Planarity-Preserving Clustering and Embedding for Large Planar Graphs[⋆]

Christian A. Duncan[⋆⋆], Michael T. Goodrich, and Stephen G. Kobourov

Center for Geometric Computing
The Johns Hopkins University
Baltimore, MD 21218

Abstract. In this paper we present a novel approach for cluster-based drawing of large planar graphs that maintains planarity. Our technique works for arbitrary planar graphs and produces a clustering which satisfies the conditions for compound-planarity (c-planarity). Using the clustering, we obtain a representation of the graph as a collection of $O(\log n)$ layers, where each succeeding layer represents the graph in an increasing level of detail. At the same time, the difference between two graphs on neighboring layers of the hierarchy is small, thus preserving the viewer's mental map. The overall running time of the algorithm is $O(n \log n)$, where n is the number of vertices of graph G.

1 Introduction

The problem of displaying large graphs often arises in the networking and telecommunications areas. While such application areas typically give rise to non-planar graphs, there are nevertheless several application areas that give rise to large graphs that are planar. Examples of such planar graph applications include computations arising in computational cartography and geographic information systems (GIS). In this paper we are therefore concerned with the visualization of large planar graphs.

There are several approaches to the visualization of planar graphs, each of which must address the fact that the resolution of most display technologies (and possibly even the human eye) simply cannot display more than a few million pixels. Moreover, no matter how many pixels a display technology has, these pixels must display not just the vertices of a graph of interest, but also, and more importantly, the edges connecting these vertices. Our approach is based on displaying the graph using a hierarchical clustering in which the graph is represented by a collection of layers, where each succeeding layer describes the graph in a decreasing level of detail. That is, together with G one gives a tree T such that the leaves of T coincide with the vertices of G, and each internal node v of T represents the *cluster* defined by the vertices of G associated with the descendent leaves of v in T. In this case G can be drawn in a "layered" manner,

[⋆] Partially supported by NSF grant CCR-9732300 and ARO grant DAAH04-96-1-0013.
[⋆⋆] Also at Max-Planck-Institut für Informatik.

where we draw each cluster on the same layer of T as a region of the plane and connect adjacent clusters by segments. We would like that each such layer be drawn planar, with no segments intersecting each other or intersecting the boundary of a non-incident cluster region. Thus, the general goal of clustered graph drawing is to preserve the global structure of a graph G by recursively clustering smaller subgraphs of G and drawing these subgraphs as single nodes or filled-in regions in a rendering of G. By grouping vertices together into clusters in this way one can recursively divide a given graph into layers of decreasing detail, which can then be viewed in a top-down fashion.

1.1 Prior Related Work on Clustered Graph Drawing

If clusters of a graph are given as input along with the graph itself, then several authors give various algorithms for displaying these clusters in two or three dimensions [2,3,5,8]. Still, as will often be the case, if clusters of a graph are not given *a priori*, then various heuristics can be applied for finding clusters using properties such as connectivity, cluster size, geometric proximity, or statistical variation [7,9,11]. If no clusters are given and no special properties are known in advance, Duncan *et al.* [1] show how to create a hierarchical decomposition and a 3-dimensional drawing for general graphs. However, for planar graphs, it is possible to introduce edge-region crossings, in which edges cross cluster regions they are not part of. Even with no edge-edge crossings, the edge-region crossings are a serious drawback to the readability of a drawing.

Eades *et al.* [3] describe a drawing algorithm that draws a planar graph G, assuming that the clusters of G preserve certain recursive conditions, which they collectively call the *c-planarity* conditions. They show that if G and its clusters satisfy the c-planarity conditions, then one can produce a drawing of G such that each layer of the cluster hierarchy is drawn planar, with each vertex drawn as a convex region and each edge drawn as a straight line segment. This approach allows the graph to be represented by a sequence of drawings of increasing detail. As illustrated by Eades and Feng [2], this hierarchical approach to drawing large graphs can be very effective. However, we are not aware of any previous work for deterministically producing a clustering of an arbitrary planar graph so as to satisfy all the c-planarity conditions.

1.2 Our Results

In this paper we provide an algorithm for constructing a clustering of any planar graph so as to satisfy the c-planarity conditions of Eades *et al.* [3]. Our algorithm runs in $O(n \log n)$ time, uses $O(n)$ space, and can be implemented using simple "off-the-shelf" data structures. We also show that the clustering tree T, defined by our algorithm, has the additional property that the number of clusters at layer i of T (i.e., the clusters associated with the nodes of T at height i) is a constant fraction fewer than the number of clusters at the next higher layer, $i + 1$. Thus, T has $O(\log n)$ height. This in turn implies faster drawing times when T is used in a clustered graph drawing algorithm.

This logarithmic height result also implies some nice properties of the clustered drawing itself. For example, had we instead produced a clustering tree T of depth $\Theta(n)$, which is possible if one uses a different clustering algorithm, then we would have a hierarchy that takes an extraordinarily long time to traverse for large planar graphs. At the same time, an $o(\log n)$ height for T would imply drastic changes between consecutive layers in the hierarchy.

In addition to this logarithmic height result, our algorithm produces a clustering such that the changes between the graphs in consecutive layers of the hierarchy T are "local." In order to preserve the viewer's mental map of the graph when moving from one layer to another, the changes in the graph should be minimal. Given the graph in layer i in T, to obtain the graph associated with the next higher layer $i+1$ in T, we need to group certain sets of vertices together and replace them by new vertices. In this paper, we consider only changes that affect pairs of vertices, so that the tree T is in fact a *binary* tree.

Thus we restrict our clustering operation so as to allow only the combining of two adjacent clusters, which is an operation typically referred to as an *edge contraction.* Through a sequence of graph contractions, we obtain the layer graphs $G_0, G_1, \ldots, G_k$, where $G_0 = G$ and G_k is a singleton graph. If the changes necessary to obtain layer $i+1$ from layer i are to be local, then the following three *locality conditions for edge contraction* must be met:

1. A vertex can participate in at most one edge contraction.
2. Changes in the drawing of the graph that result from the contraction of an edge (u, v) should only affect edges with endpoint u or v.
3. A contraction of edge (u, v) results in the creation of vertex w. The placement of w in the drawing should be "close" to the edge (u, v). Optimally, we would like that w lie along the line segment defined by (u, v).

We provide a clustering method that satisfies the above locality conditions. One of the main challenges in creating the layers in a cluster hierarchy of a planar graph is to define clusters and the drawing algorithm associated with G's clustering in such a way that no edge crossings are introduced in the drawing of each layer. We provide a drawing algorithm which makes use of our clustering method to produce a drawing that has neither edge-edge crossings nor edge-region crossings. In addition, we show that one can use our clustering as input to the clustered planar graph drawing algorithm of Eades *et al.* [3].

2 Hierarchical Embedding of Planar Graphs

Let us assume, without loss of generality, that all the graphs that we are dealing with are maximally planar. If a particular graph is not maximally planar then we can fully triangulate it. Let $G = (V, E)$ be a maximally planar graph, where $|V| = n$. $V(G)$ and $E(G)$ as usual refer to the set of G's vertices and edges, respectively and the degree of a node v in graph G is $d_G(v)$. Let $l_G(f_i)$ be the length of a face f_i in G, where by the length of the face we mean the number of vertices on that face. Further, Δuvw will refer to the triangle defined by vertices u, v, w and by the edges $(u, v), (v, w), (w, u)$.

Similar to [2] we define the *clustered graph* $C = (G, T)$ to be the graph G and a tree T such that the vertices of G coincide with the leaves of T. An internal node of T represents a cluster, which consists of all the vertices in its subtree. All the nodes of T at a given height i represent the clusters of that level. A *view at level* i, $G_i = (V(G_i), E(G_i))$, consists of the nodes of height i in T and a set of representative edges. The edge (u, v) is in $E(G_i)$ if there exists an edge between a and b in G, where a is in the subtree of u and b is in the subtree of v. Each node $u \in T$ has an associated region, corresponding to the partition given by T.

We create the graphs G_i in a bottom-up fashion, starting with $G_0 = G$ and going all the way up to G_k, where $k = \texttt{height}(T)$. We obtain G_{i+1} from G_i by contracting a carefully chosen set of edges of G_i in a certain order. The z-coordinate of a vertex $v \in V(G_i)$ is equal to i, that is, all the vertices in G_i are embedded in the plane given by $z = i$. The edges of T are defined by the edge contractions. More precisely, if $(u, v) \in E(G_i)$ is contracted to a vertex $w \in G_{i+1}$, then edges (w, u) and (w, v) are added to T.

The problem of embedding planar graphs with straight lines and no crossings is well studied [4,10,12]. Embedding clustered graphs without crossings poses additional difficulties. To embed the layers, we reverse the sequence of graph contractions: we start with embedding of G_k (which has only one vertex). To obtain an embedding for G_{i-1} from an embedding for G_i we consider the set of edges of G_{i-1} whose contraction resulted in G_i. We then reverse the process by carefully expanding and embedding one edge from that set at a time. Throughout this process we maintain the three locality conditions for edge expansions/contractions.

2.1 Edge Contraction and Separating Triangles

Contracting an edge is a standard operation on planar graphs, see [6]. We say that an edge $e = (u, v)$ of G is *contracted* when its endpoints, u and v, are replaced by a new vertex w such that all resulting multiple edges are removed. Ideally, we would like to perform edge contractions in a straight-line drawing that can be continuously animated so as to preserve planarity. Furthermore, so as to preserve the viewer's mental map, we prefer that only the endpoints of the contracted edge move, resulting in only minimal changes in the drawing.

It is well-known that contracting an edge in a planar graph results in a planar graph [6]. Note that this does not imply that contracting an edge in a straight line planar drawing of a graph results in a straight line planar drawing! More precisely, consider a straight-line planar drawing of a graph and an edge to be contracted. Suppose we are not allowed to move any other vertices in the drawing except the two involved. Then there exist drawings in which the contraction of some such edge introduces a crossing. We show this with an example in Fig. 1.

We have seen one of the problems that occur when an edge in an embedded graph is contracted. Another problem can occur even if we do not have a fixed embedding. When the contracted edge is a part of a separating triangle, the

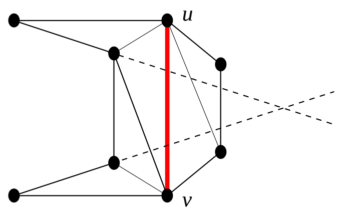

Fig. 1. A subgraph of an embedded fully triangulated graph. Edge (u, v) cannot be contracted without introducing a crossing, if we are to keep all other vertices fixed.

resulting graph is not fully triangulated and in fact may have many different embeddings. We call a triangle in G is a *separating triangle* if the removal of its vertices and their adjacent edges disconnects G.

Thus, we can divide the edges of G into two categories depending on the effect their contraction has on the resulting graph. We say that an edge is *simple* if it is not a part of a separating triangle. Edges that are part of separating triangles we call *non-simple*. Non-simple edges present problems when contracted, so we will be contracting only simple edges, for their contraction can be continuously animated while preserving planarity using straight lines. Moreover, eliminating the parallel edges after contracting a simple edge in a maximally planar graph results in a maximally planar graph.

3 Simple Matching in Maximally Planar Graphs

In this section we show that any maximal matching that uses only simple edges contains a constant fraction of all the edges in G, provided G is maximally planar. Next we show how to find a matching that can be used to contract the graph so that the resulting graph is maximally planar. Furthermore, if the size of that matching is $O(n)$, then after repeating this process $O(\log n)$ times we are left with only a constant number of vertices. Thus, we need to show that we can construct a maximal matching with $O(n)$ edges such that their contraction results in a maximally planar graph.

Let G' be the graph obtained from G by removing all the non-simple edges. We start by showing that any maximal matching in G' contains at least $n/12$ edges. To prove this claim we construct a maximal matching in G' and consider faces of different lengths. Recall that the length of a face refers to the number of vertices on that face. We break the faces of G' into three classes, A, B, C, respectively faces of length 3, faces of length 4, and faces of length 5 or more. We then count the number of unmatched nodes in faces of the different classes. Finally, when we factor in over-counting we show that any maximal matching must contain at least $n/12$ edges.

Lemma 1. *If f_i is a face in A, there is at most one unmatched vertex in f_i and this vertex has degree at least 3. If f_i is a face in B or C, then there exist at most $l(f_i)/2$ unmatched nodes in f_i.*

For the unmatched vertices on faces in A we can show that they have degree at least 3 in G'. This is not necessarily true for the unmatched vertices on faces in B or C. We show, however, that if a pair of unmatched vertices on a face in B have degrees 2 in G' then G belongs to a special class of graphs $\mathcal{H}$. If $G \notin \mathcal{H}$ then for any face $f_i \in B$, at most one vertex on that face has degree 2.

Lemma 2. *Let $\mathcal{H}$ be the class of maximally planar graphs in which there exist two vertices, u, v such that every other vertex in the graph is adjacent to both u and v. Then if $H \in \mathcal{H}$, any maximal matching that uses only simple edges contains $n/12$ or more edges.*

Lemma 3. *Let G and G' be a planar graph and the induced graph on G in which all the non-simple edges have been removed. If there exists a face in B with more than one vertex of degree 2, then $G \in \mathcal{H}$.*

We have shown that if G' has a face of length four with more than two nodes of degree 2, then $G \in \mathcal{H}$ and hence any maximal matching in G' contains at least $n/12$ edges (from Lemma 2). Finally we show that the same result holds for all maximally planar graphs.

Theorem 1. *Let G be a maximally planar graph and let M be the set of matched vertices in a maximal matching which uses only simple edges. Then $|M| \geq n/6$, where n is the number of vertices of G.*

4 Algorithm and Analysis

Before we can consider a particular embedding we must show how to obtain all the graphs in the hierarchy, $G_0, G_1, \ldots, G_k$. Recall that $G_0 = G$ is a fully triangulated planar graph on n vertices. To construct G_{i+1} from G_i we find a matching E_i of G_i and perform the graph contraction using the edges in E_i. We repeat this process until G_{i+1} is a singleton graph.

Set E_i for $0 \leq i < k$ contains a maximal matching on the edges of G_i with some added constraints. It is important that after the contraction of the edges in E_i the resulting graph G_{i+1} remains fully triangulated. In order to preserve the mental map, the three locality conditions must be maintained. Finally, in order to maintain a small hierarchical height, $|E_i|$ must be a constant fraction of the edges in G_i. Thus, the constraints that we have on E_i are as follows:

1. E_i is a matching of simple edges.
2. Using the locality conditions, contracting E_i yields a maximally planar G_{i+1}.
3. $|E_i| \geq |V(G_i)|/c$, for some constant $c > 1$.

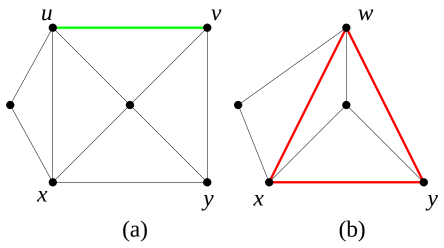

Fig. 2. Edges (u, v) and (x, y) in part (a) are both simple, do not share an endpoint, and can be contracted as a part of a matching. After (u, v) is contracted to w in part (b), edge (x, y) becomes a part of a separating triangle and so it should not be contracted.

Note that condition (1) does not imply condition (2), see Fig. 2. Before we proceed we show how to produce a set E_i which satisfies the above three conditions. Suppose we have graph G_i and we want to create set E_i so that when all the edges in E_i are contracted, we get G_{i+1}. We will contract simple edges of G_i one at a time. When an edge (u, v) is contracted, it is replaced by a node w. The next time an edge is contracted, it cannot have w as an endpoint. Let W_i be the set of vertices that were created as a result of contractions in phase i. The edges that we place in E_i must be a matching, and so when a new edge is considered for contraction, it cannot have an endpoint in W. Finally, let S_i be the set of vertices of G_i of small degrees. More precisely, let $S_i = \{v \in V(G_i) : d_{G_i}(v) < 39\}$.

In general, G_i is transformed into G_{i+1} one edge contraction at a time using the edges in E_i *in the order they were chosen.* Call the intermediate graphs from G_i to G_{i+1}, $G_i = G_{i,0}, G_{i,1}, \ldots, G_{i,j} = G_{i+1}$, and consider the algorithm on Fig. 3.

Lemma 4. *Let* $|V(G_i)| = n_i$. *Then* $|E_i| \geq n_i/50$.

Proof Sketch: For G_i with more than 3 vertices, $|E_i| \geq 1$. Then consider the sequence of intermediate graphs $G_{i,0}, G_{i,1}, \ldots, G_{i,j}$ and let $G_{i,j}$ have no more edges that could be added to E_i. Observe that we have contracted exactly j edges of G_i and so $|V(G_{i,j})| = n_i - j$. Then from Theorem 1 there are $(n_i - j)/12$ edges in any maximal matching of $G_{i,j}$ which uses only simple edges. Consider such a matching M. We are not allowed to add M edges with endpoints in W_i. But since $|W_i| = j$, at most j of the edges in M can have endpoints in W. Also note that if both endpoints of a simple edge in the matching have degrees greater than or equal to 39 in G_i they cannot be added to M. Note that if there exist at most k nodes of degree greater than or equal to 39, then there are at most $k/2$ such edges. It is easy to show that $k < n_i/12$: Suppose there are k vertices of degrees 39 or more in G_i. Since G_i is fully triangulated, every vertex has degree at least 3 and since G_i is maximally planar, the sum of the degrees

```
match($G_i, E_i$)
    $j \leftarrow 0$
    $G_{i,j} \leftarrow G_i$
    $W_i \leftarrow \emptyset$
    while ($S_i \neq \emptyset$)
        Let $u_j \in S_i$, $S_i = S_i \setminus \{u_j\}$
        if $\exists (u_j, v_j) \in G_{i,j}$, s.t. $v_j \notin W_i$ and $(u_j, v_j)$ is simple
            $E_i \leftarrow E_i \cup \{(u_j, v_j)\}$
            Contract $(u_j, v_j)$ to $w_j$ to get $G_{i,j+1}$
            $W_i \leftarrow W_i \cup \{w_j\}$
            $j \leftarrow j + 1$
    return($G_{i,j-1}$)
```

Fig. 3. Create G_{i+1} from G_i by contracting a sequence of edges in E_i.

is twice the sum of the edges. Then $39k + 3(n_i - k) \leq 6n_i - 12$. From this we get that $k < n_i/12$.

We stopped selecting good edges from G_i when we got to graph $G_{i,j}$ in which we could not find a simple edge to contract. The only other types of edges that might be available in $G_{i,j}$ but which we cannot take are those that were at some point non-simple, but later became simple. Also, there can be at most j such edges. Then $(n_i - j)/12 - 2j - n_i/24 \leq 0$ which implies $j \geq n_i/50$. Thus, if we cannot find another edge to add to the matching, we must have $|E_i| = j \geq n_i/50$ which completes the proof sketch. □

From the result above it follows that the height of the hierarchy is $O(\log n)$. We next argue that one call to match(G_i, E_i) takes $O(n_i \log n_i)$ time, and since n_{i+1} is a constant fraction of n_i, the $O(\log n)$ calls to match(G_i, E_i) take $O(n \log n)$ time overall thus yielding the desired theorem:

Theorem 2. *The clustering algorithm runs in $O(n \log n)$ time and produces a sequence of graphs $G_0, G_1, \ldots, G_k$ such that G_i is maximally planar for all $0 \leq i \leq k$ and $k = O(\log n)$.*

5 Constructing the Embedding

After we obtain the combinatorial graphs $G_0, G_1, \ldots, G_k$ we have to embed them in planes $z = 0, z = 1, \ldots, z = k$. While constructing the combinatorial graphs is a bottom up process, constructing the embedding is a top-down one. The first graph to be embedded is G_k, which only has one vertex. We then expand the edges in E_{k-1} one at a time, in the reverse order of their insertion. We then argue that this can be done in a way which guarantees that no crossings are introduced. We need the following lemma.

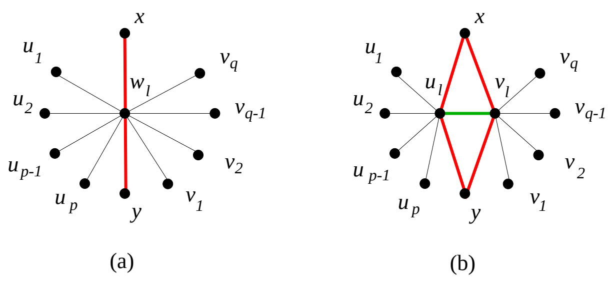

Fig. 4. Vertex w_l and its neighbors in $G_{i,l+1}$ (a) before expansion (b) after expansion.

Lemma 5. *Let G be a maximally planar graph embedded in the plane without crossings. For any $v \in V(G)$, there exists a ball of radius $\epsilon > 0$ such that if v is placed anywhere inside that ball, the embedding has no crossings.*

Proof Sketch: The main idea is to consider the visibility region around vertex v. Any point inside that region can "see" all the neighbors of v. It is not hard to show that this region cannot be empty. This would imply the existence of $\epsilon > 0$ for which the ball of size ϵ fits inside the visibility region. □

Theorem 3. *Given combinatorial representations of graphs $G_k, G_{k-1}, \ldots, G_0$ we can embed them in the planes $z = k, z = k-1, \ldots, z = 0$ so that there are no crossings in any of the drawings.*

Proof Sketch: We first embed G_k in the plane $z = k$ without crossings using any straight-line drawing method. Suppose we have embedded $G_k, G_{k-1}, \ldots, G_i$. We will show how to embed G_{i-1} given an embedding for G_i. Recall that we obtained G_i from G_{i-1} through a series of edge contractions from the edge set $E_{i-1} = \{(u_0, v_0), (u_1, v_1), \ldots (uj, vj)\}$ which produced graphs $G_{i-1,0}, G_{i-1,1}, \ldots,$ $G_{i-1,j} = G_i$. We now reverse the process and expand G_i back to G_{i-1} through the exact opposite sequence of expansions. Since we have an embedding for G_i in the plane $z = i$, we can embed $G_{i-1,j}$ in the plane $z = i-1$. Next we expand edge (u_j, v_j) by replacing vertex w_j by the pair u_j, v_j. The resulting graph is $G_{i,j-1}$ and we embed it without a crossing. We proceed until we get to $G_{i,0}$. We next show how to embed $G_{i,l}$ given an embedding for $G_{i,l+1}$, for $0 \leq l < j$.

Assume we have an straight-line embedding for $G_{i,l+1}$ without crossings on the plane $z = i$. To get $G_{i,l}$ we must expand vertex w_l back to edge (u_l, v_l). Consider the subgraph on Fig. 4. Let x and y be the neighbors in common for u_l and v_l. We then consider the ball of maximal radius around w_l which sees all neighbors (we know it is of radius $\epsilon > 0$ from Lemma 5). Consider a diameter in this ball which is perpendicular to the line connecting x and y. Place u_l and v_l on the two ends of the diagonal. □

We define the *drawing of a clustered graph* $C = (G, T)$ as in [3]. Graph G is drawn as usual, while for every node $v \in T$ the cluster is drawn as a simple closed region R such that:

– all sub-cluster regions of R are completely contained in the interior of R.
– all other cluster regions are completely contained in the exterior of R.
– if there is an edge e between two vertices contained in a cluster ν, then the drawing of e is completely contained in R.

Following the definitions of Eades *et al.*, the drawing of edge e and region R have an *edge 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. Graphs with c-planar drawings are c-planar.

Theorem 4. *The clustered graph $C = (G, T)$ produced by our algorithm is c-planar and a c-planar embedding can be obtained in $O(n^2)$ time.*

Proof Sketch: It suffices to show that there exists a drawing of C which has no edge crossings and no edge-region crossings. Let us embed G using any planar embedding algorithm. Define the region of a cluster, ν to be the simple closed curve around the subgraph of G induced by the cluster, $G(\nu)$. By the definition of the clustering in our algorithm, the subgraph $G(\nu)$ is connected.

If u is a vertex not in cluster ν, then u cannot be contained inside the region R. Assume that u is contained in R. If we contract the edges of ν in the order defined by our algorithm, eventually u will be inside a triangular face. But then none of the edges on that face can be contracted. This is a contradiction since ν is eventually contracted to one vertex.

Finally, since G is embedded in the plane without crossings and the regions are connected there can be neither edge crossings nor edge-region crossings. Therefore C is c-planar and from [3] it follows that the c-planar embedding can be produced in $O(n^2)$ time. □

References

1. C. A. Duncan, M. T. Goodrich, and S. G. Kobourov. Balanced aspect ratio trees and their use for drawing very large graphs. *Proc. of 6th Symposium on Graph Drawing (GD'98)*, LNCS 1190:101–112, 1998.
2. P. Eades and Q. W. Feng. Multilevel visualization of clustered graphs. *Proc. of 4th Symposium on Graph Drawing (GD'96)*, LNCS 1190:101–112, 1996.
3. P. Eades, Q. W. Feng, and X. Lin. Straight-line drawing algorithms for hierarchical graphs and clustered graphs. *Proc. of the 4th Symposium on Graph Drawing (GD'96)*, LNCS 1190:113–128, 1997.
4. I. Fary. On straight lines representation of planar graphs. *Acta Sci. Math. Szeged*, 11:229–233, 1948.
5. Q.-W. Feng, R. F. Cohen, and P. Eades. Planarity for clustered graphs. *ESA'95*, LNCS 979:213–226, 1995.
6. R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM J. Appl. Math.*, 36:177–189, 1979.
7. F. J. Newbery. Edge concentration: A method for clustering directed graphs. In *Proceedings of the 2nd International Workshop on Software Configuration Management*, pages 76–85, Princeton, New Jersey, October 1989.

8. S. C. North. Drawing ranked digraphs with recursive clusters. *ALCOM International Workshop PARIS 1993 on Graph Drawing and Topological Graph Algorithms (GD'93)*, September 1993.
9. Sablowski and Frick. Automatic graph clustering. *Proc. of 4th Symposium on Graph Drawing (GD'96)*, LNCS 1190:395–400, 1996.
10. W. Schnyder. Embedding planar graphs on the grid. In *Proc. 1st ACM-SIAM Sympos. Discrete Algorithms*, pages 138–148, 1990.
11. K. Sugiyama and K. Misue. Visualization of structural information: Automatic drawing of compound digraphs. *IEEE Trans. Softw. Eng.*, 21(4):876–892, 1991.
12. K. Wagner. Bemerkungen zum vierfarbenproblem. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 46:26–32, 1936.