Straight-Line Drawing Algorithms for Hierarchical Graphs and Clustered Graphs

Peter Eades[1] Qing-Wen Feng[1] Xuemin Lin[2]

[1] Department of Computer Science and Software Engineering, University of Newcastle, NSW 2308, Australia. Email: {eades,qwfeng}@cs.newcastle.edu.au
[2] Department of Computer Science, University of Western Australia, Nedlands, WA 6009, Australia. Email: lxue@cs.uwa.edu.au

(extended abstract)

Abstract. Hierarchical graphs and clustered graphs are useful nonclassical graph models for structured relational information. Hierarchical graphs are graphs with layering structures; clustered graphs are graphs with recursive clustering structures. Both have applications in CASE tools, software visualization, VLSI design, etc. Drawing algorithms for hierarchical graphs have been well investigated. However, the problem of straight-line representation has not been addressed. In this paper, we answer the question: does every planar hierarchical graph admit a planar straight-line hierarchical drawing? We present an algorithm that constructs such drawings in $O(n^2)$ time. Also, we answer a basic question for clustered graphs, i.e. does every planar clustered graph admit a planar straight-line drawing with clusters drawn as convex polygons? A method for such drawings is provided in this paper.

1 Introduction

Graph drawing algorithms are widely used in graphical user interfaces of many software systems. Examples include CASE tools, reverse engineering systems and software design systems. A good picture is worth a thousand words, while a poor drawing can be misleading.

As the information that we want to visualize becomes more and more complicated, we need more structure on top of the classical graph model. Hierarchical graphs are directed graphs with layering structures (see Fig. 1). They appear in applications where hierarchical structures are involved e.g. PERT networks and organization charts [19, 10]. Clustered graphs are graphs with clustering structures (see Fig. 2) which appear in many structured diagrams [18, 11, 12].

A hierarchical graph is conventionally drawn with vertices of a layer on the same horizontal line, and arcs as curves monotonic in y direction (see Fig. 1). A hierarchical graph is *hierarchical planar (h-planar)* if it admits a drawing without edge crossings. For a clustered graph, the clustering structure is represented by a closed curve that defines a region. The region contains the drawing of all the vertices which belong to that cluster (see Fig. 2). A clustered graph is *compound*

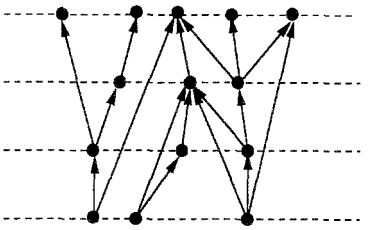

Fig. 1. An Example of a Hierarchical Graph

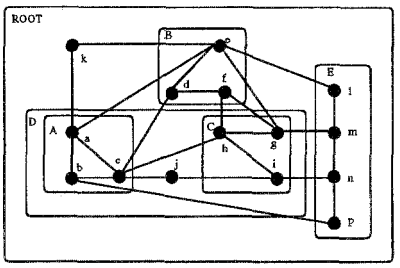

Fig. 2. An Example of a Clustered Graph

planar (c-planar) if it admits a drawing with no edge crossings or edge-region crossings.

One of the basic graph drawing convention consists of representing edges as straight-line segments. For classical graphs, it has been shown independently by Fary [7], Stein [17], and Wagner [21] that every planar graph admits a straight-line drawing without edge crossings. Tutte [20] proved that every 3-connected planar graph admits a planar straight-line drawing where all the face boundaries are drawn as convex polygons.

In this paper, we answer the question: does every planar hierarchical graph admit a planar straight-line hierarchical drawing? Although many results have been obtained on drawing hierarchical graphs [19, 10, 3], the basic problem of planar straight-line drawings has not been studied. It has been shown by Di Battista and Tamassia [1] that every planar st-graph admits an *upward drawing* i.e. a drawing where all arcs are drawn as straight-line segments pointing upward. However, the problem for hierarchical graphs is different. We have more constraints: vertices of the same layer should be drawn on the same horizontal line; layers should be equal distance apart. A method to construct straight-line drawings of hierarchical graphs has been presented by Eades, Lin and Tamassia [5]. However, it is restricted to a special class of hierarchical graphs known as *collapsible free* graphs.

The second question answered in this paper is for clustered graphs: does every

planar clustered graph admit a planar straight-line drawing with clusters drawn as convex polygons? Although an algorithm for straight-line drawing of clustered graphs has been presented by Feng, Cohen and Eades [8], it does not apply to all clustered graphs. If the graph induced by a cluster is not biconnected, then its external facial cycle is not a simple cycle. In this case, we cannot use the drawing of its external facial cycle as the region boundary, since it cannot form a simple region. This question that we answer in this paper was posed as an open problem in [8].

The rest of the paper is organized as follows. In section 2, we present some terminology for hierarchical graphs. In section 3, we prove that every planar hierarchical graph admit a planar straight-line drawing. An $O(n^2)$ time algorithm that produces such drawings is presented. We introduce the model of clustered graphs in section 4. In section 5, we show that every planar clustered graph admits a planar straight-line convex cluster drawing. This is accomplished by transforming clustered graphs into hierarchical graphs. Section 6 concludes with some remarks and discussion.

2 Hierarchical Graphs

Hierarchical graphs are directed graphs where vertices are assigned to layers. As described in [5], a *hierarchical graph* $H = (V, A, \lambda, k)$ consists of a directed graph (V, A), a positive integer k, and, for each vertex u, an integer $\lambda(u) \in 1, 2, ..., k$, with the property that if $(u, v) \in A$, then $\lambda(u) < \lambda(v)$. For $1 \leq i \leq k$, the set $\{u : \lambda(u) = i\}$ is the ith *layer* of H and is denoted by L_i. The *span* of an arc (u, v) is $\lambda(v) - \lambda(u)$. An arc of span greater than one is *long*, and a hierarchical graph with no long arcs is *proper*.

A hierarchical graph is conventionally drawn with layer L_i on the horizontal line $y = i$, and arcs as curves monotonic in y direction. If no pair of nonincident arcs intersect in the drawing, we say it is a *hierarchical planar (h-planar)* drawing. Note that a nonproper hierarchical graph can be transformed into a proper hierarchical graph by adding dummy vertices on long arcs. It can be shown that a nonproper hierarchical graph is h-planar if and only if the corresponding proper hierarchical graph is h-planar. A *hierarchical planar embedding* of a proper hierarchical graph is defined by the ordering of vertices on each layer of the graph. Note that every such embedding has an unique external face. It is easily shown that every proper h-planar graph admits a *straight-line hierarchical drawing*, that is, a drawing where arcs are drawn as straight-line segments. However, for nonproper hierarchical graphs, the problem is not trivial, since no bends are allowed on long arcs.

We call a planar embedded graph a *plane graph*. If a hierarchical plane graph has only one source s and one sink t, we call it a *hierarchical-st plane graph*. We will show that every hierarchical plane graph can be extended to a hierarchical-st plane graph. The following lemma gives some basic properties which are useful to our algorithm.

Lemma 1. *Let H be a hierarchical-st plane graph, then: (a) Every biconnected component of H is also a hierarchical-st plane graph. (b) If B_1 and B_2 are two biconnected components of H, and u is a vertex of B_1 and u is not a cut vertex, then either for all vertices v of B_2, $\lambda(u) < \lambda(v)$, or for all vertices v of B_2, $\lambda(u) > \lambda(v)$. (c) H has a planar straight-line hierarchical drawing if and only if each of its biconnected component has a planar straight-line hierarchical drawing.*

With the above lemma, we can assume that we are given a hierarchical-st plane graph that is biconnected, which implies its external face is bounded by a simple cycle if the graph is not just a single edge. Since hierarchical graphs are directed graphs, the terms "cycle" and "path" that we use in the rest of the paper are all for the underlying undirected graphs. To denote a cycle of a plane graph, we use the sequence of vertices on the cycle in clockwise order. We say a path is *monotonic* if the directions of the edges (arcs) do not change along the path.

The following lemma is also very useful for to algorithm.

Lemma 2. *Let H be a hierarchical-st plane graph which is biconnected, and has external facial cycle $C = (v_1, \ldots, v_k, v_1)$. Suppose that $\mathcal{P} = (v_i, x_1, \ldots, x_l, v_j)$ is a monotonic path from v_i to v_j in H, and the vertices of $\mathcal{P}$ are not on C except v_i and v_j. Let H_1 and H_2 be the two subgraphs bounded inside by cycles $C_1 = (v_1, \ldots, v_i, x_1, \ldots, x_l, v_j, \ldots, v_k, v_1)$ and $C_2 = (v_i, \ldots, v_j, x_l, \ldots, x_1, v_i)$ inclusive. Then H_1 and H_2 are hierarchical-st plane graphs and are biconnected (see Fig.3).*

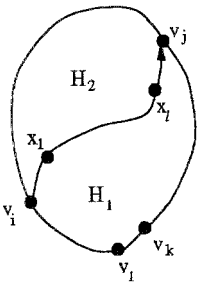

Fig. 3. Illustration of Lemma 2

3 Straight-Line Hierarchical Drawings

In this section, we show that given an n-vertex hierarchical plane graph, we can compute a planar straight-line hierarchical drawing in $O(n^2)$ time.

We apply a divide and conquer approach: divide the hierarchical graph into subgraphs, compute the drawings of the subgraphs, and obtain a drawing of the graph by combining the drawings of the subgraphs. The key part of this approach is to find a suitable partition.

Our method works on *triangular hierarchical-st plane graphs*. In a triangular hierarchical-st plane graph, the boundary of every nonexternal face consists of exactly three edges. We prove that every hierarchical plane graph can be extended to a triangular hierarchical-st plane graph which admits a straight-line drawing with a prescribed polygon as its external face. We provide a straight-line drawing algorithm based on our proof.

In our method, we are given a prescribed polygon as the external face of the drawing. Note that there can be vertices of the external facial cycle which are not drawn as apexes of the polygon. This can give some problems if the external facial cycle has a chord (i.e. an edge between two nonconsecutive vertices). To deal with this problem, we need some terminology. Let H be a hierarchical-st plane graph with source s and sink t; let cycle C be the boundary of its external face; let polygon P be a straight-line hierarchical drawing of cycle C. We say that P is *feasible* for H if the following conditions hold:

- P is a convex polygon.
- If cycle C has a chord (x, y), then on each of the two paths of cycle C between x and y, there exists a vertex v which is drawn as an apex of polygon P.

In our divide and conquer approach, we distinguish two situations. If the external facial cycle has a chord, we simply divide the graph into two parts with the chord. Otherwise, we find a vertex not on the external facial cycle, such that there are three monotonic paths that connect the vertex with the external facial cycle. Therefore, by using Lemma 2 twice, we divide the graph into three parts. The following lemma is useful in finding such vertex in the graph. We need some more terminology. For a hierarchical graph H with vertices u and v, an *st-component for* (u, v) is the union of all subgraphs of H for which u is the unique source, and v is the unique sink. In other words, the st-component for (u, v) is the maximal subgraph with a single source u and a single sink v.

Lemma 3. *Let H be a triangular hierarchical-st plane graph with single source s and single sink t. Suppose that the external facial cycle C of H has no chords. Let v be a vertex on cycle C other than s or t. Suppose that $H_{st}(v)$ is the st-component of the hierarchical graph $H - v$ for vertex pair (s, t). Then there exists a vertex w incident to v in H and not on cycle C, such that $w \in H_{st}(v)$ and w has the following properties: (1) Vertex w is on the external face of $H_{st}(v)$. (2) vertex w is not a cut vertex of $H_{st}(v)$. (3) Suppose that $H_{main}(v)$ denotes the biconnected component of $H_{st}(v)$ that contains w. Then the external facial cycle of $H_{main}(v)$ consists of two paths: path $(x, \ldots, y)$ which belongs to C, and path $(y, \ldots, w, \ldots, x)$ which does not belong to C; and path $(x, \ldots, w, \ldots, y)$ is monotonic (see Fig.4).*

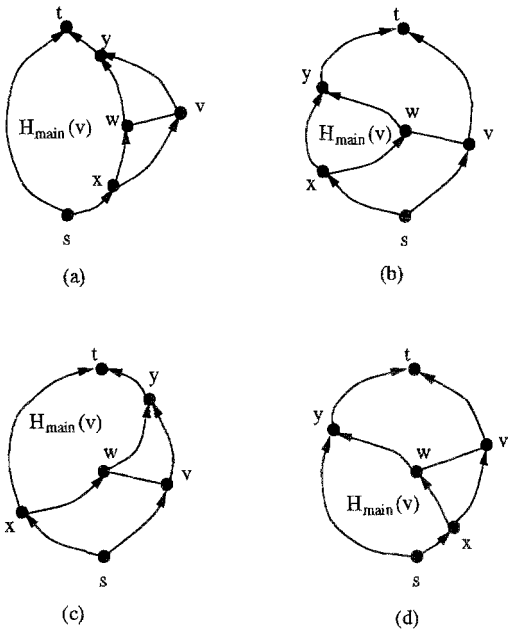

Fig. 4. Possible Partitions of H

Sketch of Proof: Fig 5(a)-(d) shows all possible situations of the path formed by the vertices incident to v. We show in each situation there exists such vertex w incident to v in H and not on cycle C , such that $w \in H_{st}(v)$. We show that the situation illustrated in Fig. 5(e) would not occur. The proofs for *Property 1* and *Property 2* of such vertex w are immediate. For *Property 3*, Fig. 4 illustrates all possible situations of the external face of $H_{main}(v)$, and we show that the property holds for all these situations. □

Theorem 4. *Suppose that H is a triangular hierarchical-st plane graph, and polygon P is a straight-line hierarchical drawing of its external facial cycle C. If P is feasible for H, then there exists a planar straight-line hierarchical drawing of H with external face P.*

Proof. We prove by induction on the number n of vertices of H. The basis of the induction, $n = 3$ is immediate. Now, assume that the theorem holds for graphs with less than n vertices. We distinguish two cases:

Case 1: The external facial cycle C of H has a chord (x, y). By Lemma 2, chord (x, y) divides H into two subgraphs H_1 and H_2. We draw a straight line segment between x and y , which divides P into two polygons P_1 and P_2. It can be verified that P_1 and P_2 are feasible for H_1 and H_2. Since both H_1 and H_2 have less than n vertices, by induction, there exist straight-line hierarchical drawings of H_1 and H_2 with external faces P_1 and P_2. Hence, by combining the

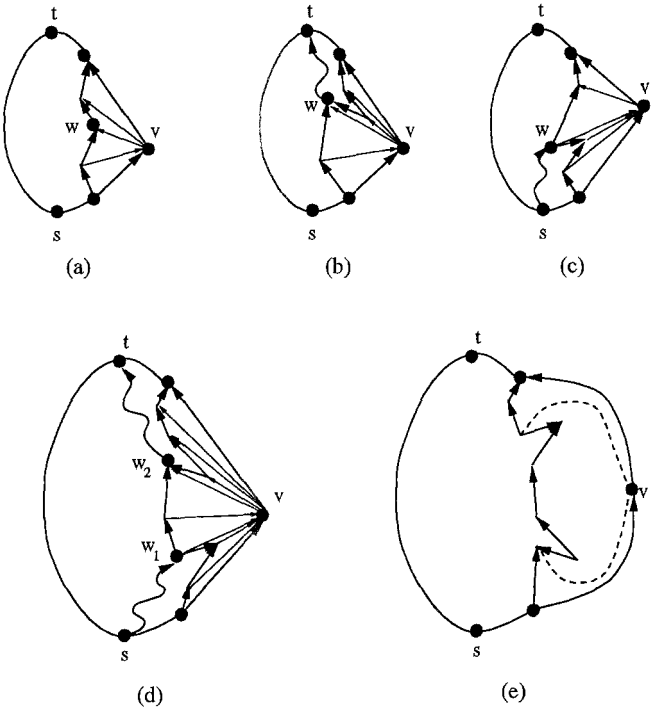

Fig. 5. Illustration of the Proof of Lemma 3

two drawings, we obtain a straight-line hierarchical drawing of H with external face P.

Case 2: The external facial cycle C of H has no chords. There exists a vertex v other than the source s or sink t on the external face, such that v is drawn as an apex of P. Otherwise P would not be a convex polygon. By Lemma 3, there exists a vertex w incident to v but not on cycle C such that $w \in H_{st}(v)$ and w has those properties stated in the lemma . Hence, we have a monotonic path $(x, \ldots, w, \ldots, y)$ inside H, and also an edge (w, v) inside H (see Fig. 4). Using Lemma 2 twice, they divide H into three parts:

- $H_{main}(v)$ bounded by cycle $(x, \ldots, y, \ldots, w, \ldots x)$;
- $H_{asso1}(v)$ bounded by cycle $(x, \ldots, w, v, \ldots x)$;
- $H_{asso2}(v)$ bounded by cycle $(y, \ldots, v, w, \ldots y)$.

Now we need to adjust this partition such that a feasible polygon can be computed for each part. Note that path $(x, \ldots, w)$ has no chords in $H_{asso1}(v)$, otherwise it would not belong to the external face of $H_{main}(v)$. Similarly, path $(w, \ldots, y)$ has no chords in $H_{asso2}(v)$.

Now consider chords in $H_{main}(v)$. We reduce $H_{main}(v)$ and accordingly, extend $H_{asso1}(v)$ or $H_{asso2}(v)$ to eliminate such chords. If path

$(x, \ldots, u_1, \ldots, u_2, \ldots, w)$ has a chord (u_1, u_2), we modify the path to $(x, \ldots, u_1, u_2, \ldots, w)$. Also, we modify path $(w, \ldots, y)$ in similar way if it has a chord. Graphs $H_{main}(v)$, $H_{asso1}(v)$ and $H_{asso2}(v)$ change accordingly when we modify the paths. After the modification, paths $(x, \ldots, w)$ and $(w, \ldots, y)$ have no chords in $H_{main}(v)$, $H_{asso1}(v)$ and $H_{asso2}(v)$.

It can be verified that $H_{main}(v)$, $H_{asso1}(v)$ and $H_{asso2}(v)$ are triangular hierarchical-st plane graphs.

Let $H_{frame}(v)$ be the graph that consists of only the external faces of $H_{main}(v)$, $H_{asso1}(v)$ and $H_{asso2}(v)$. Now $H_{frame}(v)$ is a hierarchical-st plane graph with the same external face as H. Hence polygon P is also a hierarchical planar drawing of the external face of $H_{frame}(v)$. We need to find a position for w such that the drawing of the three internal faces of $H_{frame}(v)$ are convex polygons, and therefore feasible for $H_{main}(v)$, $H_{asso1}(v)$ and $H_{asso2}(v)$. We compute the x coordinate of vertex w using the following equation derived from [5]:

$$x(a) = \frac{\frac{1}{2d_a^+}\sum_{b \to a}\frac{x(b)}{l} + \frac{1}{2d_a^-}\sum_{a \to b}\frac{x(b)}{l}}{\frac{1}{2d_a^+}\sum_{b \to a}\frac{1}{l} + \frac{1}{2d_a^-}\sum_{a \to b}\frac{1}{l}}. \quad (1)$$

Here, $x(a)$ and $x(b)$ denote the x coordinates of vertices a and b respectively; d_a^+ denotes the indegree of a; d_a^- denotes the outdegree of a; and $l = |\lambda(a) - \lambda(b)|$.

This formula computes the x coordinate of vertex w as a weighted barycenter of its neighbors x, y and v. (Vertices with degree 2 are not considered here.) We place other internal vertices of $H_{frame}(v)$ (those with degree 2) onto the line segments from x to w and from w to y at appropriate horizontal lines. It can be shown that the drawings P_0, P_1 and P_2 of the three internal faces of $H_{frame}(v)$ are convex polygons [5]. Note that edges on path $(x, \ldots, w)$ are drawn on the same line, so are the edges on path $(w, \ldots, y)$. However, since there are no chords on these paths, P_0, P_1 and P_2 are feasible for $H_{main}(v)$, $H_{asso1}(v)$ and $H_{asso2}(v)$ respectively. As each of $H_{main}(v)$, $H_{asso1}(v)$ and H_{asso2} has less than n vertices, by induction, there exist straight-line hierarchical drawings of $H_{main}(v)$, $H_{asso1}(v)$ and H_{asso2} with external faces P_0, P_1 and P_2. Hence, by combining these drawings, we obtain a straight-line hierarchical drawing of H with external face P.

The algorithm to compute a planar straight-line hierarchical drawing is based on the proof of Theorem 4. The input of the algorithm is a hierarchical plane graph H; the output is a planar straight-line hierarchical drawing of H. The algorithm consists of two phases: *Preprocessing* and *Drawing*. In the preprocessing phase, we extend the hierarchical plane graph to a triangular-st plane graph. The drawing phase is a recursive procedure that actually constructs the drawing of the graph. Now we describe them in more detail.

Preprocessing. We extend the hierarchical plane graph to a triangular hierarchical-st plane graph in three steps. (1) Extend the hierarchical plane graph such that all the sources and sinks lie on the bottom layer and top layer. We can use a method similar to those in [1, 15] which performs two sweeps from bottom to top and from top to bottom to eliminate the sources and sinks in

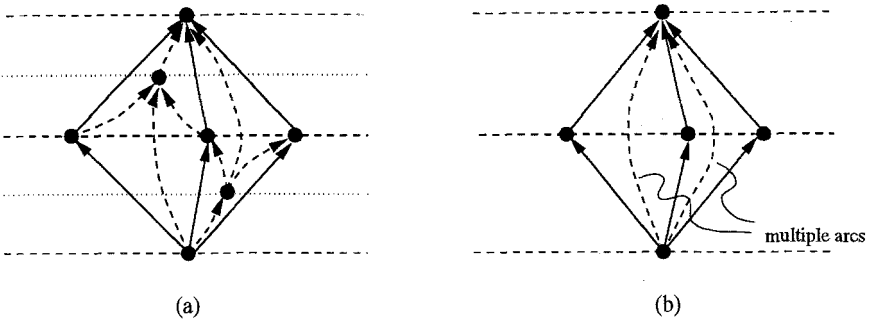

Fig. 6. Triangulating the Hierarchical Graph

between. (2) Add one more vertex s below the bottom layer and connect it to all the sources; then add one more vertex t above the top layer and connect all the sinks to it. Therefore, a hierarchical-st plane graph is obtained. (3) Extend the hierarchical-st plane graph to a triangular one as follows: insert a layer between every two consecutive layers (This ensures that original layers still to be evenly distributed.); add a "star" structure inside each face (see Fig. 6(a)), and place the center of the each star on an inserted layer. After this, every internal face is bounded by exactly three edges. Note that this operation does not increase the size of the graph by more than a constant. This triangulation method is a little unusual, but necessary. Fig. 6(b) shows that if we do not add new vertices, multiple arcs can be produced. Further, we cannot allow dummy nodes on the arcs because this may introduce bends. Also note that no arcs are allowed between two vertices of the same layer.

Drawing. The drawing phase is realized with a recursive procedure which is based on the proof of Theorem 4. Firstly, it is easy to find a feasible polygon P for an input graph H. Then we call the procedure and obtain a drawing of H.

Procedure *Straight-line_Hierarchical_Draw*$(H, \mathcal{E}, P, \Gamma)$
{H is a triangular hierarchical-st plane graph with planar embedding $\mathcal{E}$; P is a polygon feasible for H. Γ is a planar straight-line hierarchical drawing of H returned by the procedure.}

(1) If H has three vertices, then draw H as P. Let $\Gamma = P$, exit.
(2) Check H for possible chords of the external facial cycle $\mathcal{C}$.
(3) If $\mathcal{C}$ has a chord (x, y), then:
  (3.1) divide H into H_1 and H_2 with chord (x, y); draw straight-line segment between x and y in P; divide P into P_1 and P_2;
  (3.2) call Straight-line_Hierarchical_Draw(H_1, P_1, Γ_1);
  call Straight-line_Hierarchical_Draw(H_2, P_2, Γ_2);
  (3.3) Let $\Gamma = \Gamma_1 \cup \Gamma_2$, exit.
(4) If $\mathcal{C}$ has no chords, then:

(4.1) choose vertex v on $\mathcal{C}$; find the st-component $H_{st}(v)$ of $H - v$ for the source and sink pair (s, t); choose vertex w that is incident to v but not on $\mathcal{C}$, and on the external face of $H_{st}(v)$.

(4.2) find the biconnected component $H_{main}(v)$ of $H_{st}(v)$ that contains w; modify the two paths $(w, \ldots, x)$ and $(y, \ldots, w)$ on the external face of $H_{st}(v)$ to avoid chords;

(4.3) construct $H_{frame}(v)$ and compute its drawing using equation 1; hence obtain polygons P_0, P_1 and P_2 of the three internal faces of $H_{frame}(v)$;

(4.4) divide H into $H_{main}(v)$, $H_{asso1}(v)$ and $H_{asso2}(v)$ with paths $(w, \ldots, x)$, $(y, \ldots, w)$ and edge (w, v);

(4.5) call Straight-line_Hierarchical_Draw($H_{main}(v)$, P_0, Γ_0);
call Straight-line_Hierarchical_Draw($H_{asso1}(v)$, P_1, Γ_1);
call Straight-line_Hierarchical_Draw($H_{asso2}(v)$, P_2, Γ_2);

(4.6) Let $\Gamma = \Gamma_0 \cup \Gamma_1 \cup \Gamma_2$, exit.

In the preprocessing phase, each of the three steps takes linear time.

In the drawing phase, we maintain an edge list and a face list for each hierarchical-st plane subgraph through the procedure *Straight-line_Hierarchical_Draw*. With this data structure, we can check for chords of a cycle $\mathcal{C}$ (or path) of a graph in linear time; we can divide graph H with a chord of its external facial cycle or a path inside it in linear time. An st-component for vertex pair (u, v) can be found in linear time by performing depth-first search from u in one direction, and from v in the opposite direction. Also, the biconnected components of a graph can be found in linear time [2]. In the procedure call of the drawing phase, every vertex is processed at most $O(n)$ times by step (2) or step (4.2). Every vertex is processed also at most $O(n)$ times by step (3.1) or steps (4.2) and (4.4). Every vertex is processed once by step (4.3) for computing its x coordinate. Consequentially, the drawing phase costs $O(n^2)$ time.

Note that each edge appears in at most two subgraphs through the procedure. Therefore, our algorithm requires linear space.

The following theorem summarizes the performance of the algorithm.

Theorem 5. *Let H be a hierarchical plane graph with n vertices. The above algorithm constructs a planar straight-line hierarchical drawing for H in $O(n^2)$ time and $O(n)$ space.*

Based on our results for hierarchical graphs, we consider the straight-line drawing problem for clustered graphs in the following sections.

4 Clustered Graphs

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 ν of

T represents a *cluster* $V(\nu)$ of the vertices of G that are leaves of the subtree rooted at ν. Note that tree T describes an inclusion relation between clusters.

In a *drawing* of a clustered graph $C = (G, T)$, graph G is drawn as points and curves as usual. For each node ν of T, the cluster is drawn as a simple closed region R that contains the drawing of $G(\nu)$, 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 $V(\nu)$, then the drawing of e is completely contained in R.

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 C has a c-planar drawing then we say that it is *c-planar* (see Fig. 2).

An edge is said to be *incident* to a cluster $V(\nu)$ if one end of the edge is a vertex of that cluster but the other end is not in $V(\nu)$. 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 results from [9] characterize c-planarity in a way which can be exploited by our drawing algorithm.

Theorem 6. *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.*

From Theorem 6, we can assume that we are given a connected clustered graph when drawing a c-planar clustered graph. According to [13], a c-planar embedding of a connected clustered graph can be found in linear time. In the rest of the paper, we assume there are no degenerated clusters, that is, every nonleaf node of T has at least two children.

5 Straight-Line Convex Cluster Drawings

One of the fundamental questions in planar clustered graph drawing is: does every c-planar clustered graph admit a planar drawing such that edges are drawn as straight-line segments and clusters are drawn as convex polygons? In this section, we answer this question based on our results for hierarchical graphs. We transform a clustered graph into a hierarchical graph, and construct a straight-line convex cluster drawing on top of the straight-line hierarchical drawing.

By Theorem 6, we assume that we are given a c-planar connected clustered graph $C = (G, T)$ with a c-planar embedding. Roughly speaking, our algorithm works as follows. First, we triangulate G (including triangulating the external

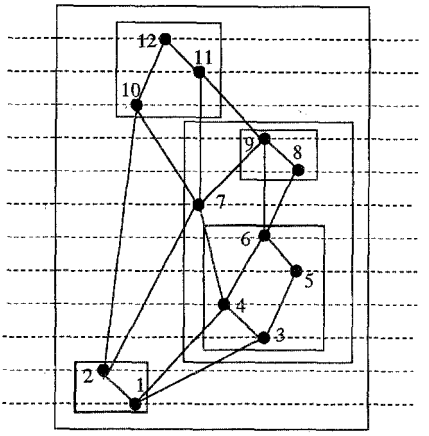

Fig. 7. Clustered Graph → Hierarchical Graph

face) [16]; then compute an *st* numbering [3] of the vertices of G such that the vertices that belong to the same cluster are numbered consecutively. We call this numbering *c-st numbering.* This numbering gives us a layer assignment of the vertices of G. Hence, the clustered graph is transformed to a hierarchical graph (see Fig. 7), and each cluster has consecutive layers. Because of this property, we show that a straight-line convex cluster drawing can be constructed from the straight-line hierarchical drawing.

The critical part of this method is the construction of the *c-st* numbering. To ensure that the vertices of the same cluster are numbered consecutively, we need to compute an ordering of the child clusters for every parent cluster ν. To do this, we construct a graph $F(\nu)$ from $G(\nu)$ by shrinking each child cluster of ν to a vertex while preserving the embedding. First of all, we add a dummy node on every edge of G; this prevents edges from collapsing into one edge when shrinking. We use a top down approach , ordering the children of the root first.

For the root node γ of T, the graph $F(\gamma)$ is constructed as follows. We choose an edge e of G that does not belong to any other cluster except the root cluster. Since we are given a connected clustered graph, such an edge exists. We choose s and t to be the two ends of this edge. Then shrink every child cluster of the root into a single vertex and preserve the planar embedding in the meantime. The resulting graph is $F(\gamma)$. Every vertex of $F(\gamma)$ represents a child cluster of the root cluster. Since *st* numberings are constructed on biconnected graphs, we need the following lemma:

[3] Given any edge (s,t) in a biconnected graph G with n vertices, a *st* numbering for G is defined as follows. The vertices of G are numbered from 1 to n so that vertex s receives number 1, vertex t receives number n, and any 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. Such a numbering is an *st* numbering for G. An *st* numbering of a biconnected graph can be computed in linear time [6].

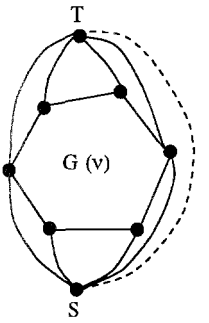

Fig. 8. Illustration of Computing c-st Numbering

Lemma 7. *Suppose that $C = (G, T)$ is a c-planar clustered graph, γ is the root of T, and G is triangulated. Then the graph $F(\gamma)$ is biconnected.*

Sketch of Proof: Since G has been triangulated and we preserve the embedding in the shrinking operation, it can be shown that every face in the resulting graph $F(\gamma)$ is bounded by a simple cycle. Therefore $F(\gamma)$ is biconnected. □

Since $F(\gamma)$ is biconnected, we can order the children of the root Γ by computing an *st* numbering, choosing the vertex that represents the cluster where s belongs as the source, and the vertex that represents the cluster where t belongs as the sink.

We proceed top down from the root. For a nonroot node ν, we construct a graph $F(\nu)$ in a similar but slightly more complex way; $F(\nu)$ depends on the place of ν in the ordering of ν and its siblings. For each child cluster μ of ν, we shrink graph $G(\mu)$ into one vertex while preserving the planar embedding. For those edges that connect cluster ν with clusters which are ordered before ν (note that this order is computed recursively as mentioned above), we connect them to a single vertex S. For those edges that connect cluster ν with clusters which are ordered after ν, we connect them to a single vertex T (see Fig. 8). Finally, we connect S and T in the external face of $G(\nu)$, hence forming $F(\nu)$. Here, if vertex $s(t)$ belongs to cluster ν, we simply choose the vertex which represents the child cluster that contains $s(t)$ as $S(T)$. We need the following lemma; its proof is similar to that of Lemma 7.

Lemma 8. *Suppose that $C = (G, T)$ is a c-planar clustered graph, and G is triangulated. For every node ν of T, the graph $F(\nu)$ is biconnected.*

□

With the lemma, we order every vertex of $F(\nu)$ by computing an *st* numbering, choosing vertex S as the source, and vertex T as the sink.

Now, each cluster ν is assigned a number of order within its parent. Therefore, a recursive hierarchy of orders is formed. We expand it lexicographically into a

linear order and hence form an ordering of the all vertices of G. It can be verified that this order gives us an *st* numbering on the vertices of G such that the vertices that belong to the same cluster are numbered consecutively.

With this *c-st* numbering, we transform a clustered graph into hierarchical graph by assigning the layer of each vertex with its *c-st* number. Then, apply the straight-line hierarchical drawing algorithm described in section 3, hence, obtain a planar straight-line hierarchical drawing of G. The *c-st* numbering ensures that each cluster occupies consecutive layers in the drawing. For every cluster, we draw a convex hull of the vertices of the cluster. In this drawing, there are no edge crossings; there are no edges that cross the region (the convex hull) of a cluster where they do not belong. Since we are given a connected clustered graph, each cluster forms a connected subgraph of G. If the drawing of an edge crosses the convex hull of a cluster where it does not belong, then there would be an edge crossing. This forms a contradiction. Note that if we draw regions as rectangles instead of convex hulls, edge-region crossings are still possible.

Since an *st* numbering can be constructed in linear time [6], the computation of *c-st* numbering takes linear time in terms of the size of the graph. An algorithm to compute a convex hull of a set of m points requires $O(m \log m)$ time [15]. By Theorem 5, our method takes $O(n^2)$ time.

The following theorem summarizes our result on planar straight-line convex cluster drawings.

Theorem 9. *Let $C = (G, T)$ be a c-planar clustered graph with n vertices. A planar straight-line convex cluster drawing of C can be constructed in $O(n^2)$ time.*

6 Conclusion and Remarks

In this paper, we answer one of the basic questions for hierarchical graphs that has not be investigated before. We show that every hierarchical planar graph admits a planar straight-line hierarchical drawing, and present an algorithm that produces such drawings in $O(n^2)$ time. With this result, we answer a similar basic question for clustered graphs that has been posed as an open problem in [8]. We show that every c-planar clustered graph admits a planar straight-line convex cluster drawing. A method to construct such drawings is provided.

The algorithms that we present in this paper take quadratic time. From the computational point of view, it is interesting to know whether the time complexity can be improved, say, to $O(n \log n)$; and whether $O(n \log n)$ is optimal for this type of problems.

The drawings produced by our algorithms may require exponential area. This is justified by the area lower bounds for these drawing conventions presented in [14, 8]. Relaxing the straight-line constraints can give us polynomial area bounds [4].

For future work on clustered graphs, other drawing conventions such as poly-line rectangular cluster drawings, and also nonplanar drawings will be studied.

For clustered graphs, we only ensure that clusters are drawn as convex polygons, while it is desirable to represent clusters as more regular bodies such as circles and rectangles. This also forms an interesting topic for our future research.

References

1. G. Di Battista and R. Tamassia. Algorithms for plane representations of acyclic digraphs. *Theoretical Computer Science*, 61:175–198, 1988.
2. J.A. Bondy and U.S.R. Murty. *Graph Theory with Applications.* North-Holland, New York, N.Y., 1976.
3. P. Eades and K. Sugiyama. How to draw a directed graph. *Journal of Information Processing*, 424–437, 1991.
4. Peter Eades and Qing-Wen Feng. Orthogonal grid drawing of clustered graphs. Technical Report 96-04, Department of Computer Science, The University of Newcastle, Australia, 1996.
5. Peter D. Eades, Xuemin Lin, and Roberto Tamassia. An algorithm for drawing a hierarchical graph. *International Journal of Computational Geometry and Applications*, 1995.
6. S. Even and R. E. Tarjan. Computing an st-numbering. *Theoretical Computer Science*, 2:339–344, 1976.
7. I. Fary. On straight lines representation of planar graphs. *Acta Sci. Math. Szeged.*, 11:229–233, 1948.
8. Qing-Wen Feng, Robert F. Cohen, and Peter Eades. How to draw a planar clustered graph. In *COCOON'95*, volume 959 of *Lecture Notes in Computer Science*, pages 21–31. Springer-Verlag, 1995.
9. Qing-Wen Feng, Robert F. Cohen, and Peter Eades. Planarity for clustered graphs. In *ESA'95*, volume 979 of *Lecture Notes in Computer Science*, pages 213–226. Springer-Verlag, 1995.
10. E.R. Gansner, S.C. North, and K.P. Vo. Dag – a program that draws directed graphs. *Software – Practice and Experience*, 18(11):1047–1062, 1988.
11. D. Harel. On visual formalisms. *Communications of the ACM*, 31(5):514–530, 1988.
12. Wei Lai. *Building Interactive Digram Applications.* PhD thesis, Department of Computer Science, University of Newcastle, Callaghan, New South Wales, Australia, 2308, June 1993.
13. Thomas Lengauer. Hierarchical planarity testing algorithms. *Journal of ACM*, 36:474–509, 1989.
14. Xuemin Lin. *Analysis of Algorithms for Drawing Graphs.* PhD thesis, Department of Computer Science, University of Queensland, Australia, 1992.
15. Franco P. Preparata and Michael I. Shamos. *Computational geometry: an introduction.* Springer-Verlag, New York, 1985.
16. R. Read. Methods for computer display and manipulation of graphs and the corresponding algorithms. Technical Report 86-12, Faculty of Mathematics, Univ. of Waterloo, July 1986.
17. S.K. Stein. Convex maps. *Proceedings American Mathematical Society*, 2:464–466, 1951.
18. 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.

19. K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical systems. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-11(2):109–125, 1981.
20. W.T. Tutte. How to draw a graph. *Proceedings London Mathematical Society*, 3(13):743–768, 1963.
21. K. Wagner. Bemerkungen zum vierfarbenproblem. *Jber. Deutsch. Math.-Verein*, 46:26–32, 1936.