# Clustered Graphs and C-planarity

Qing-Wen Feng Peter Eades Robert F. Cohen Department of Computer Science The University of Newcastle University Drive - Callaghan NSW 2308, Australia email:{qwfeng,eades,rfc}@cs.newcastle.edu.au

March 1995

(Technical Report 04/95)

#### Abstract

In this paper, we introduce a new graph model known as *clustered graphs*, i.e. graphs with recursive clustering structures. This graph model has many applications in informational and mathematical sciences. In particular, we study C-planarity of clustered graphs. Given a clustered graph, the C-planarity testing problem is to determine whether the clustered graph can be drawn without edge crossings, or edge-region crossings. In this paper, we present efficient algorithms for testing C-planarity and finding C-planar embeddings of clustered graphs.

### 1 Introduction

Representing information visually, or by drawing graphs can greatly improve the effectiveness of user interfaces in many relational information systems [12, 17, 18, 5]. Developing algorithms for drawing graphs automatically and efficiently has become the interest of research for many computer scientists. Research in this area has been very active for the last decade. A recent survey citelabel13new of literature in this area includes over 250 references.

As information systems become more and more complicated, classical graphs tend to be insufficient for modeling the information. This has motivated the development of more powerful graph models, e.g. hypergraphs [2], compound digraphs [21], cigraphs [15] and higraphs [10]. Although these graph models has provided us a high level of abstraction and can be applied to a wide range of applications, automatic layout facilities for these graphs seem hard to develop. Only heuristic algorithms for hierarchical layout of compound digraphs have been developed [21, 20]. In this paper, we introduce a practical and simple model called *clustered graphs*, i.e. graphs with recursive clustering structures (see Fig. 1). This clustering structure can be used to model information in many areas, such as software engineering [23], knowledge representation [13], idea organization [14], software visualization [22] and VLSI design [10].

Planarity is a much studied area for classical graphs. For example, the problem of minimizing edge crossings is proved to be NP-hard [9, 7]. However, efficient algorithms for testing whether a graph is planar (i.e. can be drawn without edge crossings) exist [11, 16, 3, 6]. Planarity issues relating to the more powerful graph models mentioned above have not been studied. In this paper, we introduce *C*-planarity, the planarity of clustered graphs. In a drawing of a clustered graph, vertices and edges are drawn as points and curves as usual. Clusters are drawn as simple closed



Figure 1: An Example of a Clustered Graph



Figure 2: A Non C-planar Clustered Graph with Planar Underlying Graph

curves that define closed regions of the plane. The region for each cluster contains the drawing of the subgraph induced by its vertices and no other vertices. A region for a cluster contains the regions for all its subclusters and does not intersect the region for any other cluster. A clustered graph is *C-planar* if it has a drawing with no crossings between distinct edges, or crossings between an edge and a region. Note that the planarity of the underlying graph does not imply the existence of a *C*-planar drawing of a clustered graph. For example, in Fig. 2, two edges cross a region to which they do not belong.

It appears that C-planarity testing is not a trivial extension of planarity testing of classical graphs. For example, consider the clustered graph in Fig. 3. Suppose that the vertices on three triangles belong to three separate clusters. It is obvious that the graph is planar in the usual sense. The graph induced by the vertices of each cluster is planar; and the graph obtained by collapsing any cluster to a vertex is also planar. However, this clustered graph is not C-planar. Based on analysis of other examples, it appears to us that the st-numbering based planarity testing algorithms in [16] and [3] also cannot be easily adapted to C-planarity testing.

In a clustered graph, the subgraph induced by the vertices of a cluster may not be connected even if the entire underlying graph is connected. This non-connectedness of clusters makes things more complicated. There can be many possible ways to form the regions for clusters even with a fixed embedding of the underlying graph. Fig. 4 gives an example. In Fig. 4(a) and (b), the embeddings of the underlying graph are the same, while the formations of the regions are different. Only the example in (b) gives a C-planar drawing. We concentrate on clustered graphs where the



Figure 3: A Planar but Not C-Planar Clustered Graph



Figure 4: Different Region Formations on the Same Graph

graph induced by each cluster is connected. We call such graphs connected clustered graphs.

In this paper, we develop algorithms for testing C-planarity and finding C-planar embeddings of connected clustered graphs in  $O(n^2)$  time. In Section 2, we present some terminology and some useful characterizations of C-planar clustered graphs. In Section 3, we present an algorithm for testing C-planarity in connected clustered graphs. We also extend the testing algorithm to a C-planar embedding algorithm. We conclude in Section 4 with some interesting open problems.

## 2 Preliminaries

A clustered graph C = (G, T) consists of an undirected graph G and a rooted tree T such that the leaves of T are exactly the vertices of G. Each node  $\nu$  of T represents a cluster  $V(\nu)$  of the vertices of G that are leaves of the subtree rooted at  $\nu$ . Note that the tree T describes an inclusion relation between clusters. The tree T is called the *inclusion tree* of C. The graph G is called the *underlying graph* of C. We let  $T(\nu)$  represent the subtree of T rooted at node  $\nu$  and  $G(\nu)$  denote the subgraph of G induced by the cluster associated with node  $\nu$ . We define  $C(\nu) = (G(\nu), T(\nu))$ to be the *sub-clustered graph* associated with node  $\nu$ . When necessary to avoid confusion, we refer to edges of G as *adjacency edges* and edges of T as *inclusion edges*. For the purposes of this paper, we can assume that each node in T has at least two children except for leaf nodes.

A drawing of a clustered graph C = (G, T) is a representation of the clustered graph in the plane. Each vertex of G is represented by a point. Each edge of G is represented by a simple curve between the drawing of its endpoints. For each node  $\nu$  of T, the cluster  $V(\nu)$  is drawn as a simple closed region R defined by a simple closed curve in the plane such that:

- the regions for all sub-clusters of R are completely contained in the interior of R;
- the regions for all other clusters are completely contained in the exterior of R;
- if there is an edge e between two vertices of V(ν) then the drawing of e is completely contained in R.

Given a drawing  $\mathcal{D}$  of C = (G, T), we produce a *consistent drawing*  $\mathcal{D}'$  of G by removing the region boundary curves from  $\mathcal{D}$ .

We say that the drawing of edge e and region R have an *edge-region crossing* if the drawing of e crosses the boundary of R more than once. A drawing of a clustered graph is *C-planar* if there are no edge crossings or edge-region crossings. If a clustered graph has a C-planar drawing then we say it is *C-planar* (Fig. 1 gives an example of a C-planar clustered graph). Note that C = (G, T) is C-planar only if G is planar. A C-planar drawing also contains a planar drawing of the underlying graph.

An edge is said to be *incident* with a cluster  $V(\nu)$  if one end of the edge is a vertex of  $V(\nu)$  but the other end is not in  $V(\nu)$ . An *embedding* of C = (G, T) includes an embedding of G plus the circular ordering of edges crossing the boundary of the region of each non trivial cluster (a cluster which is not a single vertex). In other words, an embedding of a clustered graph consists of the circular ordering of edges around each cluster which are incident to that cluster.

A clustered graph C = (G, T) is a connected clustered graph if each cluster induces a connected subgraph of G. The following theorem gives a necessary and sufficient condition for the C-planarity of connected clustered graphs.

**Theorem 1** A connected clustered graph C = (G, T) is C-planar if and only if graph G is planar and there exists a planar drawing  $\mathcal{D}$  of G, such that for each node  $\nu$  of T, all the vertices and edges of  $G - G(\nu)$  are in the outer face of the drawing of  $G(\nu)$ .

**Proof:** Note that since each  $G(\nu)$  is connected, the boundary of its outer face in any planar drawing of  $G(\nu)$  consists of a connected cycle.

Consider a clustered graph C = (G, T) with a C-planar drawing  $\mathcal{D}$ , let  $\mathcal{D}'$  be the consistent drawing of the underlying graph G. Suppose that there is a node  $\nu$  of T such that  $G - G(\nu)$  are not all in the outer face of the drawing of  $G(\nu)$  in  $\mathcal{D}'$ . Then there must exist a vertex  $\nu$  in  $G - G(\nu)$ which is drawn in the interior of the outer facial cycle of the drawing of  $G(\nu)$ . Then any simple region that contains the drawing of  $G(\nu)$  must also contain  $\nu$ . This contradicts the assumption that  $\mathcal{D}$  is a C-planar drawing of C.

Now, consider a planar drawing  $\mathcal{D}'$  of the underlying graph G, such that for each node  $\nu$  of T,  $G - G(\nu)$  is drawn in the outer face of the drawing of  $G(\nu)$ . We produce a drawing  $\mathcal{D}$  of clustered graph G = (C, T) by adding cluster boundaries to  $\mathcal{D}'$  recursively up tree T. For each node  $\nu$  of T, we make the boundary for cluster of  $\nu$  by drawing a simple closed curve in the outer face of  $G(\nu)$  along its outer facial cycle,  $\epsilon > 0$  distance away from the outer facial cycle of  $G(\nu)$  or the boundary of the included regions. In this construction, each region is simple, and the region inclusion convention is followed. There are no edge crossings, since D' is a planar drawing of G. By construction, the only edges that cross the boundary of the region for a node  $\nu$  of T are edges connecting vertices of  $G(\nu)$  with vertices of  $G - G(\nu)$ . Consequently, there are no edge-region crossings. Clearly, this construction produces a C-planar drawing of C.

Next, we present a characterization of C-planarity of general clustered graphs. We need some more terminology here. Suppose that  $C_1 = (G_1, T_1)$  and  $C_2 = (G_2, T_2)$  are two clustered graphs,  $T_1$  is a subtree of  $T_2$ , and for each node  $\nu$  of  $T_1$ ,  $G_1(\nu)$  is a subgraph of  $G_2(\nu)$ . Then we say  $C_1$  is a sub-clustered graph of  $C_2$ , and  $C_2$  is a super-clustered graph of  $C_1$ .



Figure 5: The Example for the Proof of Theorem 2

**Theorem 2** A clustered graph C = (G, T) is C-planar if and only if it is a sub-clustered graph of a connected and C-planar clustered graph.

**Proof:** Suppose that clustered graph C = (G, T) is C-planar, where G = (V, E); and  $\mathcal{D}$  is a C-planar drawing of C. Let  $\nu$  be a node of T, and let  $w_1, w_2, \ldots, w_k$  be the points in circular order where edges cross the region boundary of cluster  $\nu$ . Let  $v_i$  and  $v_{i+1}$  be the vertices of  $G(\nu)$ which connect to adjacent points  $w_i$  and  $w_{i+1}$  respectively. In this C-planar drawing  $\mathcal{D}$  of C, vertices  $w_i, w_{i+1}, v_{i+1}$  and  $v_i$  are on boundary of some face f (See Fig. 5(a)), since  $v_i$  and  $v_{i+1}$ are on the same side of the region boundary of cluster  $\nu$ . We add edge  $(v_i, v_{i+1})$  to  $G(\nu)$  if  $v_i$  and  $v_{i+1}$  are not connected previously. We draw it by making a curve from  $v_i$  to  $v_{i+1}$  along the curve  $(v_i, w_i, w_{i+1}, v_{i+1}) \in 0$  distance away from it, inside face f. This does not produce any crossings since the curve we draw is totally inside face f. Suppose that there is a connected component Hof  $G(\nu)$  which has no connection with  $G - G(\nu)$ . In the C-planar drawing  $\mathcal{D}$  of C, let region R be the smallest subregion of  $R(\nu)$  (the region for cluster  $\nu$ ) which contains the drawing of H. Let  $\overline{R}$  be the subregion of R bounded by the the bounding cycle of R and the outer facial cycle of H. Suppose that v is a vertex on the outer facial cycle of H and u is a vertex of  $G(\nu)$  on the boundary of region R. Any straight line through v must contain a segment (x, y) which is contained in R. with x on the outer facial cycle of H and y on the boundary of R (see Fig. 5(b)). We add an edge (v, u) to  $G(\nu)$ . We draw edge (v, u) in  $\overline{R}$  in the following manner. First, we draw a curve from v to x along the outer facial cycle of H. Then we continue along the segment (x, y) to point y. Finally, we follow the boundary of R to u. This does not introduce any crossings since this curve is formed all inside region  $\bar{R}$ . By the operations above, a super-clustered graph C'(G',T) of C is obtained, where G' = (V, E'); a C-planar drawing of C' is formed; and C' is a connected clustered graph.

Suppose that C is a sub-clustered graph of C', where C' is connected and C-planar. A C-planar drawing of C can be obtained by restricting a C-planar drawing of C' to C. Therefore, C is C-planar.

### 3 C-planarity Testing

In this section, we describe an efficient algorithm for testing C-planarity in connected clustered graphs.

Our algorithm is based on Theorem 1. For a clustered graph C = (G, T), we test whether there is a planar embedding of G such that for each node  $\nu$  of T,  $G - G(\nu)$  is embedded in the same face (the outer face) of  $G(\nu)$ . We try to embed the subgraph induced by each cluster one by one, following a traversal of T from bottom to top. For each node  $\nu$  of T, we test whether  $G(\nu)$  has any planar embeddings that satisfy the conditions of Theorem 1 for  $C(\nu)$ . If we proceed to the root cluster, and such embeddings exist for the root of T, then the clustered graph is C-planar, otherwise it is not C-planar.

### 3.1 Background

We apply the well known PQ-tree technique [3] in our algorithm. The following is a brief definition of the PQ-tree data structure.

The PQ-trees over a set U are trees whose *leaves* are elements of U and whose *internal* nodes are distinguished as being either P-nodes or Q-nodes. Reading the leaves of a tree from left to right yields its *frontier*. We can make two types of transformations on a PQ-tree:

1. arbitrarily *permute* the children of a P-node;

2. reverse the children of a Q-node.

By making such transformations on a PQ-tree, its frontiers form a set of permutations of the leaves. We say the structure of a PQ-tree *expresses* a set of permutations of its leaves.

In the PQ-tree planarity testing algorithm, graphs are decomposed into biconnected components, and each biconnected component is tested for planarity. Each vertex of a biconnected component is labeled by its *st-number* and added in the st-number order. The st-numbering is calculated in the following manner. An st-numbering consists of a biconnected graph G with nvertices and an arbitrary edge (s,t). The vertices of G can be numbered from 1 to n such that vertex s receives number 1, vertex t receives number n, and every vertex except s and t is adjacent both to a lower-numbered and a higher-numbered vertex. Vertices s and t are called the *source* and the *sink* respectively.

We need the following lemma to understand the PQ-tree planarity testing algorithm and also to show the correctness of our algorithm.

**Lemma 1** [19] Suppose that a graph G is a biconnected and st-numbered planar graph. Let  $G_k = (V_k, E_k)$  be the subgraph of G induced by vertices  $V_k = \{1, 2, ..., k\}, 1 \le k \le n$ . If edge (s, t) is drawn on the boundary of the outer face in an embedding of G, then all the vertices and edges of  $G - G_k$  are drawn in the outer face of the plane subgraph  $G_k$  of G.

Using the notation of the above Lemma, a planar drawing of G with all the vertices and edges of  $G - G_k$  drawn in the outer face of  $G_k$  is called a *planar st-drawing* of G.

The PQ-tree planarity testing algorithm maintains a PQ-tree throughout the algorithm. Whenever a vertex  $v_i$  is added, an appropriate operation (called *reduction*) on the PQ-tree is made. After each reduction step, the PQ-tree exactly expresses the set of possible permutations of the edges that connect to  $G_k$  along the outer face of planar st-drawing of  $G_k$ .

The efficient implementation of the PQ-tree technique is fully described in [3].

We use the concept of virtual edge and virtual vertex in our algorithm. For a graph G(V, E) with subgraph G'(V', E'), those edges with one end in V' and the other end in V - V' are called virtual edges of G', and those ends of the virtual edges in V - V' are called virtual vertices of G'.

#### 3.2 A Testing Algorithm

We test C-planarity based on Theorem 1. We determine whether there is a planar embedding of G such that for each node  $\nu$  of T,  $G - G(\nu)$  is embedded in the same face (the outer face) of  $G(\nu)$ .

We try to embed the subgraph induced by each cluster recursively, following a traversal of T from bottom to top. For a node  $\nu$  of T with children  $\mu_1, \ldots, \mu_d$ , we test whether  $G(\nu)$  has any



**Figure 6:** Illustration of Choosing s and t for Each Cluster

planar embeddings that satisfy the conditions of Theorem 1 for  $C(\nu)$ . We find such embeddings for  $C(\nu)$  by combining the possible embeddings of each child cluster  $\mu_i$  which are found recursively. Then we record such embeddings of  $G(\nu)$  for later testing of the parent cluster of  $\nu$ . We construct a *representative graph* that represents all the possible orderings of edges that are incident to cluster  $\nu$  around the outer face of  $G(\nu)$ ; then replace  $G(\nu)$  in G with the representative graph. Graph G is changed every time we process a node of tree T. At the time when the algorithm proceeds to cluster  $\nu$ , planar embeddings of G reflect all planar embeddings of the children of  $\nu$  that satisfy the conditions of Theorem 1.

At cluster  $\nu$ , we not only test whether  $G(\nu)$  is planar, but also test whether the edges that are incident to cluster  $\nu$  can be drawn in the outer face of  $G(\nu)$ . Therefore, we have to take into account the virtual edges of  $G(\nu)$ . We form a graph  $G'(\nu)$  by adding virtual edges to  $G(\nu)$ , and apply the PQ-tree planarity testing algorithm to  $G'(\nu)$ . We add a vertex on each virtual edge of  $G(\nu)$  to distinguish them from each other; and let  $G'(\nu)$  be the graph resulted from connecting the virtual edges of  $G(\nu)$  to a single virtual vertex (see Fig. 6).

The PQ-tree algorithm decomposes a graph into biconnected components and tests each of them for planarity respectively. The following lemma facilitates the application of the PQ-tree algorithm to  $G'(\nu)$ .

**Lemma 2** Suppose that F is a connected subgraph of G. Let F' be the graph constructed by adding virtual edges to F, and connecting each virtual edge to a single virtual vertex. If there are at least two virtual edges in F', then all the virtual edges belong to the same biconnected component of F'.

We apply the PQ-tree testing algorithm to  $G'(\nu)$ . For the biconnected component B that contains the virtual edges, we compute the st-numbering by choosing the single virtual vertex as the sink and any vertex of  $G(\nu)$  that connects to the virtual vertex as the source. If the planarity testing on  $G'(\nu)$  returns TRUE, then  $G'(\nu)$  is planar, and by Lemma 1, all the edges incident to cluster  $\nu$  can be drawn in the outer face of  $G(\nu)$ . Let  $T_{PQ}$  be the nonempty PQ-tree that results when the planarity testing on biconnected component B is completed. The tree  $T_{PQ}$  expresses all the possible orderings of the edges that are incident to cluster of  $\nu$  along the outer face of  $G(\nu)$ . We associate  $T_{PQ}$  with cluster of  $\nu$ .

At each cluster  $\nu$ , we need to determine whether we can combine the planar embeddings of each of its child cluster  $\mu_i$  that satisfy the conditions of Theorem 1 for  $C(\mu_i)$  into planar embeddings of  $G(\nu)$  that satisfy the conditions of Theorem 1 for  $C(\nu)$ . For each child cluster  $\mu_i$  of cluster  $\nu$ , we replace  $G(\mu_i)$  with a representative graph which is constructed from *wheel* graphs.

A wheel graph consists of a vertex called the hub of the wheel and a cycle called the rim of the wheel, such that the hub is connected to every vertex on the rim. (see Fig. 7). Every face of a wheel is a triangle except the face bounded by the rim. We call this face the rim face. If the rim



Figure 7: A "Wheel" Graph

face of a wheel graph is drawn as the outer face, then we say the drawing is a *canonical drawing* of the wheel.

The following lemma shows that wheel graphs have certain properties that can be exploited in construction of representative graphs.

**Lemma 3** Suppose that G is a planar graph with subgraph F. Let F' be the subgraph constructed by adding virtual edges to F. Let  $F_1, F_2, \ldots, F_k$  be a collection of wheel subgraphs of F, such that each wheel graph has a distinguished hub, the hubs only connect to vertices on the corresponding rim in F, and every two wheel graphs have at most one common vertex. If there is a planar drawing D of F' with virtual edges drawn in the outer face of F, then there must also exist a planar drawing  $\mathcal{D}'$  of F' such that :

- The circular ordering of the virtual edges along the outer face of F is preserved.
- Every wheel graph  $F_i$  is drawn canonically.
- $F F_i$  is drawn in the outer face of  $F_i$ .

In a PQ-tree, a P-node corresponds to a cut vertex in the graph the PQ-tree represents; a Q-node corresponds to a biconnected component of the graph; and the leaves correspond to the virtual edges. Given a PQ-tree associated with a graph G, we construct a representative graph  $G_{PQ}$  in the following manner. For each Q-node, we construct a wheel graph; for each P-node, we construct a vertex which serves as a cut vertex connecting the wheels (see Fig. 8). The constructed  $G_{PQ}$  has the following properties:

- The ordering of the virtual edges is the same as the ordering of the leaves of the PQ-tree.
- Biconnected components in G correspond to wheels in  $G_{PQ}$ .
- Cut vertices in G correspond to cut vertices in  $G_{PQ}$ .
- Every vertex in  $G_{PQ}$  has its counterpart in G except the vertices constructed as the hubs of wheels.

The algorithm for testing the C-planarity of a connected clustered graph contains a main loop



Figure 8: The Construction of Representative Graphs

which is a post order traversal of tree T. For each node  $\nu$  of T, we test the planarity of  $G'(\nu)$ , construct a representative graph for  $G(\nu)$ , and replace  $G(\nu)$  with the representative graph in G.

#### Algorithm 1 CPT

- Input: a connected clustered graph C = (G, T);
- Output: a boolean value indicating whether C is C-planar.
- (1) Use the PQ-tree planarity testing algorithm PT to determine whether G is planar. If G is not planar, then return FALSE and exit.
- (2) We proceed on T from bottom to top. For each non-leaf node  $\nu$  of T, perform the following: (2.1) Form graph  $G'(\nu)$  from  $G(\nu)$ .
  - Apply the PQ-tree planarity testing algorithm PT to  $G'(\nu)$ . For the biconnected component that contains the virtual edges, choose the single virtual vertex as the sink t; choose any vertex of  $G(\nu)$  that connects to vertex t as the source to compute st-numbering. Let  $T_{PQ}$  be the resulted PQ-tree when the testing of this component is completed. If any biconnected component is non-planar, then return FALSE and exit.
  - (2.2) Construct representative graph  $G(\nu)_{PQ}$  based on  $T_{PQ}$ .
  - (2.3) Replace subgraph  $G(\nu)$  in G with  $G(\nu)_{PQ}$  and update G.
- (3) When we proceed to the root of T, test the planarity of the updated G using algorithm PT. If graph G is not planar then return FALSE otherwise return TRUE.

The correctness of our algorithm follows immediately from Theorem 1 and Lemmas 1, 2, and 3. It is shown in [3] that for a given graph G with n vertices and m edges, the PQ-tree planarity testing algorithm requires at most O(n) steps. There are algorithms requiring O(n + m) steps to find biconnected components of a graph [1], and to generate st-numbering for each biconnected component [8]. Note that graph G is updated throughout our algorithm. At node  $\nu$  of T, the vertices in G that serve as hubs of wheels have no connection with the rest of the graph except the vertices on the corresponding rim. Therefore, they do not appear in the updated G when the algorithm proceeds to the parent node of  $\nu$  in T. Thus, the number of vertices of the input clustered graph. Since each of the steps 2.1, 2.2 and 2.3 takes linear time in terms of the size of  $G'(\nu)$  which is bounded above by O(n), and they are iterated |T| times, step 2 takes  $O(|T| \cdot n)$  time in all. Both of step 1 and step 3 take O(n) time. Hence algorithm CPT takes  $O(|T| \cdot n)$  time. We have assumed that each node in T has at least two children except for leaf nodes. Thus T has at most 2n nodes. Therefore algorithm CPT.

**Theorem 3** Algorithm CPT tests C-planarity of an n vertex, connected clustered graph C = (G, T)in  $O(n^2)$  time.

### 3.3 An Embedding Algorithm

In this section, we show how to extend the C-planarity testing algorithm *CPT* to a C-planar embedding algorithm *CPEmbed*. The input to algorithm *CPEmbed* is a connected clustered graph. The algorithm returns a C-planar embedding if the input clustered graph is C-planar; otherwise, returns an empty embedding. The algorithm PEmbed [4] replaces algorithm PT in our algorithm CPEmbed. The algorithm PEmbed tests the planarity of a graph and finds a planar embedding if the graph is planar. It uses the same PQ-tree technique as algorithm PT. If a graph is planar, PEmbed records a partial planar embedding of the graph and returns a PQ-tree associated with each biconnected component of the graph. By choosing an ordering of the leaves of each PQ-tree that the PQ-tree accepts, together with the partial planar embedding, a complete planar embedding of the graph can be obtained in linear time [4].

In our algorithm *CPEmbed*, we find a circular ordering of the edges incident to each cluster recursively, following a traversal of tree T from top to bottom. We modify algorithm *CPT* to *CPT*<sup>2</sup> by replacing the primitive PQ-tree planarity testing algorithm *PT* with algorithm *PEmbed*. We use a stack S to record the partial embeddings and the PQ-trees obtained by algorithm *CPT*<sup>2</sup> at each node  $\nu$  of T. Visiting the stack from top to bottom forms a traversal of the inclusion tree T from top to bottom.

The algorithm *CPEmbed* is described as follows.

### Algorithm 2 CPEmbed

- Input: a connected clustered graph C = (G, T); Output: an C-planar embedding  $\mathcal{E}$  of C which consists of a circular ordering of edges incident to each cluster of C.
- (1) Perform algorithm CPT' on clustered graph C = (G, T). At node  $\nu$  of T, push the partial planar embedding and the PQ-tree associated with cluster of  $\nu$  onto a stack S. If CPT' returns FALSE, then return an empty C-planar embedding and exit.

(2) 
$$\mathcal{E} = \emptyset$$
.

While S is not empty, perform the following:

(2.1) Pop the partial embedding and the PQ-tree from stack S which corresponds to node  $\nu$  of T.

Let  $ORD(\nu)$  be the circular ordering of edges incident to cluster  $\nu$  in embedding  $\mathcal{E}$ . Choose  $ORD(\nu)$  as the ordering of the leaves of the PQ-tree associated with  $G'(\nu)$ . Find a complete planar embedding  $\mathcal{H}_{\nu}$  of  $G(\nu)$  according to  $ORD(\nu)$  and the partial embedding popped from the stack.

- (2.2) Call procedure  $Formalize(\mathcal{H}_{\nu})$  (below) to modify  $\mathcal{H}_{\nu}$ , such that for each wheel subgraph F of  $G(\nu)$ , F is embedded canonically, and the vertices and edges of G-F are embedded in the outer face (the rim face) of F.
- (2.3) For each child  $\mu_i$  of  $\nu$ , find the circular ordering  $ORD(\mu_i)$  of the edges incident to cluster  $\mu_i$  according to  $\mathcal{H}_{\nu}$ ; and let  $\mathcal{E}$  be  $\mathcal{E} \cup ORD(\mu_i)$ .

Procedure Formalize changes a planar embedding of graph G, such that each wheel subgraph  $F_i$  is embedded canonically, and  $G - F_i$  is embedded in the outer face (the rim face) of  $F_i$ . By Lemma 3, this kind of embedding exists. We formalize the embedding of G by moving the part of  $G - F_i$ which are not embedded in the rim face of  $F_i$  to the other side of the rim of  $F_i$ . The following description of procedure Formalize completes our description of algorithm *CPEmbed*.

#### **Procedure** Formalize( $\mathcal{H}_{\nu}$ )

For each wheel subgraph  $F_i$  of  $G(\nu)$  updated in CPT':

For each vertex x on the rim of the wheel  $F_i$ :

Suppose that h is the hub of the wheel,  $r_1$ ,  $r_2$  are the two vertices on the rim adjacent to x, and  $(h, v_1, v_2, \ldots, v_p, r_1, v_{p+1}, \ldots, v_q, r_2, v_{q+1}, \ldots, v_l, h)$  is the circular order of the vertices which connect to x defined by  $\mathcal{H}_{\nu}$ . We change this circular ordering to  $(h, r_1, v_1, v_2, \ldots, v_p, v_{p+1}, \ldots, v_q, v_{q+1}, \ldots, v_l, r_2, h)$ .

The correctness of this algorithm follows from the correctness of *PEmbed* and *CPT*.

To compute the running time of algorithm CPT', we first note that algorithm PEmbed takes linear time (see [4]). Then, by a similar argument as for algorithm CPT, algorithm CPT' takes  $O(n^2)$  time. Consequently, step 1 of algorithm CPEmbed takes  $O(n^2)$  time. According to [4], step 2.1 takes linear time in terms of the size of  $G'(\nu)$ . Step 2.2, and 2.3 also take linear time in terms of the size of  $G(\nu)$ . Therefore, step 2 requires time  $O(n \cdot |T|) = O(n^2)$  in all. Thus, algorithm CPEmbed require  $O(n^2)$  time. The following theorem summarizes the performance of algorithm CPEmbed.

**Theorem 4** Algorithm CPEmbed finds a C-planar embedding of an n vertex, connected clustered graph C = (G, T) in  $O(n^2)$  time.

## 4 Conclusion and Open Problems

In this paper, we have introduced a graph model known as *clustered graphs* and investigated the planarity of clustered graphs. We have presented an efficient algorithm for testing C-planarity in connected clustered graphs, and also extend the C-planarity testing algorithm to a C-planar embedding algorithm.

Some interesting open problems include:

- Can we improve the performance of the proposed algorithms to linear time?
- For non-connected clustered graphs, with a given embedding of the underlying graph, how do we test whether the embedding admits a C-planar drawing?
- Can we find a polynomial time algorithm that tests C-planarity of non-connected clustered graphs, or show that the problem is NP-hard?

### References

- A. V. Aho, J. E. Hopcroft, and J. D. Ullman. The Design and Analysis of Computer Algorithms. Addison-Wesley, Reading, Mass., 1974.
- [2] Claude Berge. *Hypergraphs*. North-Holland, 1989.
- [3] K. Booth and G. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of Computer and System Sciences*, 13:335–379, 1976.
- [4] N. Chiba, T. Nishizeki, S. Abe, and T. Ozawa. A linear algorithm for embedding planar graphs using PQ-trees. J. of Computer and Sytem Sciences, 30(1):54-76, 1985.
- [5] M. Consens, A. Mendelzon, and A. Ryman. Visualizing and querying software structures. In 14th International Conference on Software Engineering (Melbourne), pages 11 – 15, 1992.
- [6] H. de Fraysseix and P. Rosenstiehl. A depth-first-search characterization of planarity. Annals of Discrete Mathematics, 13:75-80, 1982.

- [7] P. Eades, B. McKay, and N. Wormald. On an edge crossing problem. In Proc. 9th Australian Computer Science Conf., pages 327-334, 1986.
- [8] S. Even and R. E. Tarjan. Computing an st-numbering. Theoretical Computer Science, 2:339-344, 1976.
- M.R. Garey and D.S. Johnson. Crossing number is NP-complete. SIAM J. Algebraic and Discrete Methods, 4(3):312-316, 1983.
- [10] D. Harel. On visual formalisms. Communications of the ACM, 31(5):514-530, 1988.
- [11] J. Hopcroft and R. E. Tarjan. Efficient planarity testing. Journal of ACM, 21(4):549-568, 1974.
- [12] Silicon Graphics Inc. CASEVision/workshop user's guide. Silicon Graphics Inc, 1992. Volumes I and II.
- [13] T. Kamada. Visualizing Abstract Objects and Relations. World Scientific Series in Computer Science, 1989.
- [14] J. Kawakita. The KJ method a scientific approach to problem solving. Technical report, Kawakita Research Institute, Tokyo, 1975.
- [15] Wei Lai. Building Interactive Digram Applications. PhD thesis, Department of Computer Science, University of Newcastle, Callaghan, New South Wales, Australia, 2308, June 1993.
- [16] A. Lempel, S. Even, and I. Cederbaum. An algorithm for planarity testing of graphs. In *Theory of Graphs, International Symposium (Rome 1966)*, pages 215–232. Gordon and Breach, New York, 1967.
- [17] K. Misue and K. Sugiyama. An overview of diagram based idea organizer: D-abductor. Technical Report IIAS-RR-93-3E, ISIS, Fujitsu Laboratories, 1993.
- [18] H.A. Muller. Rigi A Model for Software System Construction, Integration, and Evalution based on Module Interface Specifications. PhD thesis, Rice University, 1986.
- [19] T. Nishizeki and N. Chiba. Planar Graphs: Theory and Algorithms, Annals of Discrete Mathematics 32. North-Holland, 1988.
- [20] S. C. North. Drawing ranked digraphs with recursive clusters. preprint, 1993. Software Systems and Research Center, AT & T Laboratories.
- [21] K. Sugiyama and K. Misue. Visualization of structural information: Automatic drawing of compound digraphs. *IEEE Transactions on Systems, Man and Cybernetics*, 21(4):876–892, 1991.
- [22] C. Williams, J. Rasure, and C. Hansen. The state of the art of visual languages for visualization. In Visualization 92, pages 202 - 209, 1992.
- [23] Rebecca Wirfs-Brock, Brian Wilkerson, and Lauren Wiener. Designing Object-Oriented Software. P T R Prentics Hall, Englewood Cliffs, NJ 07632, 1990.