IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 9, Issue 4 (Mar. - Apr. 2013), PP 92-102
www.iosrjournals.org

Survey on Planar Drawing of Clustered Graphs

Marzia Sultana
Department of Computer Science and Engineering, Military Institute of Science and Technology, Dhaka, Bangladesh

Abstract: *In recent years, there has been great interest in graph drawing to express relational information and applications range from CASE tools, reverse engineering, idea organization, and software design. However, as the graphs to be visualized become more and more complex, extended structured graph variations are needed. Clustered graphs are graphs with recursive clustering structures. The possible utility of clustered graphs include CASE tools, reverse engineering, idea organization, software design, knowledge representation, visualization of social networks and biological networks , VLSI design tools and divide and conquer approaches. Thus, clustered graphs are useful tools worth investigating.*

Keywords *– Cluster Graphs, C-Planarity, Planar Drawing, St-Numbering.*

I. INTRODUCTION

Graph drawing algorithms are a basic enabling technology which is used to help with the under-standing of large sets of inter-related data. By presenting data in a visual form it can often be easily digested by the user and both regular patterns and anomalies can be more easily identified. However most data sets do not contain any explicit information on how they should be laid out for easy viewing, although normally such a layout will depend on the relationships between pieces of data. Thus if we model the data points with the vertices of a graph and the relationships with the edges we can use graph drawing algorithms to infer a good layout from an arbitrary data set based on the relationships.

Clustered graphs are new graph formalism based on a recursive grouping structure over the set of vertices. Ways to draw these graphs, in which all the vertices of a cluster are contained in a simple region, are explored, motivated by the need to visualize more and more complete relational information. A stronger notion of planarity in the context of clustered graphs and an $O(n^2)$ algorithm to test (and embed) this version of planarity is given. Straight line convex drawings are produced by two algorithms, a general $O(n^2 \log n)$ algorithm based on a straight line drawing algorithm for hierarchical graphs and an $O(n^{2.5})$ algorithm based on Tutte's algorithm for clustered graphs meeting some connectivity conditions. As well, orthogonal grid rectangular cluster drawings which are computed to the optimal parameters of $O(n^2)$ area and at most 3 bends per edge are discussed. Multilevel representations are also investigated [1].

II. PRELIMINARIES

2.1 *Some important definition*

A graph G = (V, E) consists of a set V of vertices and a set E of edges, that is, pairs of vertices. Graphs are commonly used to model relations in computing, and many systems for manipulating graphs have recently been developed. A graph drawing algorithm reads as input a combinatorial description of a graph, and produces as output a visual representation of the graph. Such algorithms aim to produce drawings which are easy to read and easy to remember. Many graph drawing algorithms have been designed, analyzed, tested and used in visualization systems.

A vertex in a connected graph is called a cut vertex if its removal from G results in a disconnected graph. A connected graph is called biconnected if it is simple and has no cut vertex. Similarly, a pair of vertices in a connected graph is called a cut pair (or separation pair) if its removal from G results in a disconnected graph. A connected graph is called triconnected if it is simple and has no cut pair. We say that a cut pair (u, v) separates two vertices s and t if s and t belong to different components in G - (u, v).

The planarity property has been the object of much of Graph Theory. A graph G = (V, E) is called planar if its vertices and edges are drawn as points and curves in the plane so that no two curves intersect except at their endpoints, where no two vertices are drawn at the same point. 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. In such a drawing, the plane is divided into several connected regions, each of which is called a face. A face is characterized by the cycle of G that surrounds the region. Such a cycle is called a facial cycle. A set F of facial cycles in a drawing is called an embedding of a planar graph G. A plane graph G = (V, E, F) is a planar graph G = (V, E) with a fixed embedding F of G, where we always denote the outer facial cycle in F by $f_o \in F$.

A vertex (respectively, an edge) in f_o is called an outer vertex (respectively, an outer edge), while a vertex (respectively, an edge) not in f_o is called an inner vertex (respectively, an inner edge). A path Q between two vertices s and t in G is called inner if every vertex in V (Q) - (s, t) is an inner vertex.

A biconnected plane graph G is called internally triconnected if, for any cut pair (u, v), u and v are outer vertices and each component in G(u, v) contains an outer vertex. Note that every inner vertex in an internally triconnected plane graph must be of degree at least 3. For a cut pair (u, v) of an internally triconnected plane graph G = (V, E, F), if u and v are not adjacent and there is an inner facial cycle f ϵ F such that (u, v) ∈ V (f), we say that f separates two vertices s and t if the cut pair (u, v) separates them.

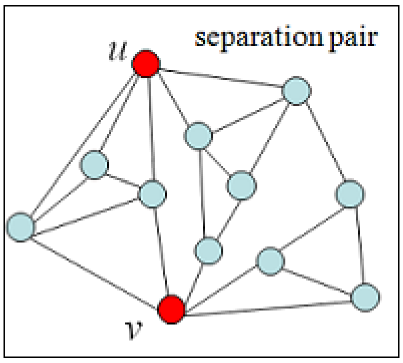

Fig 1: Separation pair (uv)

A clustered graph C = (G, T) consists of an undirected graph G = (V, E) and a rooted tree T = (V, A), such that the leaves of T are exactly the vertices of G. We call G the underlying graph and T the inclusion tree of C. To distinguish vertices in T from those in G, vertices in T are called nodes. Each node v of T represents a cluster V (v), a subset of the vertices of G that are leaves of the subtree rooted at v. Then for the root of T, V (v) = V. A node v in T (or its cluster V (v)) is called nontrivial if it is neither the root nor a leaf of T. Let G(v) denote the subgraph G[V - (v)] of G induced by V (v). Note that the tree T represents a laminar family of subsets of the vertices in G. If a node v' is a descendant of a node v in the tree T, then we say that the cluster of v' is a sub-cluster of v.

A clustered graph C = (G, T) is a connected clustered graph if each cluster V (v) induces a connected subgraph G(v) of G. A clustered graph C = (G, T) is completely connected if, for every non-root node v of T, both subgraphs G(v) and G[V - V (v)] are connected [2]. Note that G is biconnected if C = (G, T) is completely connected, since G[V - (v)] is connected for every leaf cluster (v) in T.

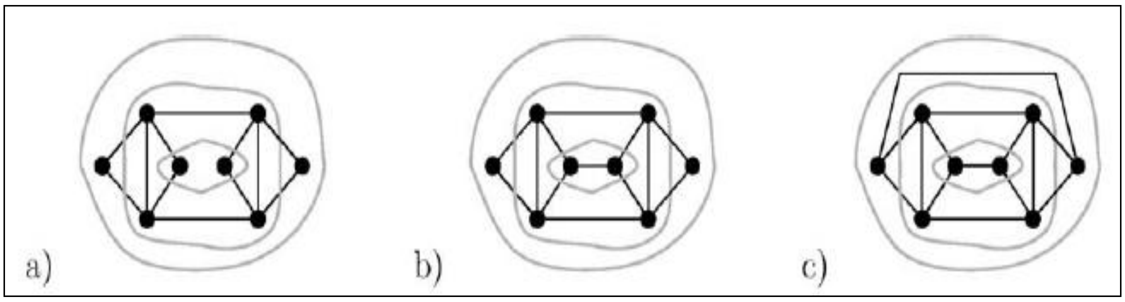

Fig 2: (a) A non-connected (b) A connected but not completely connected and (c) A completely connected cluster graph

2.2 *Clustered graph drawing*

In a drawing of a clustered graph C = (G, T), graph G is drawn as points and curves as usual. For each node v of T, the cluster is drawn as a simple closed region R(v) enclosed by a simple closed curve such that:

- The drawing of G(v) is completely contained in the interior of R(v),
- The regions for all sub-clusters of v are completely contained in the interior of R(v), and
- The regions for all other clusters are completely contained in the exterior of R(v).

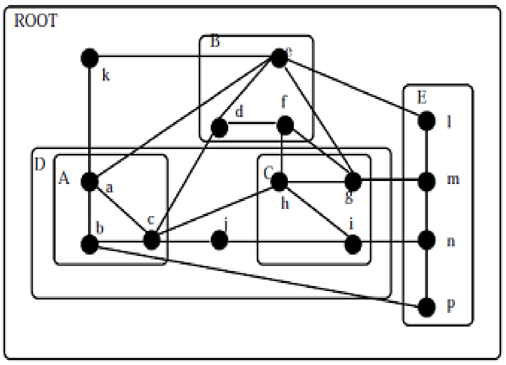

Fig 3: An example of cluster graph

The drawing of edge e and region R has an edge-region crossing if the drawing of e crosses the boundary of R more than once. A drawing of a clustered graph is compound planar (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. The characterization of c-planar clustered graphs is known only for connected clustered graphs. Figure 4 and 5 show examples of c-planar drawings of clustered graphs.

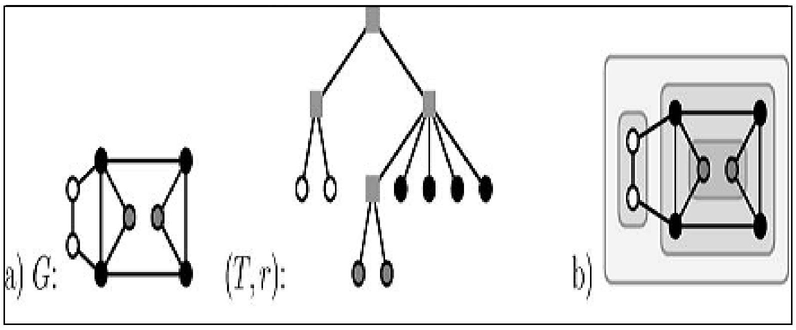

Fig 4: A c-planar cluster graph and its c-planar drawing

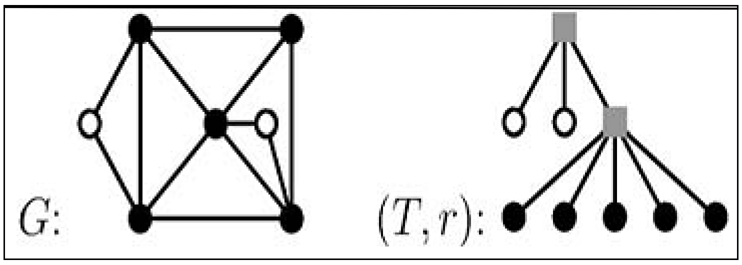

Fig 5: A connected cluster graph which is not c-planar

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 of G, such that for each node v of T, all the vertices and edges of G [V - G(v)] are in the external face of the drawing of G(v) [3].
Let $C_1 = (G_1,T_1)$ and $C_2 = (G_2,T_2)$ be two clustered graphs such that T_1 is a subtree of T_2 and for each node v of T_1, $G_1(v)$ is a subgraph of $G_2(v)$. We say that C_1 is a sub-clustered-graph of C_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 [2].

2.3 *The c-st numbering*

In this section, we define the concept of "c-st numbering" and show how to compute it in linear time.
By Theorem 2, we assume that we are given a c-planar connected clustered graph C = (G, T) with a c-planar embedding. For each vertex u, let A(u) be a doubly-linked list of edges around u, where the edges in A(u) appear along u in the order of the list in the embedding. The st numbering of the vertices of a graph has proved to be a useful tool for many graph algorithms, especially graph drawing algorithms. We next review this concept. Suppose that (s, t) is an edge of a biconnected graph G with n vertices. In a st numbering, 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 vertex and a higher-numbered vertex. Vertices s and t are called the source and the sink respectively. An st numbering of a biconnected graph can be computed in linear time [4].

They used the c-st numbering of vertices in G, which is an extension of st numberings. Suppose that (s, t) is an edge of a biconnected graph G with n vertices. In st numbering, 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 vertex and a higher-numbered vertex. A c-st numbering for a clustered graph C = (G, T) is an st numbering of the vertices in G such that the vertices that belong to the same cluster are numbered consecutively. This numbering gives us a layer assignment of the vertices of G. Hence, a c-planar clustered graph C is transformed to a hierarchical-st plane graph H, and each cluster has consecutive layers. Based on this property, Eades et al. [5] proved that a planar straight-line convex cluster drawing can be constructed from the straight-line hierarchical drawing. In this method, a given underlying graph is augmented to a triangulated graph to ensure the existence of c-st numberings in the clustered graph, and then all added edges are removed from a drawing of the triangulated graph to obtain a desired straight-line drawing of C.

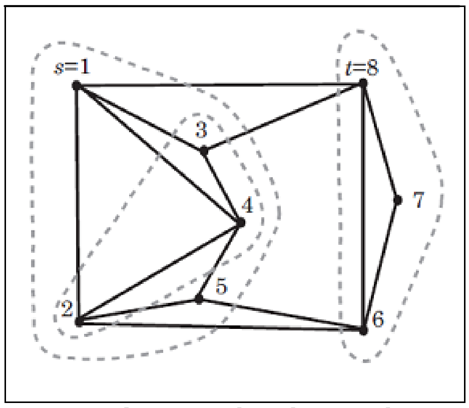

Fig 6: A cluster graph with st-numbering

III. DESCRIPTION OF SOME PLANAR DRAWINGS

3.1 *Straight-line drawing*

One of the basic graph drawing conventions consists of representing edges as straight-line segments. The straight-line drawing convention is widely used in visualization. Consequently, the straight-line drawing convention is one of many important modern aesthetic criteria [6] in graph drawing. More importantly, there are several general methods for drawing graphs which begin by adding dummy vertices on edges and then apply a straight-line drawing algorithm. This demand has spawned a considerable amount of attention to straight-line drawings in the research community. Tutte [7] proved that every triconnected planar graph admits a planar straight-line drawing where all the face boundaries are drawn as convex polygons. Algorithms for such drawings have also been investigated by Chiba et al. [8, 9]. More recently, e_cient algorithms for planar straight-line drawings were developed by de Fraysseix, Schnyder, Chrobak and Kant. These recent methods show that every planar graph admits a straight-line drawing in which each vertex is located at an integer grid point and the whole drawing uses O(n^2) area.

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? Answer of this question is based the results for hierarchical graphs, transform a clustered graph into a hierarchical graph, and construct a straight-line convex cluster drawing on top of the straight-line hierarchical drawing[5].

An outline of our algorithm is described as follows. We need to generalize this notion to clustered graphs. Given a clustered graph C = (G, T), a st-numbering of the vertices of G such that the vertices that belong to the same cluster are numbered consecutively is a c-st numbering. the c-st numbering gives us a layer assignment of the vertices of G. The algorithm for constructing the c-st numbering runs in linear time. Using this ordering Hence, the clustered graph is transformed to a hierarchical graph, and each cluster has consecutive layers. Because of this property, a straight-line convex cluster drawing can be constructed from the straight-line hierarchical drawing. The time complexity of the algorithm is linear in the output size.

Lemma A c-st numbering of a triangulated and c-planar connected clustered graph C = (G, T) can be computed in O(n) time.

Straight-Line drawing algorithm Using the c-st numbering computed in the previous section, we transform a clustered graph into a hierarchical graph by assigning the layer of each vertex with its c-st number. Then apply the straight-line hierarchical drawing algorithm, and 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. Clearly, the convex hull of a cluster contains the convex hulls of its subclusters. The c-st numbers within a cluster are consecutive, thus if the convex hulls of two clusters overlap in y-coordinate, then one is a subcluster of the other. This keeps the clusters apart; for example, if vertex u lies inside the convex hull for cluster v, then u is a member of V(v). In general, if a cluster v is not a sub-cluster of v' and neither is cluster v' a sub-cluster of v, then the convex hulls of v and v' are disjoint.

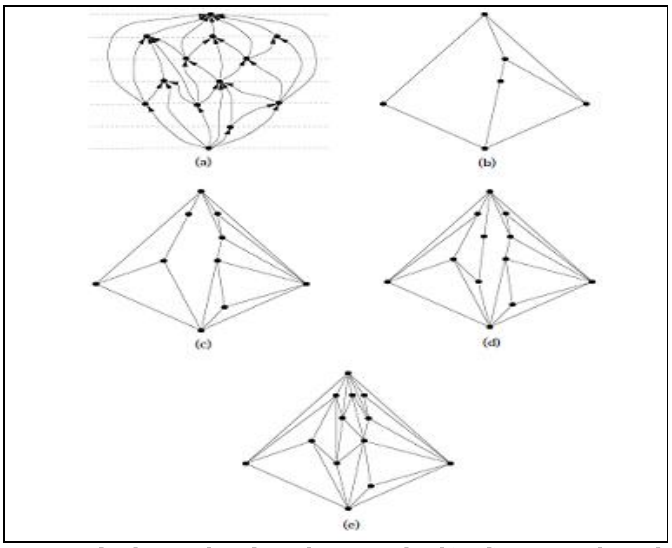

Fig 7: An example: (a) A triangular hierarchical-st plane graph, (b)-(d) intermediate drawings produced by the procedure,(e) the final drawing.

There are no edge crossings in the drawing, since our hierarchical planar drawing algorithm does not produce any edge crossings. Since we are given a connected clustered graph, each cluster forms a connected subgraph of G. If l is a straight-line segment with endpoints outside the convex hull for cluster v, and l intersects the convex hull for v, then l crosses an edge in G(v). Therefore there are no edges that cross the region (the convex hull) of a cluster where they do not belong, because otherwise there would be an edge crossing. A convex hull of a given simple polygon with m apexes can be constructed in O(m) time [10]. In fact, there is a simple O(m) time algorithm for computing a convex hull of a set of m points which are already sorted by their y-coordinates. Since all vertices in each cluster V (v) have consecutive st numbers (hence y-coordinates), a convex hull of V (v) can be computed in O(V(v)) = O(n) time. Then the total time of computing all convex hulls in C = (G, T) becomes O(n^2). This complexity is slightly reduced as follows. Let CH(v) denote the set of vertices which are on the convex hull of a cluster V(v) (hence, |CH(v)|) is the output size of the convex hull). To compute CH(v) of a cluster V (v), we can discard all vertices that are properly contained inside the convex hull CH(v) for some child cluster V (v) $\in$ chl(v). Before computing the convex hull CH(v) for a node v $\in$ V , we compute all convex hulls CH(v) for the children chl(v). Therefore, by computing convex hulls from the bottom of the tree T to the root, we can obtain all CH(v), v $\in$ V in O(n + $\sum_{v\epsilon V}$ |CH(v)|) time, which is linear in terms of the output size of a straight line convex cluster drawing of C.

Computing a straight-line drawing of the hierarchical graph with n vertices can be done in linear time. A c-st numbering of a clustered graph with n vertices can be computed in O(n) time. In summary, we establish the following result on planar straight-line convex cluster drawings.

Theorem 3 Let C = (G, T) be a c-planar clustered graph with n vertices. A planar straight-line convex cluster drawing of C in which each cluster is a convex hull of points in the cluster can be constructed in O(n + D) time, where D = O(n^2) is the total size of convex polygons for clusters in the drawing.

3.2 *Convex drawing*

In this section, the convex drawing of clustered planar graphs is described. A c-planar drawing of a clustered graph is called a planar straight-line convex cluster drawing if edges are drawn as straight-line segments and clusters are drawn as convex polygons. Note that not all planar graphs admit a convex drawing. Tutte [7] gave a necessary and suficient condition for a triconnected plane graph to admit a convex drawing. He also showed that every triconnected plane graph with a given boundary drawn as a convex polygon admits a convex drawing using the polygonal boundary. That is, when the vertices on the boundary are placed on a convex polygon, inner vertices can be placed on suitable positions so that each inner facial cycle forms a convex polygon. The drawings in Fig. 8(a) and (b) are planar straight-line convex and non-convex cluster drawing. It is proved that every internally triconnected hierarchical plane graph with the outer facial cycle drawn as a convex polygon admits a convex drawing. The algorithm which constructs such a drawing extends the previous known result that every hierarchical planar graph admits a straight-line drawing.

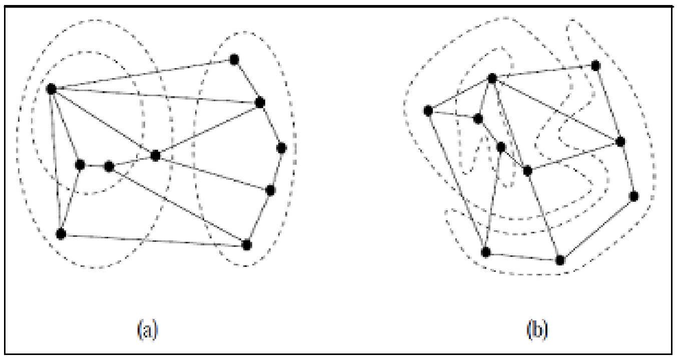

Fig 8: (a) Convex and (b) non-convex drawing of same cluster graph.

Theorem 4 Let C = (G, T) be a c-planar clustered graph with n vertices. A planar straight-line convex cluster drawing of C in which each cluster is a convex hull of points in the cluster can be constructed in $O(n^2)$ time. A fully convex drawing as a planar straight-line convex cluster drawing such that, facial cycles are also drawn as convex polygons. Among the four c-planar drawings in Fig. 8 and Fig. 9, only the drawing in Fig. 9(b) is fully convex. In what follows, only c-planar and connected clustered graphs are considered, and present a characterization of these clustered graphs that have fully convex drawings.

A region-face crossing as follows. In a c-planar drawing of a clustered graph C= (G, T), the region R of a cluster and a cycle f of G cross each other if the boundary of R cross the drawing of more than two edges in f.

Theorem 5 Let C = (G, T) be a c-planar and connected clustered graph that has a c-planar drawing such that the outer facial cycle does not cross any cluster region, and G be internally triconnected. Then,

(i) There exists a fully convex drawing of C if and only if C is completely connected.

(ii) A fully convex drawing of C (if any) can be constructed from drawing in $O(n^2)$ time, where each cluster is a convex hull of points in the cluster and n is the number of vertices in G.

Necessity of Theorem 5(i) The necessity in Theorem 12(i) is proved via several lemmas.

Lemma If a c-planar and connected clustered graph C admits a planar straight-line convex cluster drawing such that the drawing of G is inner-convex, then there is no region-face crossing between regions for clusters and inner facial cycles.

Proof: We see that if there is a region-face crossing between the region R(v) of a cluster v and an inner facial cycle f in G, then it is impossible to draw both R(v) and f as convex polygons in one drawing. Therefore, if C admits a fully convex drawing, then there is no region-face crossing between regions for clusters and inner facial cycles.

Lemma Suppose that a connected clustered graph C = (G, T) has a c-planar drawing which has no region-face crossing between regions for clusters and inner facial cycles. Then for every non-root node v of T, every component of the subgraph G[V - V (v)] contains an outer vertex.

Proof: Let ψ be such a c-planar drawing of C. In ψ , G is a plane graph (V, E, F), and we denote the outer facial cycle of G by f_o. Let v be a non-root node of T. To derive a contradiction, assume that G[V - V (v)] has a component G_A that contains no vertex in fo. In ψ , G_A is a plane graph, and we denote its boundary by B_A and the set of inner faces of G_A by F_A. Since R(v) is a simple closed region, G_A is not completely surrounded by R(v). Consider a facial cycle f $\in$ F - F_A that shares a vertex with B_A. Since G_A is an inclusionwise maximal component in G[V - V(v)], a vertex in f is contained in R(v). By assumption, there is no region-face crossing in ψ, and hence f consists of two subpaths f ' and f", one is the intersection with B_1 and the other with R(v). We now consider all such facial cycle f_1, f_2, . . . , fp $\in$ F - F_A that share vertices with B_A. Then we see that the subpaths f''_1 , f''_2 , . . . , f''_p of them form a closed curve, which implies that G_A is enclosed by R(v), a contradiction. This proves the lemma.

Lemma Suppose that a connected clustered graph C = (G, T) has a c-planar drawing which has no region-face crossing. Then C is completely connected.
Proof: Let ψ be such a c-planar drawing of C. In ψ, G is a plane graph (V, E, F), and we denote the outer facial cycle of G by f_o. Since C is a connected clustered graph, it suffices to show that, for every non-root node v of T, subgraph G[V - V (v)] is connected.
Let v be a non-root node of T. To derive a contradiction, assume that G[V - V (v)] is not connected, and G_A and G_B be two components in G[V - V (v)]. Both G_A and G_B contain outer vertices in f_o. This, however, contradicts that region R(v) and f_o has no region-face crossing. We are ready to prove the necessity in Theorem 5(i).
Let ψ be a fully convex drawing of C. Clearly ψ is a convex drawing on G, and G must be internally triconnected. We show that C is completely connected. There is no region-face crossing between regions for clusters and inner facial cycles. By assumption on C, there is no region-face crossing between regions for clusters and the outer facial cycle, either. C is completely connected. This proves the necessity in Theorem 5(i).

Sufficiency of Theorem 5(i) To prove the sufficiency of Theorem 5(i), we follow an approach due to Eades et al. [5] which was used to derive Theorem 11. They used the c-st numbering of vertices in G. Based on this property, Eades et al. [5] proved that a planar straight-line convex cluster drawing can be constructed from the straight-line hierarchical drawing. In this method, a given underlying graph is augmented to a triangulated graph to ensure the existence of c-st numberings in the clustered graph, and then all added edges are removed from a drawing of the triangulated graph to obtain a desired straight-line drawing of C. However, the resulting drawing may not be convex. Since we aim to construct a convex drawing of G, we cannot use triangulation to find c-st numberings. Fortunately, complete connectedness ensures the existence of c-st numberings instead.

Lemma Suppose that C = (G, T) is a c-planar and completely connected clustered graph. Then C has a c-st numbering such that s and t can be chosen as adjacent vertices in the outer facial cycle, and such a c-st numbering can be computed in linear time.
Proof: We give only a sketch. It is known [5] that the lemma holds if, in addition, G is triangulated, i.e., all facial cycles are triangles (note that every connected clustered graph C is completely connected if its underlying graph is triangulated). The correctness of the proof in [5] relies on only complete connectedness of C = (G, T), not on triangulation.
Hence the argument can be carried over to the case of completely connected clustered graphs, and the lemma holds.

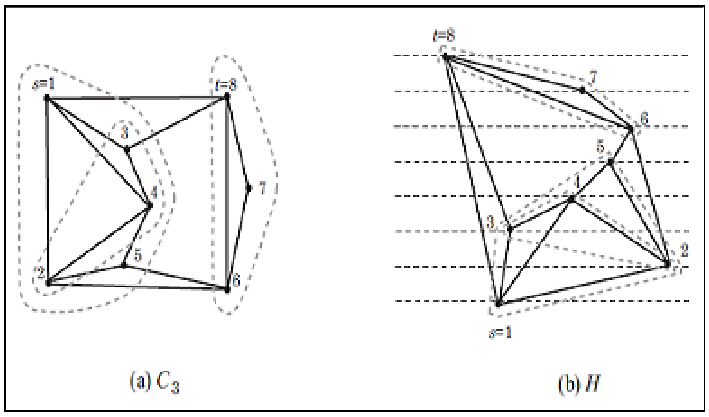

Fig 9: (a) A c-planar and completely connected cluster graph with st-numbering, (b) A convex drawing of the graph in (a).

The suficiency in Theorem 5(i) and Theorem 5(ii) Let C = (G, T) be a c-planar and completely connected clustered graph, and be a c-planar drawing such that the outer facial cycle f_o does not cross any cluster region.

First, we compute a c-st numbering of C = (G, T) such that s and t are chosen as adjacent vertices in the outer facial cycle f_o. This can be done in linear time. For example, Figure 9(a) illustrates a c-planar and completely connected clustered graph C_3 and its c-st numbering.

Next, we regard the plane graph G as a hierarchical-st plane graph by assigning the layer of each vertex with its c-st number. Figure 9(b) illustrates the hierarchical-st plane graph H obtained from clustered graph C_3 by its c-st numbering.

Then, we compute a convex drawing ψ* of H. Such a drawing ψ* can be obtained in $O(n^2)$ time.

Finally, for each cluster V (v), we let the convex hull of the vertices in V (v) in ψ* be the region R(v) of the cluster. A convex hull of a given simple polygon with m apices can be constructed in O(m) time [10]. Then the total time of computing all convex hulls in C = (G, T) is $O(n^2)$.

Overall, the entire running time of the above algorithm is O(n^2).

To prove the sufficiency in Theorem 5(i) and Theorem 5(ii), we only need to prove that the resulting drawing (ψ*, R(v)|v ∈ V) is a fully convex drawing. By Theorem 8, ψ* is a convex drawing. We show that (ψ*, R(v)|v ∈ V) is c-planar. Since R(v) is a convex hull of the vertices in V (v) in ψ* and edges are drawn as line segments, it contains the drawing of G(v). The c-st numbers within a cluster are consecutive, thus if the convex hulls of two clusters overlap in y-coordinate, then one is a sub-cluster of the other. This keeps the disjoint clusters apart; For two clusters v and v' with V (v) ∩ V (v') ≠ ϕ the convex hulls of v and v' are disjoint.

We finally see that there are no edge crossings and no edge-region crossings. Since ψ* is a plane drawing of G, it has no edge crossings. Assume that the drawing of an edge e intersects region R(v) of a cluster v twice. This, however, contradicts that ψ* is a plane drawing of G, since G(v) is connected and the line-segment for e must create an edge crossing with some edge in G(v). Therefore, (ψ*, R(v)| v ∈ V) is a fully convex drawing of C.

This completes the proof of Theorem 5. From the argument for establishing Theorem 5, we easily derive the following corollary.

Corollary Let C = (G, T) be a c-planar and connected clustered graph, and G be internally triconnected. Then,

(i) There exists a planar straight-line convex cluster drawing of C such that the drawing of G is inner-convex if and only if, for every non-root node v of T, every component of the subgraph G[V - V (v)] contains an outer vertex.

(ii) Such a drawing of C in (i) (if any) can be constructed from drawing ψ in O(n^2) time, where each cluster is a convex hull of points in the cluster and n is the number of vertices in G.

Proof: (i) By Lemmas 13 and 14, we see the necessity of this corollary. To show the sufficiency, we augment the clustered graph C as follows. Let V_o be the set of outer vertices in G. We create a new vertex s* in the exterior of G, join G and s* with new edges (s*, v), v ∈V_o to obtain a new plane graph G*, where its boundary f^*_o is a triangle, and add new clusters v* = V ∪{s*}, V_v = {v}, v ∈V_o to T. We see that the resulting clustered graph C* = (G*, T*) remains c-planar and connected. Also we easily see that C* has no region-face crossing between regions for clusters and facial cycles. Hence by applying Theorem 12 to C*, there is a fully convex drawing ψ of C*. After removing the drawing of s* and edges (s*, v), v ∈ V_o from ψ, we obtain a desired drawing of C. (ii): Immediate from (i) and Theorem 5(ii) [3].

3.3 *Orthogonal Grid drawing*

An orthogonal rectangular drawing of a clustered graph, C =(G, T) is a drawing in which edges are drawn as horizontal and vertical segments, vertices are grid points and the region boundaries of clusters are rectangles. The basic idea of the algorithm to produce such drawings is given a clustered graph, C =(G, T) with a C-planar embedding where C has a subclustered graph C_0 such that every vertex of C_0 has a degree of no greater than 4 (a vertex of degree of more than 4 is replaced by a set of vertices each having degree 4 and a path connecting them), convert G into a planar st graph through the c-st numbering, in which vertices belonging to the same cluster are numbered consecutively, described in the previous section. From the planar st graph in which directions have been added though the c-st numbering, a constrained visibility representation having vertices of selected paths verticality aligned, is made which goes through an orthogonalization process and a bend reduction procedure [3, 5, 11].

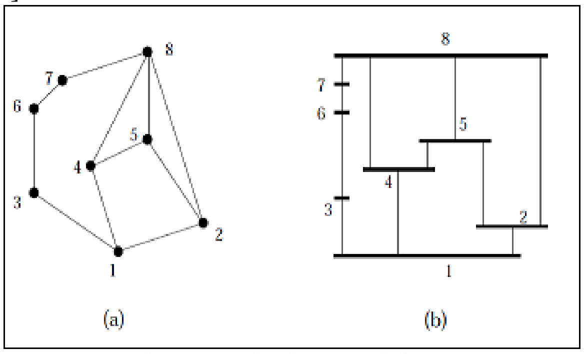

Fig 10: (a) A planar graph, (b) A visibility representation

A visibility representation, γ, for the planar st graph obtained from G has a horizontal line segment γ (v) for a vertex v and a vertical line segment γ(e) for an edge e such that the following conditions are met: whenever vertices u and v are distinct, segments γ(u) and γ(v) are disjoint and if e is the edge (u, v), the bottom segment that e intersects is γ(u) and the top segment that e intersects is γ(v) and e does not intersect anything else [9]. The constrained visibility algorithm takes a planar st graph and a set of nonintersecting all inclusive paths such that edges have the same x value (are vertically aligned) only if they are on the same path and paths are non-intersecting if the edges are disjoint and the edges do not cross at common vertices. Each cluster v, is assigned a vertical range and to make the bounding rectangle for v, four dummy vertices, b(v) (bottom), t(v) (top), r(v) (right), and l(v) (left) one for each side of the rectangle are added. As well, two paths for the vertical sides, (b(v), r(v), t(v)) and (b(v), l(v), t(v)) are added and all vertices entering are bunched in b(v) and those vertices leaving are bunched in t(v); all intercluster edges become paths, the vertices on the path representing the region boundaries crossed. It is clear that the resulting graph F is a planar st graph so the constrained visibility algorithm can be used. The two added paths for each cluster, every edge on an intercluster path and paths composed of a single edge are required to be aligned, in addition to having certain edges around a vertex aligned (say if a vertex has two incoming and two outgoing vertices, the right incoming edge is aligned with the left outgoing vertex) in order to avoid extra bends in the orthogonalization stage. Note that only the edges in C' are considered for alignment and although these paths may share common vertices, they are nonintersecting. If C has O(n) edges (it is planar), F, i. e. G with the dummy vertices and edges added recursively through T, has O(n^2) edges and the constrained visibility algorithm takes O(n^2) time and produces a visibility drawing on an O(n) X O(n) grid [9]. In the orthogonalization phase, the extra edges that are in C but not in C' are removed and some local operations are performed to transform horizontal segments to a point. Bends can only occur near endpoints. In the resulting orthogonal grid rectangular clustered (OGRC) drawing, every edge has at most 3 bends except for an edge connecting a source of degree 4 and a sink of degree 4 which has 4 bends, two near each endpoint. Notice that for a source or sink of degree 4 either the leftmost edge or the rightmost edge has two bends not both; this edge is called the extreme edge. For an edge with two bends near each endpoint, (u, v) fix one end u and swing all the edges around the other end v so that e bends only once near v. After this, e is called processed and u and v are called visited. But this operation creates another bend around the other extreme edge e'= (v, w). Now if w is a source or sink of degree 4 and e' is an extreme edge of w. The operation is repeated until an edge, e'= (v, w), is come across such that e' is not an extreme edge and w is visited. In both cases, e' with have at most three edges. The process is repeated until there are no edges with four bends. Bend reduction takes O(n^2) time. Thus, the OGRC drawing algorithm takes O(n^2) time and the resulting drawing has an area of O(n^2) with edges having at most three bends. There is a class of clustered graphs in which O(n) edges require more than two bends so in a sense the algorithm described is optimal. It is instructive to note that by lifting the straight line requirement, the area requirement of clustered graphs is reduced from O(2n) to O(n^2), from exponential time to polynomial time [5]. It is also profitable to note that the commonly used technique of using a visibility representation to obtain an orthogonal drawing works for clustered graphs as well [9].

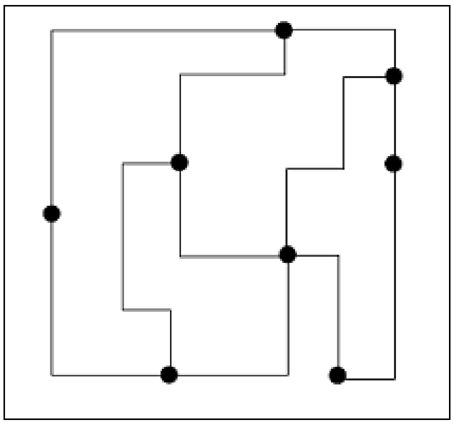
Fig 11: An orthogonal grid drawing

3.4 *Multilevel Visualization drawing*

When clustered graphs are drawn in the typical two-dimensional way, clusters are shown as recursive nested regions or included regions. However, when the graphs become too large or complex, two dimensional drawings may appear to be insufficient. A good way to set up a multilevel drawing of a clustered graph is to have a three dimensional representation of the inclusion tree and to put each two dimensional drawing of abstraction level at a different z coordinate so that different abstraction levels can be differentiated, thus preserving the mental map. Naive but useful methods for multilevel drawings of clustered graphs come easily after some further definitions [11, 12].

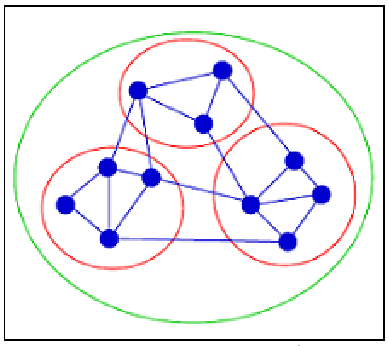
Fig 12: Two dimensional view of cluster graph

The height of a cluster is the depth of in T (the counting is actually backwards so that the leaves are at level 0). For a clustered graph, C = (G, T), the view at level i is a graph $G_i = (V_i, E_i)$ such that V_i are the nodes at level i of T (if i is not 0, there are clusters) and an edge (u, v) $\in E_i$ if $\exists$(u; v) $\in$ E such that u is in cluster and v is in cluster ; such an edge may correspond to several edges in G. A planar drawing of a view involves drawing each node in V_i as a region and edges as curves between the region boundaries of the endpoints; such a view is C-planar, if there are no edge crossings or edge region crossings. A multilevel drawing of clustered graph, C =(G, T), consists of a sequence of views of each level of the tree T with the two dimensional drawing of level i placed at z=i and a three dimensional representation of the tree T such that a node at level i is drawn as a point at z=i and within the region of is the drawing of the view at that level; a multilevel drawing is C-planar if the plane drawing of all views at all levels are C-planar; if a two-dimensional plane graph is C-planar, the derived view drawing is also C-planar. The derived view also preserves the region conventions of the two-dimensional plane drawing. To draw the inclusion tree, the position of each node must be determined and then the edges are routed around these node positions by drawing straight line segments between the nodes. Since the edges of the tree cannot span more than one level, they cannot cross. The position of each node is found recursively from the tree T from bottom to top. For i=1 to the full depth of the tree,

(i) If i=0, each leaf node is positioned where it is positioned on the two-dimensional plane drawing (the view at level 0 is actually G).

(ii) For every node at level i> 0, take the average xy coordinates of the children of as the xy coordinate of . Thus, is placed above most of its children, so the angle to the xy plane is quite large (if there is one child, the line is vertical) [12].

Using this method, every node is positioned within its corresponding region in the view. Note that although [11] mentions that it describes the multilevel algorithm in terms of the two-dimensional drawing algorithms; the actual algorithm is indifferent to the particular two dimensional drawing algorithm.

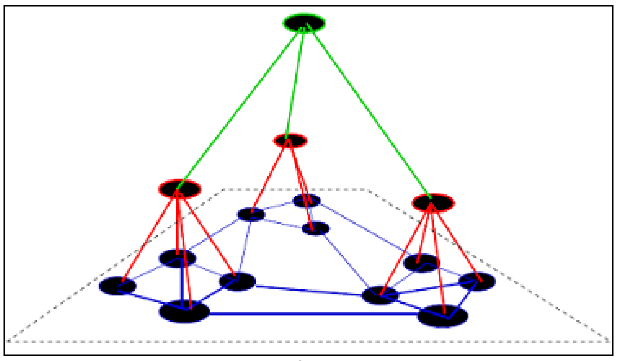
Fig 13: Representation of cluster in three-dimension

IV. CONCLUSION AND FUTURE WORK

The possible utility of clustered graphs include CASE tools, reverse engineering, idea organization, software design, knowledge representation, visualization of social networks and biological networks, VLSI design tools and divide and conquer approaches[3, 5, 6, 7, 8, 9, 10]. All in all, a rather wide variety of drawing algorithms for a type of structured graph extension, clustered graphs, with diverse applications has been described from two straight line convex drawing algorithms, one involving hierarchical graphs and one based on Tuttes algorithm to visibility representations to orthogonal grid rectangular cluster drawings to even multilevel representations. Each of these algorithms assumes a C-planar embedding for the clustered graph to be drawn; C-planarity testing and embedding algorithms have been discussed as well. Yet, there are still many open problems to be solved as evidenced by the long list following.

One characterization of c-planarity for general clustered graph is not constructive. The c-planarity testing problem for non-connected clustered graph is still open. Adding more constraints such as angles of vertices to convex representations may be an interesting research direction in the future. For large sized graphs further research can be done to improve the performance time. For multilevel representation more experimental research is needed. In many applications, the relational structure to be visualized is dynamic and it is inefficient to apply a static drawing algorithm whenever an incremental change is made. Further research is needed in dynamic drawing algorithm for cluster graphs.

REFERENCES

[1] G. A. Grun, Cluster Graphs: A Survey of Drawing Algorithms, *School of computer science, Simon Fraser University*, September 1998.
[2] S. Cornelsen , D. Wagner, Completely connected clustered graphs, *Journal of Discrete Algorithms 4*, 2006.
[3] S-H. Hong, H. Nagamochi, *Convex Drawings of Hierarchical Planar Graphs and Clustered Planar Graphs*, Technical report 2007-004, January 28, 2007.
[4] S. Even and R. R. Tarjan, Computing an st-numbering, *Theoretical Computer Science, 2*, 1976, 339-344.
[5] P. Eades, Q. Feng, X. Lin and H. Nagamochi, Straight-line drawing algorithms for hierarchical graphs and clustered graphs, *Algorithmica, 44*, 2006, 1-32.
[6] W. T. Tutte, How to draw a graph, *Proceeding London mathematical society,* 1963.
[7] W. T. Tutte, Convex representations of graphs, *Proc. of London Math. Soc.* 10, no. 3, 1960, 304-320.
[8] Q. Feng, P. Eades, and R. F. Cohen, *Planar drawing of clustered graphs*, Technical report, Department of Computing Science and Software Engineering, University of Newcastle, 1995.
[9] P. Eades and Q. Feng, *Drawing clustered graphs on an orthogonal grid*, Technical report, Department of Computing Science and Software Engineering, University of Newcastle, 1997.
[10] R. S. Preparata, F. P. Preparata, M. I. Shamos, *Computational Geometry: An Introduction*, (Springer, 1993).
[11] Q. Feng, *Algorithms for Drawing Clustered Graphs*, PhD thesis, Department of Computer Science and Software Engineering, University of Newcastle, 1997.
[12] P. Eades and Q. Feng, *Multilevel visualization of clustered graphs.* Technical report, Department of Computing Science and Soft-ware Engineering, University of Newcastle, 1996.
[13] T. Nishizeki, M. S. Rahaman, *Planar Graph Drawing*, (World scientific, singapur, 2004).
[14] Q. Feng, P. Eades, and R. F. Cohen, *Clustered graphs and c-planarity*, Technical report, Department of Computing Science and Software Engineering, University of Newcastle, 1995.