Università degli Studi di Roma Tre
Dipartimento di Informatica e Automazione
Via della Vasca Navale, 79 – 00146 Roma, Italy

C-Planarity of C-Connected Clustered Graphs

Part II – Testing and Embedding Algorithm

P. F. Cortese, G. Di Battista, F. Frati, M. Patrignani, and M. Pizzonia

RT-DIA-110-2006 **June 2006**

Dipartimento di Informatica e Automazione,
Università di Roma Tre,
Rome, Italy.
{cortese,gdb,frati,patrigna,pizzonia}@dia.uniroma3.it

Work partially supported by European Commission - Fet Open project DELIS - Dynamically Evolving Large Scale Information Systems - Contract no 001907 and by "Project ALGO-NEXT: Algorithms for the Next Generation Internet and Web: Methodologies, Design, and Experiments", MIUR Programmi di Ricerca Scientifica di Rilevante Interesse Nazionale.

ABSTRACT

We present a linear time c-planarity testing and embedding algorithm for c-connected clustered graphs. The algorithm is based on a characterization of the clustered planarity given in a companion paper [3]. The algorithm is reasonably easy to implement, since it exploits as building blocks simple algorithmic tools like the computation of lowest common ancestors, of minimum and maximum spanning trees, and of bucket sorts. It also makes use of data structures like the SPQR-trees and the BC-trees. If the test fails it gives a structural certificate of the intrinsic reasons causing the non c-planarity.

1 Introduction

Testing a clustered graphs for c-planarity is a problem of unknown time complexity in the general case [2]. However, there exist three polynomial time algorithms, discussed below, to test the c-planarity of a c-connected clustered graph. (For basic terminology on clustered graphs and c-planarity see [3]).

Feng, Cohen, and Eades presented in [9, 8] a quadratic time algorithm. Their algorithm visits the inclusion tree of the clusters bottom-up, starting from the leaves. Each cluster is tested for planarity with the constraint that the edges to other clusters stay on 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 [11] found a result analogous to the one in [9, 8], 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, such a size can be quadratic in the size of the represented clustered graph.

Dahlhaus [4] 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 on a deep characterization of the c-planar embeddings. The testing algorithm is based on the incremental construction of a certain planar embedding and on a final test that checks if such embedding is c-planar. The work in [4] 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 a new linear time c-planarity testing and embedding algorithm for c-connected clustered graphs. The algorithm is based on a characterization of clustered planarity given in a companion paper [3]. It is reasonably easy to implement, since it exploits as building blocks simple algorithmic tools like the computation of lowest common ancestors, of minimum and maximum spanning trees, and of bucket sorts. It also makes use of data structures like the SPQR-trees and the BC-trees [6, 10] (both in their simple static version). Further, if the test fails, it gives a structural certificate of the intrinsic reasons causing the non c-planarity.

The paper is organized as follows. In Section 2 we provide basic terminology and recall the characterization presented in [3]. In Section 3 we recast this characterization in a form more suitable for an algorithm. In Section 4 we describe a linear time algorithm for testing the c-planarity of c-connected clustered graphs whose underlying graph is biconnected. In Section 5 we extend our algorithm to handle clustered graphs whose underlying graph is simply connected.

2 Background

We assume familiarity with planarity and connectivity of graphs [7]. We also assume familiarity with graph drawing [5]. For a survey on the definitions of clustered graphs, c-planarity, and c-connectivity, and for a definition of SPQR-tree and of BC-tree see

the companion paper [3]. We only recall here a characterization of the c-planarity of c-connected clustered graphs given in [3] and the definitions that are needed to read the characterization.

Given a connected 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'. 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.

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. The *lowest connecting cluster* of e, denoted by $lcc(e)$ is the allocation cluster of the lowest connecting path of 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. We adopt the same definition of lowest connecting cluster also for cycles and faces of $skeleton(\mu)$. Also, for technical reasons we define the lowest connecting cluster of an external face of a skeleton as the root of the inclusion tree T.

An embedding of $skeleton(\mu)$ (where μ is a node of $\mathcal{T}$) is *c-planar* when any cycle c of the edges of $skeleton(\mu)$ does not enclose an edge e of $skeleton(\mu)$ with $lcc(e) \prec lcc(c)$.

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 $ac(hs(\Gamma, s)) \preceq ac(ls(\Gamma, s))$. It can be shown that 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, $hsc(e) = ac(pertinent(e))$.

Also, the value of $ac(ls(\Gamma, s))$ does not depend on the choice of s. Hence, we can define the *lowest side cluster* of Γ $lsc(\Gamma) = ac(ls(\Gamma, s))$ and the *lowest side cluster* of e, $lsc(e) = \max_{\Gamma}\{lsc(\Gamma)\}$. The definitions of $hsc(e)$ and of $lsc(e)$ hold only if $pertinent(e)$ is c-planar. 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.

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 $hsc(e_2) \prec llc(e_1)$; (ii) $lcc(e_1) = lcc(e_2)$, $hsc(e_1) \prec lcc(e_1)$, and $hsc(e_2) \prec llc(e_2)$.

Given a c-connected clustered graph $C(G, T)$, we have an easy characterization of a c-planar embedding for G.

Theorem 1 *[3] A planar embedding Γ of a c-connected clustered graph is c-planar if and only if every cycle c of Γ does not enclose any edge e such that $ac(e) \prec ac(c)$.*

The characterization for biconneced graphs is as follows:

Theorem 2 *[3] 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 $lcc(f_1) \preceq hsc(e)$ and $lcc(f_2) \preceq lsc(e)$.*

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 $lsc(e^*) \prec lcc(e^*)$ and, if there exists such e^*, then for each edge $e \neq e^*$ of skeleton(μ) we have $lcc(e) \preceq lsc(e^*)$.*

The following theorem completes the characterization for general clustered graphs.

Theorem 3 *[3] 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: (i) the parent cut-vertex of μ (if any) is on the external face of Γ_μ and (ii) each child cut-vertex ρ of μ (if any) is incident to a face f with $lcc(f) \preceq ac(pertinent(\rho))$.*

3 Encoding the Cluster Hierarchy

In this section we show how the c-planarity characterizations mentioned in the previous section can be modified in such a way to produce conditions that are easy to check in linear time. Observe that the characterizations provided by Theorems 1, 2 and 3 only require to test if a cluster is an ancestor or proper ancestor of another cluster. In practice, we only need to perform comparisons between clusters 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 $\psi(\mu) > \psi(\nu)$, where 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 $d(e)$, the value of $d(lcc(e))$. 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 3, 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.

4 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. More formally, 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 non-c-planar triconnected component of G. The algorithm consists of two phases that we sketch below and fully describe 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 triconnected component of G, such that the pertinent of its children are c-planar but $pertinent(\mu)$ 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 $\mathcal{T}$.

4.1 The Preprocessing Phase

The **SPQR-tree Decomposition** step can be performed in linear time [10]. The depth of each edge is computed in constant time with a lowest common ancestor query performed with the data structure in [13].

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 1 *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. □

Since skeleton(μ) is planar and weighted with integer values, a maximum spanning tree can be constructed in linear time (see for example [1, 12]) with respect to the size of skeleton(μ). Hence, because of Lemma 1 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 bucket 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 bucket sort of E_P with respect to $hsd(e)$. Third, we perform a second bucket 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.

4.2 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'. Suppose $skeleton(\mu)$ 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 (counterclockwise) 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 $pertinent(\mu)$ 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.

4.2.1 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 $lsd(e') = \min_i lsd(e_i)$.

4.2.2 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 2 *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 4 *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 **??** 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 **??**, 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 2 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 2 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 **??**, 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 4 has been presented in [4]. We have the following lemma.

Lemma 3 *Let G be an embedded planar graph, let D be its dual with edges weighted with the depth 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 3 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^* [1, 12]. 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 $pertinent(e')$.

4.2.3 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 considering the graph of the incompatibilities between edges and checking whether this graph is bipartite. 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. The fact that the incompatibility graph is bipartite ensure the existence of 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 4 *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 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 4 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 1 *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 [3], 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 2, 3, and 4. From the above discussion we can state the following theorem.

Theorem 5 *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.*

5 Testing and Embedding Algorithm: General Case

In this section we extend the algorithm presented in Section 4 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 when it chooses a certain embedding of the cutvertices.

Lemma 5 *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).*

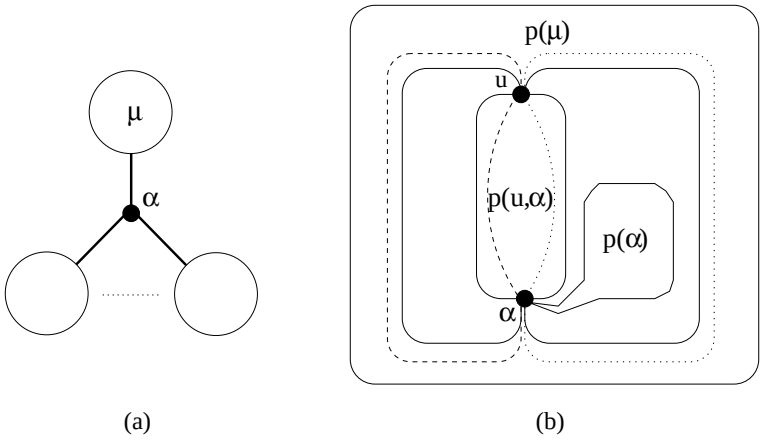

Figure 1: (a) A portion of the BC-tree for the proof of Lemma 5. (b) The relationships between three subgraphs *pertinent*(μ), *pertinent*(u,α), and *pertinent*(α), denoted $p(\mu)$, $p(u,\alpha)$ and $p(\alpha)$, respectively.

Proof. Suppose that there is no c-planar embedding of G unless *pertinent*(α) is inside *pertinent*(u,α). This implies that in any drawing of C with *pertinent*(α) embedded outside *pertinent*(u,α) 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,α); (ii) there are two cycles c_1 and c_2 of depth greater than $d(pertinent(\alpha))$ passing through *pertinent*(u,α) and enclosing the two faces outside *pertinent*(u,α) (see the dotted and dashed cycles of Fig. 1). In case (i), since c encloses both the faces outside *pertinent*(u,α), there can not be a c-planar embedding with *pertinent*(α) inside *pertinent*(u,α). In case (ii), from Fig. 1 it is apparent that the parts of the two cycles c_1 and c_2 outside *pertinent*(u,α) form a cycle enclosing *pertinent*(u,α). Hence, there can not be a c-planar embedding with *pertinent*(α) inside *pertinent*(u,α). *pertinent*(u,α). □

Lemma 6 *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(μ_2) appears in an internal face of the embedding of pertinent(μ_1). There exists a c-planar embedding of C such that pertinent(μ_2) appears in the external face of the embedding of pertinent(μ_1).*

Proof. Suppose that there is no c-planar embedding of G unless *pertinent*(μ_2) is not placed inside a face of *pertinent*(μ_1). This implies that in any drawing of C with *pertinent*(μ_2) embedded outside *pertinent*(μ_1) there is a cycle c of depth $d(c) > d(pertinent(\mu_2))$ enclosing *pertinent*(μ_2). Since c necessarily encloses μ_1 and μ_2, there can not be a c-planar embedding of C such that *pertinent*(μ_2) is placed inside a face of *pertinent*(μ_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 [3] the external face can be changed so that the parent cutvertex is incident to the external face and hence the condition of Theorem 3, modified as in Section 3, 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 4).

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 5 and 6.

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 3, modified as in Section 3, 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 5 and 6. 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 4 **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 5. Due to the above description the following theorem holds.

Theorem 6 *The c-planarity of a c-connected clustered graph can be tested, and possibly a c-planar embedding can be built, in linear time.*

References

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

[2] P. F. Cortese and G. Di Battista. Clustered planarity. In *SCG '05*, pages 32–34, 2005.

[3] P. F. Cortese, G. Di Battista, F. Frati, M. Patrignani, and M. Pizzonia. C-planarity of c-connected clustered graphs: Part I – Characterization. Tech. Report RT-DIA-109-2006, Dip. Informatica e Automazione, Univ. Roma Tre, Jun 2006. `http://web.dia.uniroma3.it/ricerca/rapporti/rapporti.php`.

[4] E. Dahlhaus. Linear time algorithm to recognize clustered planar graphs and its parallelization. In C.L. Lucchesi, editor, *LATIN ́98*, volume 1380 of *LNCS*, 1998.

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

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

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

[8] Q. W. Feng, R. F. Cohen, and P. Eades. How to draw a planar clustered graph. In Ding-Zhu Du and Ming Li, editors, *Proc. COCOON'95*, volume 959 of *LNCS*, pages 21–30. Springer-Verlag, 1995.

[9] Q. W. Feng, R. F. Cohen, and P. Eades. Planarity for clustered graphs. In *Proc. ESA '95*, volume 979 of *LNCS*, pages 213–226. Springer-Verlag, 1995.

[10] C. Gutwenger and P. Mutzel. A linear time implementation of spqr-trees. In Joe Marks, editor, *Graph Drawing, 2000*, pages 77–90. Springer, 2001.

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

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

[13] Baruch Schieber and Uzi 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 triconnected component causing non-c-planarity 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 μ in $\mathcal{T}$ in post-order traversal **do**
let e' be the virtual edge representing μ in the skeleton of its parent node.
$hsd(e') = \min_{e \in skeleton(\mu)} hsd(e)$
if μ is an S node **then**
$d(e') = \min_{e \in skeleton(\mu)} d(e)$
else if μ is a P node **then**
$d(e') = \max_{e \in skeleton(\mu)} d(e)$
else if μ is an R node **then**
Compute a Maximum Spanning Tree MST of skeleton(μ)
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 unique bucket sort.
Embedding Construction Phase
for all node μ in $\mathcal{T}$ in post-order traversal **do**
if μ 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 μ is an P node **then**
if ProcessPNode(μ,e')=False **then**
return μ
end if
else if μ is an R node **then**
if ProcessRNode(μ,e')=False **then**
return μ
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 2: The c-planarity testing and embedding algorithm for c-connected clustered graphs whose underlying graph is biconnected.

Procedure ProcessPNode(μ,e')

{The edges of skeleton(μ) are already ordered in a list $I(\mu)$}
let e_1 be the first element of $I(\mu)$
if skeleton(μ) contains three pairwise incompatible edges **then**
  return False
end if
for all $e \neq e_1$ in skeleton(μ) **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(μ) **do**
  e_l=last element in $\overline{I_L}$, e_r=last element in I_R
  if e is incompatible with e_l **then**
    append e to I_R
  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(μ) 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 3: Testing and embedding procedure for P nodes.

Procedure ProcessRNode(μ,e')

construct the graph G^* from skeleton(μ)
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(μ) **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(μ) connecting u to v clockwise
let b_l the path on the external face of skeleton(μ) 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 4: 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 ρ in $\mathcal{B}$ in post-order traversal **do**
  compute the depth of $pertinent(\rho)$
end for
for all node μ 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 ρ_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 unique bucket sort
Block Embedding Phase
for all node μ in $\mathcal{B}$ **do**
  for all node $\sigma \in \mathcal{T}_\mu$ in post-order traversal **do**
    let ρ_i be the cutvertices in $skeleton(\sigma)$ different from the poles
    if σ is an S node **then**
      process σ as in the biconnected case
      embed the blocks connected to ρ_i in the highest side of $skeleton(\sigma)$
    else if σ is an P node **then**
      process σ as in the biconnected case
    else if σ is an R node **then**
      test the condition on $skeleton(\sigma)$ as in the biconnected case
      if each ρ_i is not incident to a face f with $d(f) \leq d(pertinent(\rho_i))$ **then**
        return False
      else
        embed the blocks of ρ_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 Γ_μ of μ as in the biconnected case
  let f be a face with minimum depth incident to the root cutvertex of μ
  choose f as external face for Γ_μ
end for
merge the embedding of the blocks

Figure 5: The c-planarity testing and embedding algorithm for c-connected clustered graphs