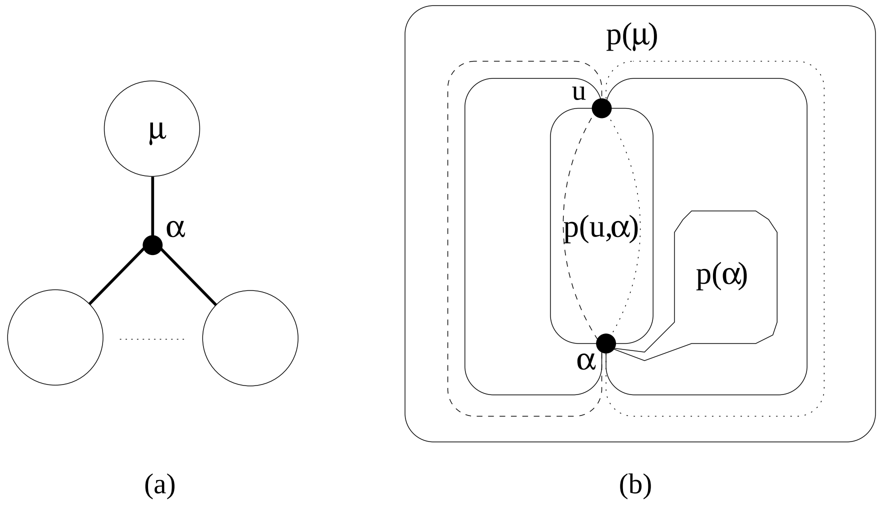

Figure 5: (a) A portion of the BC-tree for the proof of Lemma 15. (b) The relationships between three subgraphs $pertinent(\mu)$, $pertinent(u,\alpha)$, and $pertinent(\alpha)$, denoted $p(\mu)$, $p(u,\alpha)$ and $p(\alpha)$, respectively.

Lemma 16 *Let $C(G,T)$ be a c-planar clustered graph and let $\mathcal{B}$ be the block-cutvertex tree of G. Let α be a cutvertex of $\mathcal{B}$ with children μ_1 and μ_2. Suppose that in a c-planar embedding of C $pertinent(\mu_2)$ appears in an internal face of the embedding of $pertinent(\mu_1)$. There exists a c-planar embedding of C such that $pertinent(\mu_2)$ appears in the external face of the embedding of $pertinent(\mu_1)$.*

Proof: Suppose that there is no c-planar embedding of G unless $pertinent(\mu_2)$ is not placed inside a face of $pertinent(\mu_1)$. This implies that in any drawing of C with $pertinent(\mu_2)$ embedded outside $pertinent(\mu_1)$ there is a cycle c of depth $d(c) > d(pertinent(\mu_2))$ enclosing $pertinent(\mu_2)$. Since c necessarily encloses μ_1 and μ_2, there can not be a c-planar embedding of C such that $pertinent(\mu_2)$ is placed inside a face of $pertinent(\mu_1)$. □

We now show a linear-time algorithm for testing and embedding a general c-connected clustered graph.

BC-tree Decomposition. First, for each edge e of G we compute $d(e)$. Second, we compute the BC-tree $\mathcal{B}$ of G and root $\mathcal{B}$ to a block ν containing an edge $\overline{e}$ such that $d(\overline{e}) = 0$.

BC-tree Labelling. We traverse $\mathcal{B}$ bottom-up and compute for each cutvertex ρ_i the depth of pertinent(ρ_i). This is done by taking the minimum depth of the pertinents of the children blocks of ρ_i.

Block Preprocessing We perform a second bottom-up traversal of $\mathcal{B}$ and execute on each block μ a variation of the **Preprocessing** phase for biconnected graphs, where the sorting phase is factored out and cut-vertices are considered. Namely, for each block μ the following two steps are performed.

SPQR-tree Decomposition. First, we compute an SPQR-tree $\mathcal{T}_\mu$ rooted at any edge e_r whose depth is the minimum depth of the block.

Skeleton Labelling. For each node σ in $\mathcal{T}_\mu$, consider each edge e of $skeleton(\sigma)$ such that $pertinent(e)$ is a single edge e'. We label e such that $hsd(e) = lsd(e) = d(e) = d(e')$. We perform a bottom-up traversal of $\mathcal{T}_\mu$ in order to label each virtual edge e with $d(e)$ and $hsd(e)$. Let e be a virtual edge of any skeleton. The value of $d(e)$ is computed with the same operations used for biconnected graphs. Let $\rho_1, \ldots, \rho_k$ be the cutvertices of μ contained in $skeleton(e)$ that are not poles of e, possibly comprehensive of the parent of μ. The value of $hsd(e)$ is the minimum of the highest side depths of the edges of $skeleton(e)$ and the depths of $pertinent(\rho_i)$.

This implies that the parent cutvertex of μ is adjacent to a face f with lowest depth in the computed embedding for μ. As stated in 4 the external face can be changed so that the parent cutvertex is incident to the external face and hence the condition of Theorem 4, modified as in Section 5.1, is verified.

Edges Sorting. We simultaneously sort the edges of all P nodes of all the computed SPQR-trees with respect to the value of their depth, and secondarily with respect to their highest side depths. We use a strategy analogous to that used for biconnected graphs in order to preserve the linearity of this algorithmic step.

Block Embedding Construction. For each block μ we consider its SPQR-tree $\mathcal{T}_\mu$ and perform a bottom-up traversal of it. We check if a non-planarity condition (see Theorem 2) is verified for the current node σ, possibly computing a c-planar embedding of $skeleton(\sigma)$ and the value of $lsd(e)$ for the virtual edge e which represents σ in the skeleton of its parent σ'.

In the case σ is a P node, the test of the c-planarity conditions, the computation of the embedding of $skeleton(\sigma)$, and the computation of $lsd(e)$ follow the same rules described for biconnected graphs (see Section 5).

In the case σ is an S node, we proceeds as for biconnected graphs. Plus, consider each vertex ρ of $skeleton(\sigma)$ which is also a cutvertex and is not a pole of σ. All the blocks that are children of ρ in $\mathcal{B}$ are embedded in the side where all the highest sides of the children of σ in $\mathcal{T}$ are embedded. The correctness of this approach is implied by Lemmas 15 and 16.

In the case σ is an R node, the existence of cutvertices in $skeleton(\sigma)$ must be taken into account. Besides the tests performed for the biconnected case we have to make sure that the second condition of Theorem 4, modified as in Section 5.1, is verified. Namely, each cutvertex ρ that is not a pole of σ must be incident to a face f of $skeleton(\sigma)$ with $d(f)$ less or equal than the depth of $pertinent(\rho)$. When choosing f, an internal face is always preferred if it respects this condition. All blocks that are children of ρ in $\mathcal{B}$ are embedded in f. The correctness of this approach is implied by Lemmas 15 and 16. If such a face does not exist the algorithm fails since the graph is not c-planar.

We compute $flips(\cdot)$ of the children of σ as for biconnected graphs. When computing $lsd(e')$ and $high(e')$ we proceed as for the biconnected graphs but for the computation of d_l and d_r, see Section 5 **Embedding Construction for R Nodes**. Namely, the computation of d_r (d_l) must take into account the depth of the cutvertices in b_r (b_l) that have their blocks embedded in the external face of $skeleton(\sigma)$.

Observe that, as in the biconnected case, the adopted procedure assures that the embedding described by $flip(\cdot)$ and by the choices on the cutvertices, is one with minimum value of $lsd(e')$ among the possible c-planar embeddings of $pertinent(e')$.

In the case σ is the unique child of the root of $\mathcal{T}_\mu$ with poles u and v, besides the regular operations described above, we check if u or v are cutvertices and embed all their blocks in the external face.

The reporting of the embedding of μ is performed as for biconnected graphs.

Block Re-rooting and Merging. We consider the computed embedding Γ_μ of each block μ of $\mathcal{B}$ and we adopt as external face of Γ_μ a face with minimum depth incident to the parent cutvertex of μ. We merge together the obtained embeddings of the blocks.

The whole algorithm is summarized in Figure 9. Due to the above description the following theorem holds.

Theorem 7 *The c-planarity of a c-connected clustered graph can be tested, and possibly a c-planar embedding can be built, in linear time.*

7 Comparison with Previous Results

Few algorithms have been described in the literature to test the c-planarity of a c-connected clustered graph and possibly compute a c-planar embedding of it.

The algorithm by Feng, Cohen, and Eades [20, 19], which has quadratic-time complexity, has been regarded for a long time as the only algorithm available for implementation. It was noted, though, that the results by Lengauer [25], based on graph grammars, could be used to compute a c-planar embedding of a graph in linear time with respect to the grammar size [9]. However, it is quite easy to find examples of clustered graphs whose description in terms of graph grammars is quadratic in the number of the vertices.

In [9] Dahlhaus presented an algorithm for c-planarity testing and embedding of c-connected clustered graphs, both in the case of biconnected and of simply-connected underlying graphs. Dahlhaus' algorithm has several differences and similarities with respect to the algorithms described in Sections 5 and 6 that we are going to discuss hereunder.

The main similarity is that both algorithms are based on the SPQR and BC-tree decomposition of the graph. However, in [9] SPQR-trees are not explicitly mentioned, the decomposition is improperly referred to as a "graph grammar" decomposition, and the distinction between a structural component and the corresponding pertinent graph is sometimes blurred.

The main difference, instead, is that the algorithm in [9] is based on a characterization of c-planar embeddings, while the approach described in this paper is based on a characterization of c-planar graphs. Such a characterization, which relies on well-established and solid graph theoretic concepts, is provided in Section 3. A characterization similar to that used in [9] is provided, instead, by Theorem 5.

Our approach allows us to directly attribute a non-c-planarity to the interplay between the clusters hierarchy and the structure of the graph, rather than to a particular embedding of it and our definitions are always referred to a graph and its SPQR or BC-tree decomposition (see, for example, the role played by the concept of c-planar skeleton). Also, a beneficial consequence of our characterization is that a c-planar embedding of the graph can be incrementally built until the characterization conditions hold. Otherwise, the algorithm terminates reporting the non-c-planarity of the clustered graph with a clear identification of the reason for its failure. In contrast, given a clustered graph, the algorithm described in [9] always computes a (possibly non-c-planar) embedding of it. Subsequently, a c-planarity test has to be performed on the produced embedding in order to check whether it is actually c-planar. This strategy relies on the assumption that if the clustered graph admits some c-planar embeddings, one of them is indeed found by the algorithm. However, the correctness of this assumption is not proved.

Although some algorithmic steps are only sketched in [9], it is possible to recognize several other points in which the two algorithms differ and in which the approach described in this paper is, in our opinion, easier to understand and to implement. For example, when labeling a virtual edge with its depth, we simply compute a maximum spanning tree of the skeleton of the corresponding component (see Section 5.3 and Lemma 11). In [9], the concept that plays a role analogous to the depth of a virtual edge is the "axis weight" of a component, which is computed by means of a circuitous process involving a spanning tree of the whole graph and a classification of the components into "social" and "introverted" ones.

Also, the embedding of a P-node is performed in [9] by means of a complex machinery requiring the computation of a spanning forest of a suitable graph. Our approach immediately detects a non-c-planarity and then, if it is possible, computes a c-planar embedding with an intuitive greedy algorithm (see Lemma 14 and the following description).

Further, note that the algorithm in [9] has quadratic-time complexity if implemented as described in the paper. In fact, the computation of the embedding of each P-node requires to sort the parallel components based on integer labels which, in the worst case, range in $[1, n]$. Since the number of P-nodes is $O(n)$, this operation clearly yields an overall $O(n^2)$ time complexity. Conversely, our algorithm performs a counting sort of all virtual edges corresponding to P-nodes only two times in the preprocessing phase, preserving linearity. Such a strategy could be incorporated into the algorithm in [9] amending its complexity problem.

Apart from these algorithmic differences, the solid theoretic foundations of the algorithm described in this paper allows us to focus on structural properties rather than on implementation codings, providing new insight into the interplay between the inclusion tree and the graph decomposition trees. For example, the "weight" of a cluster is defined in [9] as the number of vertices of the graph contained into the cluster, and is extensively used in all main statements. On the contrary, we base our characterizations on the natural concept of inclusion between clusters, which is accounted for by the lowest common ancestor relationship. In Section 5.1, we show that, since when we need to compare two clusters one is the ancestor of the other, the concept of lowest common ancestor can be replaced, for the convenience of the implementation only, by any function of a large family, of which the "weight" is only an example (we use the depth, instead).

8 Conclusions and Open Problems

In this paper we present the first characterization of c-planarity for a c-connected clustered graph $C(G, T)$, both in the case that G is biconnected and in the general case. Based on such a characterization, we provide a linear-time algorithm to test the c-planarity of C. If C is non-c-planar, our algorithm identifies a structural element responsible for non-c-planarity. The algorithm is fully described in terms of elementary steps and is easily implementable in linear time.

One of the most interesting open problems in this field is that of stating the complexity of the c-planarity testing for non-connected clustered graphs. However, in our opinion, the c-planarity of c-connected clustered graphs still deserves further investigation. For example, it would be really interesting to provide in this context a characterization in terms of obstructive patterns, analogous to $K_{3,3}$ and K_5 in the Kuratowski's theorem. Also, the investigation of c-planarity with additional constraints may be interesting from an applicative point of view.

References

[1] T. C. Biedl. Drawing planar partitions III: Two constrained embedding problems. Tech. Report RRR 13-98, RUTCOR Rutgen University, 1998.

[2] T. C. Biedl, M. Kaufmann, and P. Mutzel. Drawing planar partitions II: HH-Drawings. In J. Hromkovic and O. Sýkora, editors, *Workshop on Graph-Theoretic Concepts in Computer Science (WG '98)*, volume 1517, pages 124–136. Springer, 1998.

[3] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, and D. Wagner. On finding graph clusterings with maximum modularity. In A. Brandstädt, D. Kratsch, and H. Müller, editors, *Proc. Graph-Theoretic Concepts in Computer Science (WG '07)*, volume 4769 of *LNCS*, pages 121–132. Springer, 2007.

[4] D. Cheriton and R. E. Tarjan. Finding minimum spanning trees. *SIAM J. Comput.*, 5(10):724–742, 1974.

[5] S. Cornelsen and D. Wagner. Completely connected clustered graphs. *J. Discrete Algorithms*, 4(2):313–323, 2006.

[6] P. F. Cortese and G. Di Battista. Clustered planarity. In J. S. B. Mitchell and G. Rote, editors, *Symposium on Computational Geometry (SoCG '05)*, pages 32–34. ACM, 2005.

[7] P. F. Cortese, G. Di Battista, M. Patrignani, and M. Pizzonia. Clustering cycles into cycles of clusters. In J. Pach, editor, *Proc. Graph Drawing 2004 (GD '04)*, volume 3383 of *LNCS*, pages 100–110. Springer, 2004.

[8] P. F. Cortese, G. Di Battista, M. Patrignani, and M. Pizzonia. On embedding a cycle in a plane graph. In P. Healy and N. Nikolov, editors, *Proc. Graph Drawing 2005 (GD '05)*, volume 3843 of *LNCS*, pages 46–60. Springer, 2005.

[9] E. Dahlhaus. Linear time algorithm to recognize clustered planar graphs and its parallelization. In C. L. Lucchesi and A. V. Moura, editors, *Proc. Latin American Theoretical INformatics (LATIN '98)*, volume 1380 of *LNCS*, pages 239–248. Springer, 1998.

[10] G. Di Battista, W. Didimo, and A. Marcandalli. Planarization of clustered graphs. In P. Mutzel, M. Jünger, and S. Leipert, editors, *Proc. Graph Drawing 2001 (GD '01)*, LNCS, pages 60–74. Springer, 2001.

[11] G. Di Battista, G. Drovandi, and F. Frati. How to draw a clustered tree. In F. K. H. A. Dehne, J.-R. Sack, and N. Zeh, editors, *Proc. Algorithms and Data Structures (WADS '07)*, volume 4619 of *LNCS*, pages 89–101. Springer, 2007.

[12] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing.* Prentice Hall, Upper Saddle River, NJ, 1999.

[13] G. Di Battista and F. Frati. Efficient c-planarity testing for embedded flat clustered graphs with small faces. In S.-H. Hong, T. Nishizeki, and W. Quan, editors, *Proc. Graph Drawing 2007 (GD '07)*, volume 4875 of *LNCS*, pages 291–302. Springer, 2008.

[14] G. Di Battista and R. Tamassia. On-line planarity testing. *SIAM J. Comput.*, 25:956–997, 1996.

[15] C. A. Duncan, M. T. Goodrich, and S. G. Kobourov. Planarity-preserving clustering and embedding for large planar graphs. *Comput. Geom.*, 24(2):95–114, 2003.

[16] P. Eades, Q.-W. Feng, X. Lin, and H. Nagamochi. Straight-line drawing algorithms for hierarchical graphs and clustered graphs. *Algorithmica*, 44(1):1–32, 2006.

[17] P. Eades, Q.-W. Feng, and H. Nagamochi. Drawing clustered graphs on an orthogonal grid. *J. Graph Algorithms Appl.*, 3(4):3–29, 1999.

[18] S. Even. *Graph Algorithms.* Computer Science Press, Potomac, Maryland, 1979.

[19] Q. W. Feng, R. F. Cohen, and P. Eades. How to draw a planar clustered graph. In D.-Z. Du and M. Li, editors, *Proc. Computing and Combinatorics (COCOON '95)*, volume 959 of *LNCS*, pages 21–30. Springer, 1995.

[20] Q. W. Feng, R. F. Cohen, and P. Eades. Planarity for clustered graphs. In *Proc. European Symposium on Algorithms (ESA '95)*, volume 979 of *LNCS*, pages 213–226. Springer, 1995.

[21] M. T. Goodrich, G. S. Lueker, and J. Z. Sun. C-planarity of extrovert clustered graphs. In P. Healy and N. Nikolov, editors, *Proc. Graph Drawing 2005 (GD '05)*, volume 3843 of *LNCS*, pages 211–222. Springer, 2005.

[22] C. Gutwenger, M. Jünger, S. Leipert, P. Mutzel, M. Percan, and R. Weiskircher. Advances in C-planarity testing of clustered graphs. In S. G. Kobourov and M. T. Goodrich, editors, *Proc. Graph Drawing 2002 (GD '02)*, volume 2528 of *LNCS*, pages 220–235. Springer, 2002.

[23] C. Gutwenger and P. Mutzel. A linear time implementation of SPQR-trees. In J. Marks, editor, *Proc. Graph Drawing 2000 (GD '00)*, pages 77–90. Springer, 2001.

[24] E. Jelínková, J. Kára, J. Kratochvíl, M. Pergel, O. Suchý, and T. Vyskocil. Clustered planarity: Small clusters in Eulerian graphs. In S.-H. Hong, T. Nishizeki, and W. Quan, editors, *Proc. Graph Drawing 2007 (GD '07)*, volume 4875 of *LNCS*, pages 303–314. Springer, 2008.

[25] T. Lengauer. Hierarchical planarity testing algorithms. *J. ACM*, 36(3), 1989.

[26] T. Matsui. The minimum spanning tree problem on a planar graph. *Discrete Appl. Math.*, 58(1):91–94, 1995.

[27] B. Schieber and U. Vishkin. On finding lowest common ancestors: simplification and parallelization. *SIAM J. Comput.*, 17(6):1253–1262, 1988.

C-planarity algorithm for biconnected graphs

input: A c-connected clustered graph $C(G,T)$, where G is a planar biconnected graph

output: A c-planar embedding of G if C is c-planar, a node of $\mathcal{T}$ violating the c-planarity conditions otherwise

```
  Preprocessing Phase
  for all edge $e \in G$ do
    compute $d(e)$, $hsd(e)$, $lsd(e)$
  end for
  compute the SPQR tree $\mathcal{T}$ of $G$, rooted to an edge with $d(e) = 0$
  for all node $\mu$ in $\mathcal{T}$ in post-order traversal do
    let $e'$ be the virtual edge representing $\mu$ in the skeleton of its parent node.
    $hsd(e') = \min_{e \in skeleton(\mu)} hsd(e)$
    if $\mu$ is an $S$ node then
      $d(e') = \min_{e \in skeleton(\mu)} d(e)$
    else if $\mu$ is a $P$ node then
      $d(e') = \max_{e \in skeleton(\mu)} d(e)$
    else if $\mu$ is an $R$ node then
      Compute a Maximum Spanning Tree $MST$ of skeleton($\mu$)
      Let $p$ be the path between the poles in $MST$.
      $d(e') = d(p)$
    end if
  end for
  sort the edges of each $P$ node using a single counting sort.
  Embedding Construction Phase
  for all node $\mu$ in $\mathcal{T}$ in post-order traversal do
    if $\mu$ is an $S$ node then
      for all $e \in skeleton(\mu)$ do
        if $high(e) = left$ then
          $flip(e) = true$
        else
          $flip(e) = false$
        end if
      end for
      $lsd(e') = \min_{e \in skeleton(\mu)} lsd(e)$
      $high(e') = right$
    else if $\mu$ is an $P$ node then
      if ProcessPNode($\mu$,$e'$)=False then
        return $\mu$
      end if
    else if $\mu$ is an $R$ node then
      if ProcessRNode($\mu$,$e'$)=False then
        return $\mu$
      end if
    end if
  end for
  construct the c-planar embedding by performing a top-down traversal of $\mathcal{T}$ and
  considering values of $flip(.)$
  return the embedding of $G$
```

Figure 6: The c-planarity testing and embedding algorithm for c-connected clustered graphs whose underlying graph is biconnected.

```
Procedure ProcessPNode($\mu$,$e'$)
  {The edges of skeleton($\mu$) are already ordered in a list $I(\mu)$}
  let $e_1$ be the first element of $I(\mu)$
  for all $e \neq e_1$ in skeleton($\mu$) do
    if $d(e) \neq lsd(e)$ or $d(e) > lsd(e_1)$ then
      Return False
    end if
  end for
  initialize lists $\overline{I_L} = \{e_1\}$ and $I_R = \{\}$
  for all $e \neq e1$ in skeleton($\mu$) do
    $e_l$=last element in $\overline{I_L}$, $e_r$=last element in $I_R$
    if $e$ is incompatible with $e_l$ then
      if $e$ is incompatible with $e_r$ then
        return False
      else
        append $e$ to $I_R$
      end if
    else if $e$ is incompatible with $e_r$ then
      append $e$ to $\overline{I_L}$
    else
      append $e$ to the list containing $\min\{hsd(e_l), lsd(e_r)\}$
    end if
  end for
  the embedding of skeleton($\mu$) is $I_L I_R$, where $I_L$ is the reverse of $\overline{I}_L$
  for all $e$ in $I_L$ do
    if $high(e) \neq left$ then
      $flip(e) = true$
    end if
  end for
  for all $e$ in $I_R$ do
    if $high(e) \neq right$ then
      $flip(e) = false$
    end if
  end for
  $lsd(e') = \max\{\min_{e \in I_L} hsd(e), \min_{e \in I_R} hsd(e)\}$
  if $hsd(e_l) \leq hsd(e_r)$ then
    $high(\mu) = left$
  else
    $high(\mu) = right$
  end if
  return True
```

Figure 7: Testing and embedding procedure for P-nodes.

```
Procedure ProcessRNode($\mu$,$e'$)
  construct the graph $G^*$ from skeleton($\mu$)
  compute the planar embedding of $G^*$ with the poles on the external face
  compute the dual graph $D$ of $G^*$
  compute the minimum spanning tree $mST$ of $D$
  if mST is non-monotonic then
    return False
  end if
  for all $e$ in skeleton($\mu$) do
    let $f_1$ and $f_2$ be the faces incident to $e$, with $d(f_1) \leq d(f_2)$
    if $hsd(e) < d(f_1)$ or $lsd(e) < d(f_2)$ then
      return False
    else
      Let $f_{high}$ be the face incident to $e$ identified by $high(e)$
      if $f_1$ is the external face AND $hsd(e) \geq d(f_2)$ then
        if $f_1 = f_{high}$ then
          $flip(e) = true$
        else
          $flip(e) = false$
        end if
      else
        if $f_1 \neq f_{high}$ then
          $flip(e) = true$
        else
          $flip(e) = false$
        end if
      end if
    end if
  end for
  let $\{u, v\}$ the ordered split pair of $e'$
  let $b_r$ the path on the external face of skeleton($\mu$) connecting $u$ to $v$ clockwise
  let $b_l$ the path on the external face of skeleton($\mu$) connecting $u$ to $v$ counterclockwise.
  for all $e_i \in b_r$ do
    let $w_{r,i}$ be the depth of the side of $e_i$ to be turned towards the external face
  end for
  for all $e_i \in b_l$ do
    let $w_{l,i}$ be the depth of the side of $e_i$ to be turned towards the external face
  end for
  $d_r = \min_i w_{r,i}$
  $d_l = \min_i w_{l,i}$
  if $d_l < d_r$ then
    $lsd(e') = d_r$
    $high(e') = left$
  else
    $lsd(e') = d_l$
    $high(e') = right$
  end if
  return true
```

Figure 8: Testing and embedding procedure for R-nodes.

```
C-planarity testing and embedding algorithm for connected graphs
input: A c-connected clustered graph $C(G,T)$, where $G$ is a planar graph
output: "True" and a c-planar embedding of $G$ if $C$ is c-planar, "False" otherwise
  Block Preprocessing Phase
  for all edge $e \in G$ do
    Compute $d(e)$, $hsd(e)$, $lsd(e)$
  end for
  compute the BC tree $\mathcal{B}$ of $G$, rooted to a block containing an edge $e$ with $d(e) = 0$
  for all cutvertex $\rho$ in $\mathcal{B}$ in post-order traversal do
    compute the depth of $pertinent(\rho)$
  end for
  for all node $\mu$ in $\mathcal{B}$ in post-order traversal do
    compute the SPQR tree $\mathcal{T}_\mu$ rooted to an edge with minimum depth
    For each non-virtual edge $e \in \mathcal{T}_\mu$ compute $d(e)$, $hsd(e)$, $lsd(e)$
    for all node $\sigma \in \mathcal{T}_\mu$ in post-order traversal do
      compute $d(\sigma)$ as in the biconnected case
      let $\rho_i$ be the cutvertices in $skeleton(\sigma)$ different from the poles
      compute $hsd(\sigma) = \min_i\{hsd(e_i), d(pertinent(\rho_i))\}$, with $e_i \in skeleton(\sigma)$
    end for
  end for
  Sort the edges of each $P$ node of each block with a single counting sort
  Block Embedding Phase
  for all node $\mu$ in $\mathcal{B}$ do
    for all node $\sigma \in \mathcal{T}_\mu$ in post-order traversal do
      let $\rho_i$ be the cutvertices in $skeleton(\sigma)$ different from the poles
      if $\sigma$ is an $S$ node then
        process $\sigma$ as in the biconnected case
        embed the blocks connected to $\rho_i$ in the highest side of $skeleton(\sigma)$
      else if $\sigma$ is an $P$ node then
        process $\sigma$ as in the biconnected case
      else if $\sigma$ is an $R$ node then
        test the condition on $skeleton(\sigma)$ as in the biconnected case
        if each $\rho_i$ is not incident to a face $f$ with $d(f) \leq d(pertinent(\rho_i))$ then
          return False
        else
          embed the blocks of $\rho_i$ in a suitable (possibly internal) face $f$
        end if
        compute the flip for each virtual edge as in the biconnected case
        compute $lsd(\sigma)$ considering the blocks embedded on the external face
        compute $high(\sigma)$ considering the blocks embedded on the external face
      end if
    end for
    construct the embedding $\Gamma_\mu$ of $\mu$ as in the biconnected case
    let $f$ be a face with minimum depth incident to the root cutvertex of $\mu$
    choose $f$ as external face for $\Gamma_\mu$
  end for
  merge the embedding of the blocks
```

Figure 9: The c-planarity testing and embedding algorithm for c-connected clustered graphs