C-Planarity of Extrovert Clustered Graphs*

Michael T. Goodrich, George S. Lueker, and Jonathan Z. Sun

Department of Computer Science,
Donald Bren School of Information and Computer Sciences,
University of California, Irvine, CA 92697-3435, USA
{goodrich, lueker, zhengsun}(at)ics.uci.edu

Abstract. A clustered graph has its vertices grouped into clusters in a hierarchical way via subset inclusion, thereby imposing a tree structure on the clustering relationship. The c-planarity problem is to determine if such a graph can be drawn in a planar way, with clusters drawn as nested regions and with each edge (drawn as a curve between vertex points) crossing the boundary of each region at most once. Unfortunately, as with the graph isomorphism problem, it is open as to whether the c-planarity problem is NP-complete or in P. In this paper, we show how to solve the c-planarity problem in polynomial time for a new class of clustered graphs, which we call *extrovert* clustered graphs. This class is quite natural (we argue that it captures many clustering relationships that are likely to arise in practice) and includes the clustered graphs tested in previous work by Dahlhaus, as well as Feng, Eades, and Cohen. Interestingly, this class of graphs does not include, nor is it included by, a class studied recently by Gutwenger *et al.*; therefore, this paper offers an alternative advancement in our understanding of the efficient drawability of clustered graphs in a planar way. Our testing algorithm runs in $O(n^3)$ time and implies an embedding algorithm with the same time complexity.

1 Introduction

A *clustered graph* (or *c-graph*) consists of a pair $C = (G, \tau)$, where $G = (V, E)$ is an undirected graph having vertex set V and edge set E, and τ is a rooted tree defining a hierarchy of vertex clusters, which are subsets of V organized hierarchically by subset inclusion. That is, each node of τ represents a cluster that is a subset of V (with the root of τ representing V), and the ancestor-descendant relation of two nodes corresponds to the inclusion relation of two clusters. Any two clusters in this hierarchy are either disjoint or one is completely included in the other. We refer to G and τ as being the *underlying graph* and the *inclusion tree* of C, respectively. Throughout this paper, we reserve the Greek letters ν and μ for clusters and the Roman letters x and y for vertices.

Clustered graphs arise naturally from any context where a hierarchy is imposed on a set of interrelated objects. Naturally, we would like to visualize the hierarchical

* This is an extended abstract. Work by the first and the third authors is supported by NSF Grants CCR-0225642 and CCR-0312760.

P. Healy and N.S. Nikolov (Eds.): GD 2005, LNCS 3843, pp. 211–222, 2005.

information and relationships that are represented in a clustered graph, and a way of doing so with minimal confusion is to draw the clustered graph in a planar way. Deciding if such a drawing is possible is one of the most interesting problems involving clustered graphs, and was posed by Feng, Eades and Cohen [12]. Formally, this problem, which is called the *c-planarity problem*, asks if C can be drawn (or embedded) in the plane satisfying the following criteria:

1. There is no crossing between any edges of the underlying graph G.
2. Each cluster $\nu \in \tau$ can be enclosed in one simple closed region by a closed curve $b(\nu)$ which is called the *boundary curve* of ν.
3. There is no crossing between boundary curves of any two clusters.
4. There is exactly one crossing between an edge (x, y) and a boundary curve $b(\nu)$ if $x \in \nu$ and $y \notin \nu$. Otherwise there is no crossing between an edge and a boundary curve.

Such a drawing is called a *c-planar drawing*, and a clustered graph is *c-planar* iff it admits such a drawing. Clustered graphs and c-planar drawings bear both significance in theory and interest in practice. For example, if we visualize the communication network of a company such that the vertices, clusters, and edges represent respectively workstations, departments, and communications between two workstations, clearly we want a simple region and single boundary curve for each department and no crossings except those in the above Criterion 4, so that, for any department, we can identify (e.g., for monitoring, blocking, or firewalling) that department's external communications just by looking at the boundary curve of the corresponding cluster. Another example application is in VLSI, where in addition to designing a planar circuit, we might want to piece each functional module together in a hierarchical way.

While the problem of determining if a given graph is planar is well-known to be solvable in linear time (e.g., see [3, 14, 17]), the general c-planarity problem

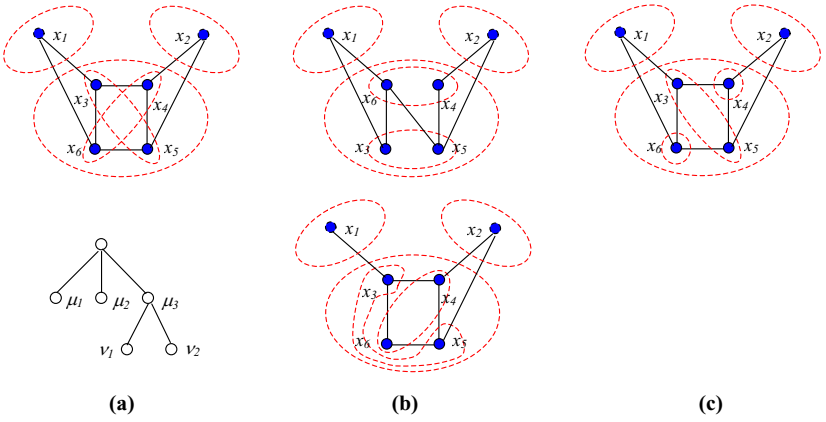

Fig. 1. (a) A c-graph C with 5 clusters $\mu_1 = \{x_1\}$, $\mu_2 = \{x_2\}$, $\mu_3 = \{x_3, x_4, x_5, x_6\}$, $\nu_1 = \{x_3, x_5\}$ and $\nu_2 = \{x_4, x_6\}$, and the inclusion tree τ. C is not c-planar although the underlying graph is planar. (b) Removing any edge of the underlying graph will make C c-planar. (c) Splitting cluster ν_1 or ν_2 will also make C c-planar.

is not known to be solvable in polynomial time. In particular, the existence of boundary curves makes the c-planar testing and embedding significantly harder than simply testing a graph for planarity. In Fig. 1, for example, the c-graph in (a) is not c-planar although the underlying graph is obviously planar, but removing one edge or splitting one cluster in this c-graph will make it c-planar, as illustrated in (b) and (c).

Although a number of papers have addressed the problem of how to draw a c-planar c-graph in the plane [1, 6, 7, 8, 9, 10], very little progress has been made in testing the c-planarity of a given c-graph. Previous work provides effective tests only for a few special classes of c-graphs [4, 5, 12, 13]. In this paper we define and test a new class of c-graphs, which generalizes the result in [5, 12] but is not comparable with [4, 13]. The general problem is still open. So far the testing problem and the embedding problem appear to be equivalent in all solved cases (since each testing algorithm implies an embedding algorithm), so we don't distinguish them unless necessary.

1.1 Previous Results

Let $G(\nu)$ be the subgraph of G induced by cluster ν. (ν is a node in τ and an associated set of vertices in V.) Then ν is a *connected cluster* iff $G(\nu)$ is a connected subgraph. Otherwise $\nu = (\nu^1, \ldots, \nu^k)$ is a *disconnected cluster*, where each $G(\nu^i)$ is a connected component of $G(\nu)$.[1] That is, each ν^i is a set of vertices in a connected component of $G(\nu)$. For simplicity, we call ν^i a *chunk* of ν. A connected cluster is considered to have itself as the only chunk, i.e., $\nu = (\nu^1)$ when ν is connected. C is *c-connected* iff all clusters in τ are connected.[2] The c-planarity problem for c-connected c-graphs was solved in $O(n^2)$ time by Feng, Eades and Cohen [12], and then in linear time by Dauhlhaus [5]. For general c-graphs, it is unknown if the problem is NP-hard or not. Gutwenger *et al.* [13] solved in $O(n^2)$ time the case of *almost c-connected* c-graphs, namely, those c-graphs in which either each disconnected cluster $\nu \in \tau$ has its parent and all siblings connected, or all disconnected clusters lie on a path in τ. Cortese *et al.* [4] recently solved in polynomial time another special case, which we call the *cycles of clusters*, where the underlying graph is a cycle and the clusters at each level of the inclusion tree, when contracted into vertices, also form a cycle. To the best of our knowledge, these three classes of c-graphs are the only ones for which c-planarity has been tested in polynomial time.

1.2 Extrovert C-Graphs

We introduce the concept of classifying the disconnected clusters into *extrovert* and *introvert*, and will later solve the c-planarity problem for the case that all disconnected clusters are extrovert.

[1] In this paper, we will always use superscripts to denote the partition of an object, and subscripts to distinguish different objects.

[2] The previous papers simply used the term *connected* instead of *c-connected*, but we consider it desirable to introduce the new terminology to distinguish between a connected graph G and a c-connected c-graph C.

An edge $(x, y) \in E(G)$ is called an *extrovert edge* of a cluster ν iff $x \in \nu$ and $y \notin \nu$.[3] We call x an *extrovert vertex* of ν in this case. We denote by E^*_ν and V^*_ν respectively the sets of extrovert edges and extrovert vertices of ν. For a subset $\nu^0 \subset \nu$, we denote by $E^*_\nu(\nu^0)$ and $V^*_\nu(\nu^0)$ respectively the corresponding subsets of E^*_ν and V^*_ν, i.e., $E^*_\nu(\nu^0) = \{e \in E^*_\nu : e \text{ is incident on a vertex of } \nu^0\}$ and $V^*_\nu(\nu^0) = V^*_\nu \cap \nu^0$.

Definition 1. (extrovert chunks, clusters, and c-graphs)

- *A chunk ν^i of a disconnected cluster $\nu = (\nu^1, \ldots, \nu^k)$ is an extrovert chunk iff the parent cluster μ of ν is connected, and $E^*_\mu(\nu^i) \neq \emptyset$.*
- *A disconnected cluster $\nu = (\nu^1, \ldots, \nu^k)$ is an extrovert cluster iff each chunk ν^i, $i \in \{1, \ldots, k\}$, is extrovert.*
- *$C = (G, \tau)$ is an extrovert c-graph iff all clusters in τ are either connected or extrovert.*

Otherwise the corresponding chunks, clusters and c-graphs are introvert. (See Fig. 1 (a) for example of an extrovert c-graph with extrovert clusters ν_1 and ν_2.)

Like the almost-connected c-graphs of [13], extrovert c-graphs include the class of c-connected c-graphs. Extrovert c-graphs appear to allow a greater degree of disconnectivity than almost-connected c-graphs, since many sibling clusters are allowed to be disconnected. Extrovert c-graphs are also more flexible than the cycles of clusters of [4].

Extrovert c-graphs are a significant generalization of c-connected c-graphs, and we hope they will find use in practice. Intuitively, why might several chunks of a cluster need be drawn together (in the same cluster) when they have no relationships (edges) between them? Perhaps it is because they have similar relationships to entities outside of the cluster. Thus, since our definition requires each chunk of a disconnected cluster ν to have at least one edge going out of the parent cluster of ν, we might expect that this sort of situation arises in practice.

2 Preliminaries

2.1 PQ-Tree and PQ-Reduction

A PQ-tree [3] $T(U)$ is a tree on a set U of n leaves that has two types of internal nodes, P-nodes and Q-nodes, where a P-node can permute its children arbitrarily but a Q-node can only reverse the order of its children. Various combinations of permuting the children of the P-nodes and reversing the order of children of some the Q-nodes result in various permutations of U at the tree leaves. The set of all achievable permutations of the leaves is called the *consistent set* of $T(U)$ and is denoted by CONSISTENT($T(U)$). We say that a subset $S \subseteq U$ of leaves in $T(U)$ is *consecutive* in a permutation π of U if the elements in S appear as a consecutive subsequence in π. PQ-trees support a reduction operation

$$\text{PQ-REDUCE}(T(U), S) \tag{1}$$

[3] An *extrovert edge* is called a *virtual edge* in [12].

that returns a new PQ-tree whose consistent set contains exactly those elements of CONSISTENT($T(U)$) in which S is consecutive; if there are no such elements the operation fails.

2.2 Circular Permutations

Suppose we wish to read off the order in which a set of elements appear on the circumference of a circle. Depending on where we start, and whether we read clockwise or counterclockwise, we can obtain various permutations; we will say these permutations are *circularly equivalent*. For example, the permutations $(3, 5, 2, 4, 1, 6)$, $(2, 4, 1, 6, 3, 5)$, and $(5, 3, 6, 1, 4, 2)$ are circularly equivalent. We call an equivalence class of this relation a *circular permutation*. We say any element of the equivalence class is a *representative* of the circular permutation. Informally, a circular permutation represents the order of objects that appear around a circle.

Say a set S is *consecutive* in a circular permutation π if it is consecutive in any representative of π. Informally, this means that the elements of S appear consecutively around the circle.

2.3 PC-Trees and PC-Reduction

PC-trees [15] provide an elegant structure that both simplifies PQ-trees and allows convenient operations on circular permutations. A PC-tree is an unrooted tree with two types of internal nodes, P-nodes and C-nodes, where a P-node can permute its neighbors and a C-node is assigned a cyclic order to its neighbors and can only reverse the order. The *circular consistent set* of a PC-tree $T(U)$ on a set of leaves U, denoted C-CONSISTENT($T(U)$), is the set of all permissible circular permutations of the leaves. Much as with PQ-trees, PC-trees support an operation

$$\text{PC-REDUCE}(T(U), S) \tag{2}$$

that returns a new PC-tree whose circular consistent set contains exactly those circular permutations in C-CONSISTENT($T(U)$) for which the subset S of leaves in $T(U)$ is consecutive; again, if there are no such elements, the operation fails. These trees will be very useful in our algorithm.

It's clear that a PQ-tree is a rooted image of a PC-tree where the Q-nodes correspond to the C-nodes. Therefore the concept of circular consistent set also applies to PQ-trees and the operation PC-REDUCE can take a PQ-tree as input as well. We will not distinguish PQ-trees and PC-trees any more, but use PQ-REDUCE and PC-REDUCE as two operations that can act on the same tree.

2.4 C-Planarity of C-Connected C-Graphs

Let $\mathcal{D}$ be a planar embedding of G. Then for a subgraph H of G we use $\mathcal{D}(H)$ to denote the subembedding of H in $\mathcal{D}$. The boundary of a face in a planar embedding consists of the vertices and edges incident with this face. When ν is a connected cluster, criteria 2–4 in the definition of a c-planar embedding are actually equivalent to requiring that all extrovert vertices of ν are at theboundary

of the outer face of $\mathcal{D}(G(\nu))$, and all extrovert edges are in the outer face of $\mathcal{D}(G(\nu))$. The boundary curve $b(\nu)$ can always be obtained by slightly expanding the boundary of the outer face of $\mathcal{D}(G(\nu))$. (See cluster μ_3 in Fig. 1 (a).) In the figures, we use solid lines for the boundary of the outer face of $\mathcal{D}(G(\nu))$ and dashed lines for the boundary curve $b(\nu)$. The definition of c-planarity restricted to c-connected c-graphs then translates into the following property for each cluster.

Property 1 (simple). For a connected cluster ν, a *simple planar embedding of* ν is a planar embedding $\mathcal{D}$ of the graph $(G(\nu) \cup E^*_\nu)$ with the vertices of V^*_ν drawn at the boundary of the outer face of the subembedding $\mathcal{D}(G(\nu))$ and the edges of E^*_ν drawn in the outer face of $\mathcal{D}(G(\nu))$. For a planar embedding $\mathcal{D}$ of G, we say ν is *simple in* $\mathcal{D}$ if the subembedding $\mathcal{D}(G(\nu) \cup E^*_\nu)$ is a simple planar embedding of ν. (In both cases the boundary curve $b(\nu)$ is a slight expansion of the boundary of the outer face of $\mathcal{D}(G(\nu))$.)

The following three lemmas can be deduced from the results of [12]. We summarize and restate them in a particular way to facilitate the presentation of our work. The first lemma is equivalent to Theorem 1 in [12].

Lemma 1. *A c-connected c-graph $C = (G, \tau)$ is c-planar iff G is planar and there exists a planar embedding $\mathcal{D}$ of G in which each $\nu \in \tau$ is simple.*

The next two lemmas are deduced from the testing algorithm of [12]. We provide them without proofs. We also omit the original constructions of [12] that fulfill these procedures.

Lemma 2. *For any connected cluster ν of size m, we can build in $O(m)$ time a PQ-tree on the set of leaves E^*_ν, say $T(E^*_\nu)$, such that the circular consistent set of $(T(E^*_\nu))$ equals the set of circular permutations of the edges of E^*_ν on $b(\nu)$ resulting from all possible simple planar embeddings of ν.*

We write the procedure of building $T(E^*_\nu)$ from ν in [12] as the following operation that converts a subgraph to a PQ-tree.

$$T(E^*_\nu) \leftarrow \text{CONVERT}(\nu). \tag{3}$$

Lemma 3. *For any PQ-tree $T(E^*_\nu)$ resulting from Lemma 2, we can build in $O(m)$ time a representative subgraph R_ν as a replacement of $G(\nu)$ in G with the vertex set r_ν of R_ν being a replacement of the cluster ν, and the extrovert vertex and extrovert edge sets of r_ν remaining V^*_ν and E^*_ν. If we substitute R_ν for $G(\nu)$ in G, then*

- *the circular consistent set of $(T(E^*_\nu))$ equals the set of circular permutations of E^*_ν at $b(r_\nu)$ resulting from all possible simple planar embeddings of r_ν.*
- *G is planar iff G has a planar embedding in which r_ν is simple.*

We write the procedure of building the representative subgraph R_ν from $T(E^*_\nu)$ in [12] as

$$R_\nu \leftarrow \text{REPRESENT}(T(E^*_\nu)), \tag{4}$$

and the procedure of substituting R_ν for $G(\nu)$ in G as

$$G \leftarrow \text{SUBSTITUTE}(G(\nu), R_\nu). \tag{5}$$

The above lemmas characterize the c-planarity of c-connected c-graphs, and provide gadgets to test it. Intuitively, the processes in Lemma 2 and 3 provide that, G has a planar embedding with ν being simple $\Leftrightarrow$ G after the substitution has a planar embedding with r_ν being simple $\Leftrightarrow$ G after the substitution is planar. Then the testing algorithm CPT in [12] traverses τ bottom-up and performs operations (3),(4),(5) for each cluster. After substituting all clusters, G is planar if and only if the original G has a planar embedding that makes all original clusters simple, so that c-planarity testing is converted into planarity testing. The algorithm runs in $O(n^2)$ time.

3 C-Planarity of Extrovert C-Graphs

In this section we characterize the c-planarity of extrovert c-graphs. The following lemma is a straightforward characterization of the c-planarity of c-graphs.

Lemma 4 (Theorem 2 in [12]). *A c-graph $C = (G = (V, E), \tau)$ is c-planar, iff there exists a c-connected c-planar c-graph $C' = (G' = (V, E'), \tau)$ with $E \subset E'$.*

C' is called a *super c-graph* of C. Our idea to characterize the c-planarity of extrovert c-graphs is to treat each chunk of a cluster as a small connected cluster and use the following two properties together with Property 1.

Property 2 (connectable). Let $\nu = (\nu^1, \ldots, \nu^k)$ be a disconnected cluster with its parent cluster μ being connected, and $\mathcal{D}$ be a simple planar embedding of μ in which each chunk ν^i of ν is also simple. We say that ν is connectable in $\mathcal{D}$ iff there is a way to draw $k-1$ extra edges inside $b(\mu)$ that connect the k chunks of ν into one connected component, without introducing any edge crossings. We call the extra edges *bridges* of ν.

Property 3 (conflict). Let $\nu_l = (\nu_l^1, \ldots, \nu_l^{k_l})$, $l = 1, \ldots$, be sibling disconnected clusters with their parent cluster μ being connected, and $\mathcal{D}$ be a simple planar embedding of μ in which each chunk ν_l^i of each ν_l is also simple. We say that the ν_l's conflict in $\mathcal{D}$ iff each ν_l is connectable, but there is no way to connect all of the ν_l's inside $b(\mu)$ simultaneously without introducing edge crossings.

Theorem 1. *An extrovert c-graph $C = (G, \tau)$ is c-planar iff G is planar and there exists a planar embedding $\mathcal{D}$ of G such that, each chunk of each cluster is simple in $\mathcal{D}$; each extrovert cluster is connectable in the subembedding of its parent cluster; and no sibling extrovert clusters conflict.*

Proof. [sketch]*Sufficiency.* Assume there is such an embedding $\mathcal{D}$ of G. Since each chunk of cluster is simple, we can add a boundary curve for each chunk in $\mathcal{D}$, which is slightly outside the boundary of the outer face of this chunk. The only thing disqualifying this drawing to be a c-planar drawing is that an extrovert cluster is enclosed in not a single but many regions. Connect each

extrovert cluster ν with $k > 1$ chunks by adding $k-1$ bridges. As required by Property 2 and 3, the bridge between two chunks ν^i and ν^j will cross only with the boundary curves of ν^i and ν^j. So we can merge the k simple closed regions for chunks into one region by "digging tunnels" along the bridges as shown in Fig 2. Since each chunk region is simple and there are only $k-1$ bridges spanning k chunks, the resulting region for the whole cluster is still simple. Doing this for all extrovert clusters gives a c-planar embedding of C.

Fig. 2. Merge two chunk regions and boundary curves by digging a tunnel along a bridge

Necessity. Assume C is c-planar. By Lemma 4, there exists a c-connected super c-graph $C' = (G', \tau)$ of C and it has a c-planar embedding $\mathcal{D}$. We only need to show that in $\mathcal{D}$ of G', each chunk of a cluster is simple; each extrovert cluster is connectable in the subembedding of its parent; and no sibling extrovert clusters conflict. Consider an extrovert cluster $\nu = \{\nu^1, \nu^2, \ldots\}$ with parent cluster μ. Since each chunk ν^i is extrovert, there is an extrovert edge in $E^*_\mu(\nu^i)$ crossing $b(\mu)$. Therefore any chunk ν^i cannot be enclosed in an inner face of another chunk ν^j, so each chunk must be simple. The properties of being connectable and not conflicting are obvious, noting that the extra edges in C' include all the bridges. □

4 Testing Algorithm

We first convert the inclusion tree τ into τ' by splitting each disconnected cluster into its chunks. Each node $\nu \in \tau$ is a cluster and each node $\nu^i \in \tau'$ is a chunk. (See Fig. 3.) We always use μ for a parent and ν a for child. The frame of our testing algorithm EXTROVERT-CPT is shown in Fig. 4. It inherits the algorithm CPT in [12], except that we process the chunks in τ' instead of the clusters in τ, and insert the following subroutine to filter the permissible circular permutations of extrovert edges at $b(\mu^i)$. (See Fig. 5.)

$$T'(E^*_{\mu^i}) \leftarrow \text{FILTER}(T(E^*_{\mu^i})). \tag{6}$$

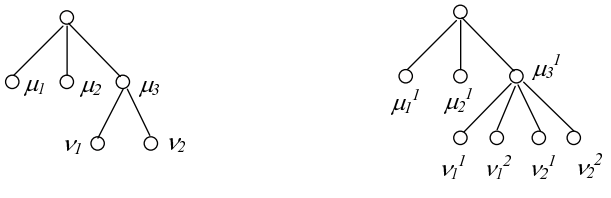

Fig. 3. τ and τ' for the c-graph in Fig. 1 (a)

Algorithm EXTROVERT-CPT

1: **for** each $\mu^i \in \tau'$ in postorder **do**
2: test planarity of $G(\mu^i)$
3: **if** μ^i is not the root of τ' **then**
4: $T(E^*_{\mu^i}) \leftarrow$ CONVERT$(G(\mu^i))$
5: $T'(E^*_{\mu^i}) \leftarrow$ FILTER$(T(E^*_{\mu^i}))$
6: $R_{\mu^i} \leftarrow$ REPRESENT$(T'(E^*_{\mu^i}))$
7: $G \leftarrow$ SUBSTITUTE$(G(\mu^i), R_{\mu^i})$

Fig. 4. The frame algorithm for testing c-planarity of an extrovert c-graph. If any subroutine at any moment fails, (either a subgraph is not planar or a reduction is not doable,) then the algorithm stops and returns "not c-planar". Otherwise it returns "c-planar" after passing the planarity test of $G(root(\tau'))$ in Step 2.

Algorithm FILTER$(T(E^*_{\mu^i}))$

for each τ-child ν of μ^i **do**
contract $S(\nu)$ into a vertex in S
for each ν that is an extrovert τ-child cluster of μ^i **do**
for each set of vertices $G(\mu^i\backslash\nu)^j$ in $G(\mu^i\backslash\nu)$ that contracts into a connected component $C^j_{S(\mu^i\backslash\nu)}$ in $S(\mu^i\backslash\nu)$ **do**
$T(E^*_{\mu^i}) \leftarrow$PC-REDUCE$(T(E^*_{\mu^i}), E^*_{\mu^i}(G(\mu^i\backslash\nu)^j))$
return $T(E^*_{\mu^i})$

Fig. 5. The filter algorithm

Now we describe the filter algorithm FILTER$(T(E^*_{\mu^i}))$ shown in Fig. 5. We call a node $\nu \in \tau$ a τ-child of a node $\mu^i \in \tau'$, and μ^i the τ'-parent of ν, if every chunk ν^i of ν is a child of μ^i in τ'. Since the parent cluster of an extrovert cluster is connected, each $\nu \in \tau$ has exactly one τ'-parent. In addition to G, we maintain a *skeleton* S of G which is initially equal to G but, at the time any μ^i is processed in FILTER$(T(E^*_{\mu^i}))$, contracts every τ-child ν of μ^i into a vertex. We denote by C^j_F the j-th connected component of a disconnected graph F. Let $S(\mu^i\backslash\nu)$, the subgraph resulting from removing ν from $S(\mu^i)$, have connected components $C^1_{S(\mu^i\backslash\nu)}, C^2_{S(\mu^i\backslash\nu)}, \ldots$. Let $G(\mu^i\backslash\nu)^j$ be the part of $G(\mu^i\backslash\nu)$ that contracts into $C^j_{S(\mu^i\backslash\nu)}$ in S (as every τ-child of μ^i is contracted into a vertex). Then we require that in the output of FILTER$(T(E^*_{\mu^i}))$ the extrovert edges of μ^i in each $G(\mu^i\backslash\nu)^j$ are always consecutive among all extrovert edges of μ^i on $b(\mu^i)$. This will be achieved by doing a PC-reduction for each $G(\mu^i\backslash\nu)^j$ as Fig. 5 shows. (Note that $G(\mu^i\backslash\nu)^j$ may not be a connected component of $G(\mu^i\backslash\nu)$, but consist of multiple connected components.)

Recall that by Theorem 1, to qualify a planar embedding of G to be a c-planar embedding of C, we only need to maintain the property of simple for each chunk of each cluster, the property of connectable for each extrovert cluster, and the property of no conflict among all extrovert child clusters in each connected parent cluster. By inheriting the algorithm CPT in [12] but using τ' instead

of τ, EXTROVERT-CPT maintains all the diversity of embedding each chunk of cluster to be simple. In addition, when μ^i is the only chunk of a connected cluster μ, FILTER($T(E^*_{\mu^i})$) will further filter the simple planar embeddings of μ, by doing a sequence of PC-reductions for each extrovert child cluster of μ, so that only those in which all extrovert child clusters of μ are connectable and don't conflict are left in the circular consistent set of $T'(E^*_{\mu^i})$. (If μ^i is a chunk of an extrovert cluster, then by the definition of extrovert c-graph all of its τ-children are connected clusters and FILTER($T(E^*_{\mu^i})$) does nothing.) We'll prove in the next section that FILTER($T(E^*_{\mu^i})$) fulfills this purpose by performing a PC-reduction for the extrovert edges of μ^i incident with each $G(\mu^i \backslash \nu)^j$.

5 Proof of Correctness

In this section we show why making some certain sets of extrovert edges consecutive among all extrovert edges at $b(\mu)$ can provide Property 2 and 3 in Sec. 3 to the planar embedding inside $b(\mu)$. In order to prove the main Theorem 2, we first prove the following lemma.

Lemma 5. *Let $\nu = (\nu^1, \ldots, \nu^k)$ be an extrovert cluster with parent cluster μ, and $G(\mu \backslash \nu)$ have connected components $C^1_{G(\mu\backslash\nu)}, C^2_{G(\mu\backslash\nu)}, \ldots$. Let $\mathcal{D}$ be a simple planar embedding of μ in which each ν^i is also simple, and $\pi(E^*_\mu)$ be the circular permutation of E^*_μ at $b(\mu)$. Then ν is connectable in $\mathcal{D}$, iff for each $C^j_{G(\mu\backslash\nu)}$, $j = 1, 2, \ldots$, $E^*_\mu(C^j_{G(\mu\backslash\nu)})$ is consecutive in $\pi(E^*_\mu)$.*

Proof. [sketch]*Necessity.* If there is a $C^j_{G(\mu\backslash\nu)}$ such that $E^*_\mu(C^j_{G(\mu\backslash\nu)})$ is not consecutive in $\pi(E^*_\mu)$, then there are $e_i \in E^*_\mu$, $i \in \{1, 2, 3, 4\}$, such that $e_1, e_2 \in E^*_\mu(C^j_{G(\mu\backslash\nu)})$ are separated by $e_3, e_4 \notin E^*_\mu(C^j_{G(\mu\backslash\nu)})$ at $b(\mu)$. Then there is a path $p \in C^j_{G(\mu\backslash\nu)}$ from e_1 to e_2 cutting $b(\mu)$ into two halves with e_3 and e_4 being on different sides. (See Fig. 6.) We show that each side contains some chunk ν^i of ν, so that ν cannot be connected inside $b(\mu)$ without crossing p. See the side of e_3. If e_3 is incident with some ν^i, then ν^i is on this side. Otherwise e_3 is incident with some $C^{j'}_{G(\mu\backslash\nu)}$ with $j' \neq j$, in which case there must also be some ν^i on this side because $C^j_{G(\mu\backslash\nu)}$ and $C^{j'}_{G(\mu\backslash\nu)}$ were in a connected graph $G(\mu)$ but become disconnected in $G(\mu\backslash\nu)$. Similarly the side of e_4 contains another chunk $\nu^{i'}$ of ν.

Sufficiency. Suppose ν is not connectable inside $b(\mu)$. We greedily connect the chunks of ν until getting a maximal set of connected $\cup\nu^i$ which doesn't contain some $\nu^{i'}$. Then there must be some paths in $G(\mu)$ with two ends $e_1, e_2 \in E^*_\mu$ cutting $b(\mu)$ into two halves and $\cup\nu^i$ and $\nu^{i'}$ on different sides. We can show that among all such paths there is a path p with no vertex of p belonging to ν, which means that p is contained in some connected component $C^j_{G(\mu\backslash\nu)}$ and $E^*_\mu(C^j_{G(\mu\backslash\nu)})$ is not consecutive in $\pi(E^*_\mu)$ since e_1 and e_2 are separated at $b(\mu)$ by the extrovert edges coming from $\cup\nu^i$ and those from $\nu^{i'}$. Details are omitted. □

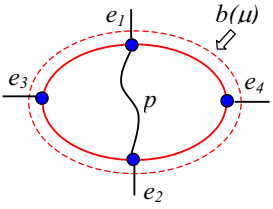

Fig. 6. e_1, e_2 and p are in $C^j_{G(\mu\backslash\nu)}$, and e_3, e_4 are not in $C^j_{G(\mu\backslash\nu)}$. There must be some ν^i on the left of p and another $\nu^{i'}$ on the right.

We conclude with the following two theorems with Theorem 3 showing the correctness and running time of the testing algorithm. Proofs of these theorems are omitted in this extended abstract. An embedding algorithm is implied by the testing algorithm. Details are also omitted.

Theorem 2. *Let $\nu_l = (\nu_l^1, \ldots, \nu_l^{k_l})$, $l = 1, 2, \ldots$, be sibling extrovert clusters with connected parent cluster μ, $\mathcal{D}$ be a simple planar embedding of μ in which each ν_l^i is also simple, and $\pi(E^*_\mu)$ be the circular permutation of E^*_μ at $b(\mu)$. Let S be the skeleton of G in which each child cluster of μ is contracted into a vertex, $S(\mu\backslash\nu_l)$ have connected components $C^1_{S(\mu\backslash\nu_l)}, C^2_{S(\mu\backslash\nu_l)}, \ldots$, and each $C^j_{S(\mu\backslash\nu_l)}$, $j = 1, 2, \ldots$, be contracted from a subgraph $G(\mu\backslash\nu_l)^j$ of $G(\mu\backslash\nu_l)$. Then each ν_l is connectable and all of the ν_l's don't conflict in $\mathcal{D}$, iff for each $G(\mu\backslash\nu_l)^j$, $l = 1, 2, \ldots$ and $j = 1, 2, \ldots$, $E^*_\mu(G(\mu\backslash\nu_l)^j)$ is consecutive in $\pi(E^*_\mu)$.*

Theorem 3. *The algorithm EXTROVERT-CPT correctly tests the c-planarity of an extrovert c-graph in $O(n^3)$ time.*

References

1. G. D. Battista, W. Didimo, and A. Marcandalli. Planarization of clustered graphs. In *Graph Drawing (GD'01), LNCS 2265*, pages 60–74, 2001.
2. G. D. Battista and R. Tamassia. On-line planarity testing. *SIAM J. Comput.*, 25(5):956–997, 1996.
3. K. Booth and G. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *J. Comput. Systems Sci.*, 13(3):335–379, 1976.
4. P. G. Cortese, G. D. Battista, M. Patrignani, and M. Pizzonia. Clustering cycles into cycles of clusters. In *Graph Drawing (GD'04)*, 2004.
5. E. Dahlhaus. A linear time algorithm to recognize clustered planar graphs and its parallelization. In *LATIN'98, LNCS 1380*, pages 239–248, 1998.
6. C. A. Duncan, M. T. Goodrich, and S. G. Kobourov. Planarity-preserving clustering and embedding for large planar graphs. In *Graph Drawing (GD'99), LNCS 1731*, pages 186–196, 1999.
7. P. Eades, Q. Feng, and H. Nagamochi. Drawing clustered graphs on an orthogonal grid. *Journal of Graph Algorithms and Applications*, 3(4):3–29, 1999.

8. P. Eades and Q.-W. Feng. Multilevel visualization of clustered graphs. In *Graph Drawing, GD'96, LNCS 1190*, pages 101–112, 1996.
9. P. Eades, Q.-W. Feng, and X. Lin. Straight-line drawing algorithms for hierarchical graphs and clustered graphs. In *Graph Drawing, GD'96, LNCS 1190*, pages 113–128, 1996.
10. P. Eades and M. L. Huang. Navigating clustered graphs using force-directed methods. *J. Graph Algorithms and Applications: Special Issue on Selected Papers from 1998 Symp. Graph Drawing*, 4(3):157–181, 2000.
11. S. Even and R. E. Tarjan. Computing an st-numbering. *Theoretical Computer Science*, 2(3):339–344, 1976.
12. Q.-W. Feng, P. Eades, and R. F. Cohen. Clustered graphs and C-planarity. In *3rd Annual European Symposium on Algorithms (ESA'95), LNCS 979*, pages 213–226, 1995.
13. C. Gutwenger, M. Jnger, S. Leipert, P. Mutzel, M. Percan, and R. Weiskircher. Advances in C-planarity testing of clustered graphs. In *Graph Drawing (GD'02), LNCS 2528*, pages 220–235, 2002.
14. J. Hopcroft and R. E. Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, 1974.
15. W. Hsu and R.M.McConnell. PC trees and circular-ones arrangements. *Theoretical Computer Science*, 296(1):59–74, 2003.
16. A. Lempel, S. Even, and I. Cederbaum. An algorithm for planarity testing of graphs. In *Theory of graphs: International symposium*, pages 215–232, 1966.
17. T. Nishizeki and N. Chiba. *Planar Graphs: Theory and Algorithms*, volume 32 of *Ann. Discrete Math.* North-Holland, Amsterdam, The Netherlands, 1988.