8. Drawing Clusters and Hierarchies

Ralf Brockenauer and Sabine Cornelsen

Large graphs such as WWW connection graphs or VLSI schematics cannot be drawn in a readable way by traditional graph drawing techniques. An approach to solve this problem is, for example, fish-eye representation, which allows one to display a small part of the graph enlarged while the graph is shown completely (see e.g. Formella and Keller (1995)). Another way is drawing only a part of the graph. The method presented here is clustering, that is grouping the vertex set.

Apart from the use of clustering to draw large graphs, already clustered graphs occur in applications such as statistics (see e.g. Godehardt (1988)), linguistics (see e.g. Batagelj et al. (1992)) or divide and conquer approaches. To visualize these structures it is also important to find a method of drawing clustered graphs in an understandable way.

In this chapter, Section 8.1 gives an overview of several terms occurring in connection with clustering in the literature. Section 8.2 presents a few main methods of finding good clusters. The following three sections introduce some graph drawing algorithms for clustered graphs. Beginning with the special case of planar drawing methods in Section 8.3, Section 8.4 works on general graphs with a hierarchical structure, called compound graphs, and Section 8.5 deals with arbitrarily clustered graphs using force-directed methods. Finally, Section 8.6 shows a case study for drawing partially known huge graphs.

8.1 Definitions

The usage of the term clustering is not determined uniquely in the literature. In this chapter several terms concerning clustering are defined to give an overview. We only consider vertex clustering, but it is worth mentioning that clustering with respect to edges can be of interest as well. A method to do this can be found in Paulish (1993, Chapter 5).

Clustering of graphs means grouping of vertices into components called clusters. Thus, clustering is related to partitioning the vertex set.

Definition 8.1 (Partition). *A* (k-way) partition *of a set C is a family of subsets $(C_1, \ldots, C_k)$ with*

- $\bigcup_{i=1}^{k} C_i = C$ *and*
- $C_i \cap C_j = \emptyset$ *for* $i \neq j$.

The C_i are called parts. *We refer to a* 2*-way partition as a* bipartition.

Now, we can define one of the most basic definitions of clustered graphs.

Definition 8.2 (Clustered Graph). *A* clustered graph *is a graph with a partition* $(C_1, \ldots, C_k)$ *on the vertex set. The* C_i *are called* cluster.

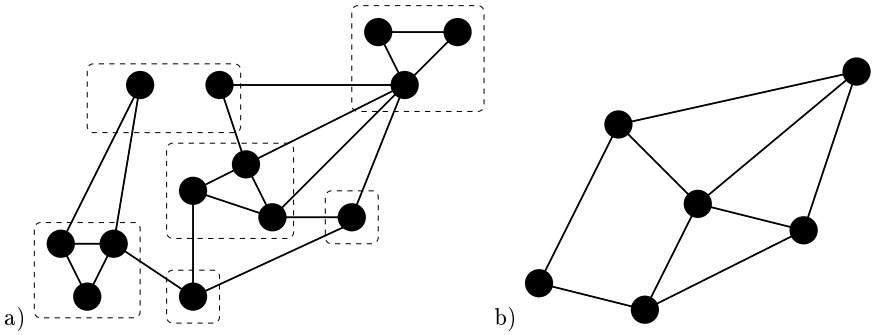


Fig. 8.1. a) A clustered graph. Clusters are framed with rectangles. b) Quotient graph of the clustered graph in a).

Sometimes (e.g. Alpert and Kahng (1995)) the term clustering is only used for large $k \in \theta(n)$ where n is the number of vertices. The expression clustered graph is also used to denote the quotient graph defined by the partition.

Definition 8.3 (Quotient Graph). *For a partition* $(C_1, \ldots, C_k)$ *on the vertex set of a graph* $G = (V, E)$*, the quotient graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *is defined by* shrinking *each part into a single* node*, i.e.*

– $\mathcal{V} = \{C_1, \ldots, C_k\}$ *and*
– $(C_i, C_j) \in \mathcal{E} \iff i \neq j$ *and* $\exists v \in C_i, w \in C_j \quad (v, w) \in E$.

The elements of $\mathcal{V}$ *are called* nodes.

The quotient graph of the clustered graph in Figure 8.1 a) is shown in Figure 8.1 b). A drawing of the quotient graph is also called a *black-box drawing* (Paulish, 1993) of the corresponding clustered graph. Another way of shrinking subgraphs into a single node is proposed by Lengauer (1990). He calls the construction hierarchical graph. Lengauer used this structure to find faster algorithms, e.g., for planarity testing (Lengauer, 1989) or connectivity testing, for large graphs.

Definition 8.4 (Hierarchical Graph). *A* hierarchical graph *is a finite sequence* $\Gamma = (G_1, \ldots, G_k)$ *of graphs* G_i *called* cells*. The vertex set of the cells is divided into* pins *and* inner vertices*. The set of inner vertices, again, is divided into* terminals *and* non-terminals*. Each non-terminal has a* type*. The type of a non-terminal in* G_i *is a cell* G_j *with* $j < i$*. The degree of a non-terminal* v *is the number of pins of its type* G *and the neighbors of* v *are bijectively associated with the pins of* G.

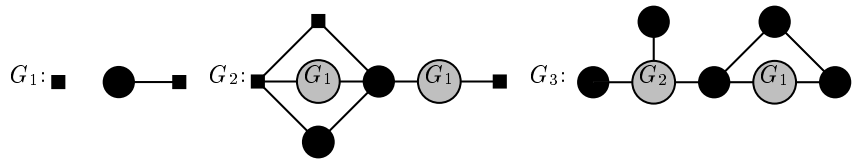


Fig. 8.2. Example for a hierarchical graph. Pins are drawn as rectangles, inner vertices as circles and non-terminals are shaded grey. The bijection between the neighbors of a non-terminal and the associated pins is given via the position in the figure.

An example for a hierarchical graph is shown in Figure 8.2. Note, that the cells need not to be connected. A hierarchical graph represents a graph which is obtained by *expansion*. It is a substitution mechanism that glues pins of a cell to neighbors of non-terminals the type of which is this cell. Note that if $\Gamma = (G_1, \ldots, G_k)$ is a hierarchical graph, so is any *prefix* $\Gamma_i = (G_1, \ldots, G_i)$, $1 \leq i \leq k$.

Definition 8.5 (Expansion). *The* expansion $G(\Gamma)$ *of a hierarchical graph* $\Gamma = (G_1, \ldots, G_k)$ *is obtained recursively as follows:*

$k = 1$: $G(\Gamma) = G_1$

$k > 1$: *For each non-terminal* v *of* G_k*, let* v *be of type* G_j*. Delete* v *and its incident edges and insert a copy of* $G(\Gamma_j)$ *by identifying pins of* $G(\Gamma_j)$ *with their associated vertex in* G_k*.*

Thus, a hierarchical graph is a clustering of the expansion graph and clusters can include other clusters. This can be illustrated as a tree – the *hierarchy tree*. The expansion and the hierarchy tree of the hierarchical graph in Figure 8.2 are shown in Figure 8.3.

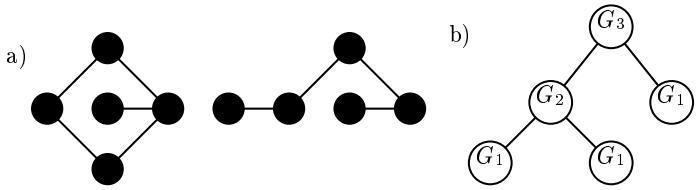


Fig. 8.3. Example for the a) expansion and b) inclusion tree of the hierarchical graph in Figure 8.2.

A quite similar concept introduced by Feng et al. (1995) are hierarchical clustered graphs.

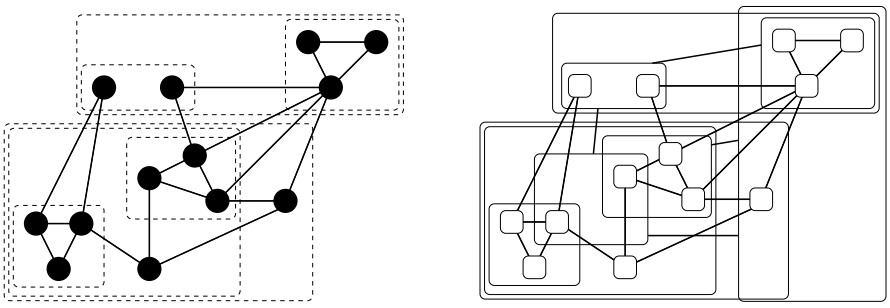
Definition 8.6 (Hierarchical Clustered Graphs). Hierarchical clustered graphs $C = (G, T)$ *consist of a graph* $G = (V, E)$ *and a rooted tree*

T such that the leaves of T are exactly V. Vertices of T are called nodes. *Each node ν of T represents the* cluster *$V(\nu)$ of leaves in the subtree of T rooted at ν. T is called the* inclusion tree *of C. An edge e is said to be* incident *to a cluster $V(\nu)$, if $|e \cap V(\nu)| = 1$.*

An example of a hierarchical clustered graph is shown in Figure 8.4 a). Algorithms for drawing planar hierarchical clustered graphs are presented in Section 8.3. A way of generalization is to allow clusters to intersect. Combined with a hierarchical structure this leads to compound graphs presented by Sugiyama and Misue (1991).

Definition 8.7 (Compound Graphs). *A* compound (directed) graph *is a triple $D = (V, E, I)$ such that $D_a = (V, E)$ is a (directed) graph and $D_c = (V, I)$ is a directed graph. The elements of E are called* adjacency edges*, those of I* inclusion edges.

Thus, $(v, w) \in I$ means that v includes w. Of course, this interpretation only makes sense if the directed graph D_c is acyclic. An example of a compound graph is shown in Figure 8.4 b). In Section 8.4 a drawing algorithm for compound graphs is given in the special case where D_c is a rooted tree and D_a is a directed graph, such that no vertex is adjacent to an ancestor. A hierarchical clustered graph can be seen as a compound graph where D_c is a rooted tree and adjacency edges are only incident to leaves. Note the different meaning of V.



(a) (b)

Fig. 8.4. a) A hierarchical clustered graph. The inclusion tree is drawn by the inclusion representation. b) A compound graph. Inclusion edges are drawn as including rectangles.

8.2 Clustering Methods

There is a large number of clustering approaches. This section will only mention some of them to give an idea of the main methods. A good overview is given in Alpert and Kahng (1995) or Jain and Dubes (1988). The approaches introduced in this section that are not cited separately can be found there as well. Some partitioning methods concerning VLSI are also summarized in Lengauer (1990).

Clustering is a type of classification. This classification can be *extrinsic* or *intrinsic*. Extrinsic classification uses category labels on the objects and clusters are defined by these categories. For example, take trade relation between world wide spread companies and countries as clusters. On the other hand intrinsic classification is only based on the structure of the graph. In the following discussion, we consider intrinsic classification.

There are two main goals in clustering graphs. The first, which has applications, for example, in VLSI, parallel computing and divide-and-conquer algorithms, is to partition a graph into clusters of about the same size and with as few edges connecting the clusters as possible. The second one, which is a method used in statistical applications, is to explore the structure of the data. Thus, the number of clusters is not fixed.

8.2.1 k-Way Partition

The first goal can be formalized as the *Min-Cut k-Way Partition* (cf. Lengauer (1990) p. 253). In the course of the following passage, let $G = (V, E)$ be a graph with vertex-weight $c : V \to \mathbb{N}$ and edge-weight $w : E \to \mathbb{N}$.

Definition 8.8 (Min-Cut k-Way Partition). *Given a fixed $k \in \mathbb{N}$ and $b(i), B(i) \in \mathbb{N}$ for $i = 1, \ldots, k$, find among all k-way-partitions $(C_1, \ldots, C_k)$ of V which satisfy $b(i) \leq c(C_i) \leq B(i)$ for all $i = 1, \ldots, k$ one that minimizes the weight of the partition*

$$w(C_1, \ldots, C_k) = \frac{1}{2} \sum_{i=1}^{k} \sum_{\substack{e \in E \\ |e \cap C_i| = 1}} w(e) \,.$$

Exact cluster size balance is achieved by setting $b(i) = c(V)/k - \epsilon$ and $B(i) = c(V)/k + \epsilon$, where $\epsilon > 0$ may be necessary to obtain a solution at all.

Move-based Approaches. Unfortunately, the multiway partition problem is $\mathcal{NP}$-complete even in the special case of bipartition. A classical good graph bipartitioning heuristic was introduced by Kernighan and Lin (1970). A natural local search method for solving this problem is to start with an initial bipartition and to exchange pairs of vertices across the cut, if doing so improves the cut-size. To reduce the danger of being trapped in local minima, Kerninghan and Lin modified the search, proceeding in a series of passes.

During each pass of the algorithm, every vertex moves exactly once. At the beginning of a pass each vertex is unlocked. Iteratively the pair of unlocked vertices with the highest gain is swapped, where the *gain* of vertices $v_1 \in C_1, v_2 \in C_2$ is defined by

$$w(C_1, C_2) - w((C_1 \cup \{v_2\}) \setminus \{v_1\}, (C_2 \cup \{v_1\}) \setminus \{v_2\})$$

that is the decrease in cut-weight that results from the pair swap. Then, both swapped vertices are locked. The swapping process is iterated until all vertices become locked. The bipartition with the lowest cut-weight observed during the pass is the initial bipartition for the next pass. The algorithm terminates when a pass fails to find a solution with lower weight than its initial bipartition.

Maintaining a sorted list of gains, the complexity of this algorithm is in $O(n^2 \log n)$. Fiduccia and Mattheyses modified the algorithm of Kerninghan and Lin to permit an $O(|E|)$ implementation. The main difference is that a new bipartition is derived by moving a single vertex either from C_1 to C_2 or from C_2 to C_1 instead of exchanging two of them. Therefore, the algorithm must violate the exact cluster size balance constraint. The solution is permitted to deviate from an exact bipartition by the size of the largest vertex. The algorithm of Fiduccia and Mattheyses can also be extended to $k > 2$.

There are several functions combining cluster size balance and minimization of cut weights within a single objective. One of them is the *Ratio Cut Partition* proposed by Wei and Cheng (1991) for $k = 2$. Their approach has several generalizations for arbitrary k. One of them is presented by Roxborough and Sen (1997).

Definition 8.9 (Ratio Cut Partition). *Find among all k-way-partitions $(C_1, \ldots, C_k)$ of V one that minimizes*

$$R(C_1, \ldots, C_k) = \frac{w(C_1, \ldots, C_k)}{c(C_1) \cdot \ldots \cdot c(C_k)}.$$

Finding a ratio cut partition is also $\mathcal{NP}$-complete. In Wei and Cheng (1991) a heuristic based on the algorithm of Fiduccia and Mattheyses is proposed to find good bipartitions. In Figure 8.5 an example is shown where the ratio cut partition is a much more intuitive one than the exact cluster size balanced min-cut bipartition.

Spectral Methods. For a graph $G = (\{v_1, \ldots, v_n\}, E)$ with enumerated vertex set, a partitioning solution can be represented in terms of vectors and matrices.

Definition 8.10 (Characteristic Vector). *For a graph G with a given a k-way partition $(C_1, \ldots, C_k)$ the* characteristic vector *for cluster C_h is the n-dimensional vector $\mathbf{x}_h = (x_{1h}, \ldots, x_{nh}) \in \{0,1\}^n$ with $x_{ih} = 1$ if and only if $v_i \in C_h$. The $n \times k$ matrix X with column h equal to $\mathbf{x}_h$ is the* assignment matrix *of the partition.*

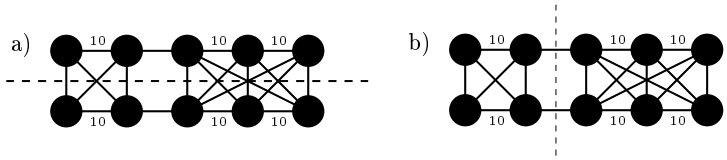


Fig. 8.5. a) Exact cluster size balanced min-cut bipartition and b) ratio cut bipartition.

Let $A = (a_{ij})$ be the adjacency matrix of an undirected graph G and $D = (d_{ij})$ the degree matrix, that is $d_{ij} = \deg(v_i)$ for $i = j$ and zero otherwise. The *Laplacian matrix* of G is defined as $L = D - A$. Since L is a symmetric matrix

- all eigenvalues of L are real and
- there is a basis of the n-dimensional space of mutually orthogonal eigenvectors of L.

Since $\mathbf{x}^T L\mathbf{x} = \frac{1}{2}\sum\sum a_{ij}(x_i - x_j)^2 \geq 0$ for all $\mathbf{x} \in \mathbb{R}^n$, matrix L is positive semi-definite and thus, all eigenvalues are non-negative. Furthermore, the columns of L add to zero and we get

- the smallest eigenvalue of L is 0 with corresponding eigenvector $(1, \ldots, 1)$. (The multiplicity of 0 as an eigenvalue of L is equal to the number of connected components of G.)

Now, let $\mathbf{x}$ denote the characteristic vector for one part of a given bipartition (C_1, C_2) in a connected undirected graph, then we can express the value of the cut defined by the bipartition by

$$w(C_1, C_2) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}(x_i - x_j)^2 = \mathbf{x}^T L\mathbf{x}.$$

Allowing non-discrete solutions, a normalized eigenvector μ to the smallest positive eigenvalue minimizes $\mathbf{x}^T L\mathbf{x}$ among all normalized $\mathbf{x}$. Although a non-discrete solution for $\mathbf{x}$ is meaningless, this result suggests heuristically finding the discrete solution closest to μ. Given cluster size constraints $|C_1| = m_1$ and $|C_2| = m_2$, the closest discrete solution is obtained by placing the m_1 vertices with the highest coordinates of μ in C_1 and the rest in C_2 or vice versa.

This approach is known as *spectral bipartitioning*. Unfortunately, it can be arbitrarily worse than optimal, as illustrated by the following example: Let G be the graph shown in Figure 8.6 in which two $n/4$-cliques are connected, each by a single edge, to an $n/2$-clique. Spectral bipartition will cut $K_{n/2}$ into equal halves, cutting $(n/4)^2$ edges. But the optimal cluster size balanced bipartition has weight 2.

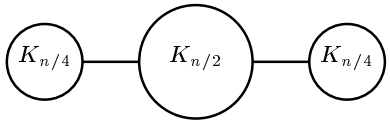


Fig. 8.6. Example of bad spectral bipartition.

To improve the result, Frankle and Karp proposed to find an characteristic vector that is close to a linear combination of the eigenvectors of the d smallest eigenvalues. A summary of spectral clustering including an extension to $k > 2$ is given in Alpert and Kahng (1995) Section 4.

8.2.2 Structural Clustering

We now turn to the second goal of graph clustering which tries to identify certain intuitive properties ρ such as cliques or connectivity. In contrary to the last section, the number of clusters is not given. One important algorithm used to solve this problem is *agglomerative clustering*, which starts with the n-way partition and constructs iteratively a k-way partition from the $k+1$-way partition.

For a complete graph $G = (V, E)$ with edge weight $w : E \to \mathbb{R}$ and no two edges having the same weight, the iterative step is defined in more detail, for example, by Hubert (see Jain and Dubes (1988) p. 63). For a given *threshold* d let $G_d = (V, \{e \in E; w(e) \leq d\})$. For a pair of clusters (C_r, C_s) in the $k+1$-way partition, define

$$Q_\rho(C_r, C_s) = \min\{d; \text{the subgraph induced by } C_r \cup C_s \text{ in } G_d \text{ is either complete or has property } \rho\}.$$

Merge cluster C_p and C_q if

$$Q_\rho(C_p, C_q) = \min_{r,s} Q_\rho(C_r, C_s).$$

Each specification of property ρ defines a new clustering method. Every cluster must at least be connected. Some suitable graph properties $\rho(k)$ with integer parameter k are listed below:

k-edge-connectivity: All pairs of vertices are joined by at least k edge disjoined paths. 1-edge-connectivity is also called *single-link* property and n-edge-connectivity is called *complete-link* property.

k-vertex-connectivity: All pairs of vertices are joined by at least k vertex disjoined paths.

vertex degree k: A connected graph such that each vertex has at least degree k.

diameter k: All pairs of vertices are joined by a path of length at most k.

An agglomerative algorithm constructs a hierarchical clustering. The above mentioned algorithm is serial. To minimize the height of the inclusion tree, an alternative strategy is to find many good clusters to merge, then perform all merges simultaneously.

A special drawing of the inclusion tree of a hierarchical clustered graph, that reveals the order in which clusters are merged, is called *dendrogram*. Cutting a dendrogram horizontally creates a partition of the vertex set. Figure 8.7 gives two examples on a weighted K_5, one for single- and the other for complete-link property.

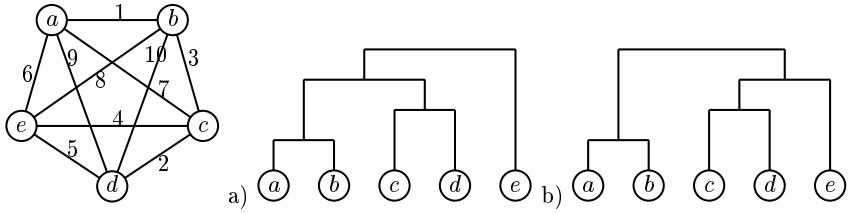


Fig. 8.7. Two dendrograms on a weighted K_5. a) single link, b) complete link.

8.2.3 Other Approaches

There are several other methods for clustering graphs. Duncan et al. (1998) presented a method to partition an already laid out graph along horizontal, vertical and diagonal lines. The approach of Sablowski and Frick (1996) is based on the successive identification of patterns in the graph. Other approaches consider constraints, e.g., some vertices should belong to different clusters. Nakano et al. (1997) presented a work on this topic.

Most of the clustering methods presented so far produce connected clusters. But it is important to know, that unconnected clusters might occur in practice. This can be seen immediately by considering extrinsic clustering, because it can be defined arbitrarily. But there are also some intrinsic classifications which produce unconnected components. They are, for example, used in social network analysis. The following two classifications presented by Wasserman and Faust (1994) are of this type.

Definition 8.11 (Structural Equivalence). *Two vertices v, w of a graph (V, E) are* structural equivalent *if and only if they have the same neighborhood, that is if and only if for all $u \in V$ holds*

$$(v, u) \in E \iff (w, u) \in E$$
$$(u, v) \in E \iff (u, w) \in E.$$

Thus, two vertices are structural equivalent if their rows and columns in the adjacency matrix are identical.

Definition 8.12 (Regular Equivalence). *Two vertices v and w of a graph (V, E) are* regular equivalent *($v \cong w$) only if they have equivalent neighborhoods, that is only if for all $u \in V$ holds*

$$(v,u) \in E \Longrightarrow \exists u' \in V \quad u' \cong u \ \wedge \ (w,u') \in E$$
$$(u,v) \in E \Longrightarrow \exists u' \in V \quad u' \cong u \ \wedge \ (u',w) \in E.$$

Structural equivalence is a special case of regular equivalence. Regular equivalence partitioning is not uniquely determined. The partition with the fewest equivalence classes that is consistent with the definition of regular equivalence is called the *maximal regular equivalence*. For example, in a tree where all leaves have the same height, taking the levels as equivalence classes yields the maximal regular equivalence on that tree.

8.3 Planar Drawings of Hierarchical Clustered Graphs

In her PhD thesis, Feng (1997) presented a characterization for planar connected hierarchical clustered graphs and introduced some algorithms for drawing them.

Definition 8.13 (Connected Hierarchical Clustered Graphs). *A hierarchical clustered graph $C = (G, T)$ is* connected*, if each cluster induces a connected subgraph of G.*

Definition 8.14 (Drawing of Hierarchical Clustered Graph). *A* drawing *$\mathcal{D}$ of a hierarchical clustered graph $C = (G, T)$ includes the drawing of the underlying graph G and of the inclusion tree T in the plane. Each vertex of G is represented as a point and each edge $\{v, w\}$ as a simple curve between $\mathcal{D}(v)$ and $\mathcal{D}(w)$. Each non-leaf node ν of T is drawn as a simple closed region $\mathcal{D}(\nu)$ bounded by a simple closed curve such that*

- *$\mathcal{D}(\mu) \subset \mathcal{D}(\nu)$ for all descendents μ of ν.*
- *$\mathcal{D}(\mu) \cap \mathcal{D}(\nu) = \phi$ if μ is neither a descendent nor an ancestor of ν.*
- *$\mathcal{D}(e) \subset \mathcal{D}(\nu)$ for all edges e of G with $e \subset V(\nu)$.*
- *$\mathcal{D}(e) \cap \mathcal{D}(\nu)$ is one point if $|e \cap V(\nu)| = 1$.*

The drawing of an edge e and a region $\mathcal{D}(\nu)$ have an *edge-region-crossing*, if $e \cap V(\nu) = \phi$ but $\mathcal{D}(e) \cap \mathcal{D}(\nu) \neq \phi$. Drawings where this occurs, are allowed, but they are not c-planar.

Definition 8.15 (c-Planar). *A drawing of a hierarchical clustered graph is* c-planar *(compound planar), if there are no crossing edges and no edge-region-crossings.*

For example, the drawing of the graph shown in Figure 8.4 on page 197 is c-planar.

Theorem 8.16 (Characterization of c-Planar Graphs). *A connected hierarchical clustered graph $C = (G, T)$ is c-planar if and only if there exists a planar drawing of G, such that for each node ν of T all vertices of $V - V(\nu)$ are in the outer face of the drawing of $G(\nu)$.*

Proof. Consider a clustered graph $C = (G, T)$ with a c-planar drawing $\mathcal{D}$. Suppose there is a node ν of T and a vertex $v \in V - V(\nu)$ which is not drawn in the outer face of $\mathcal{D}(G(\nu))$. Hence, any simple region that contains $\mathcal{D}(G(\nu))$ must also contain v, contradicting the c-planarity properties.

Consider now a planar drawing of G, such that for each node ν in T, $G - G(\nu)$ is drawn in the outer face of the drawing of $G(\nu)$. It remains to add cluster boundaries. Since $G(\nu)$ is connected for each ν in T, the outer face is bounded by a – not necessarily simple – cycle. Thus, cluster boundaries are constructed recursively, following T from bottom to top, along their external facial cycle.

Based on Theorem 8.16, Feng et al. (1995) developed an algorithm which constructs a *c-planar embedding* of a hierarchical clustered graph, that is a circular ordering of the incident edges ordered around each cluster. The algorithm applies the PQ-tree technique presented by Booth and Lueker (1976) and modified by Chiba et al. (1985) and takes time $O(n^2)$ under the additional condition, that each non-leaf node of T has at least two children.

It tries to embed the subgraph $G(\nu)$ induced by each cluster $V(\nu)$ recursively, following T from bottom to top. To guarantee the conditions in Theorem 8.16 for each *virtual edge* $e \in E$, that is an edge such that $e \cap V(\nu) = \{v_e\}$ has cardinality one, an additional vertex w_e and an edge $\{v_e, w_e\}$ is added to $G(\nu)$. Further one of the additional vertices is connected to all other additional vertices (see Figure 8.8).

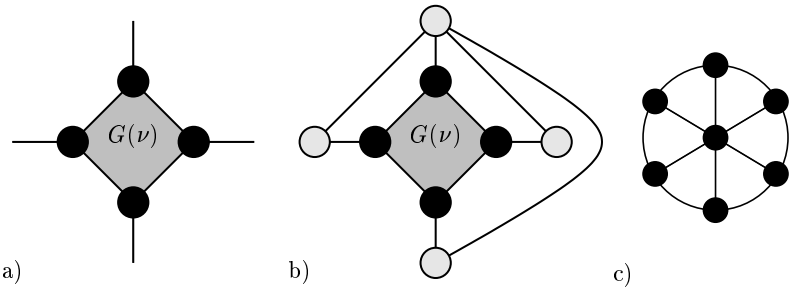


Fig. 8.8. a) Graph $G(\nu)$ with virtual edges is transformed into graph b). Additional vertices are shaded light grey. c) A wheel graph with 6 vertices on the rim.

To determine whether the embeddings of the children of a cluster ν can be combined to an embedding of $G(\nu)$, the graph $G(\mu)$ for each child μ of ν is replaced by a *representative graph* which is more or less constructed by

replacing 2-connected components in $G(\mu)$ by *wheel graphs*. A wheel graph consists of a vertex called *hub* and a simple cycle called *rim*, such that the hub is connected to every vertex on the rim (see Figure 8.8 c)). They showed that a representative graph with given ordering of the virtual edges can always be embedded in such a way that the rims are in the outer face without changing the ordering of the virtual edges.

8.3.1 Straight-Line Drawings with Convex Clusters

For a given c-planar embedding of a connected hierarchical clustered graph $C = (G, T)$, Eades et al. (1996a) gave an algorithm to construct a drawing of C such that the edges of G are drawn as straight lines and the regions are convex. This drawing of C can be constructed in time $O(n^2 \log n)$ which is dominated by the time needed for constructing the convex hull of the clusters.[1]

The algorithm works as follows. First, graph G is triangulated. Then an st-numbering[2] of the vertices of G is computed such that vertices in the same cluster are numbered consecutively. Such a numbering is called *c-st-numbering*. These numbers are now used as a layer assignment – thus, there is one vertex per layer – and an algorithm for constructing planar straight-line drawings of layered graphs, which is also presented by Eades et al. (1996a), is applied to draw the graph. Since each cluster has consecutive layers, the convex hull of its vertices satisfies all conditions of a region in a c-planar drawing.

Apart from the construction of a planar straight-line drawing of layered graphs, 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, Eades, Feng and Lin used a top-down approach, ordering the children of the root of T first and thus having a lexicographical numbering on the clusters. To compute an order of the child cluster of ν, an *auxiliary graph* $F(\nu)$ is computed from $G(\nu)$ by shrinking each child cluster to a vertex. If ν is the root of T, an edge $\{s, t\}$ not belonging to any child cluster of T is chosen and an st-numbering is computed.

If ν is not the root, vertices σ and τ are added to the auxiliary graph $F(\nu)$. For a virtual edge $\{v, w\}$ with $v \in V(\nu)$, let μ be the lowest ancestor of ν with $w \in V(\mu)$ and k the number of the child cluster of μ which w belongs to. If $g(\nu) < k$ then edge $\{v, \tau\}$ is added to $F(\nu)$. Otherwise, edge $\{\sigma, v\}$ is added. In case $s \in V(\nu)$, the vertex representing the cluster containing s is set to be σ; similarly for t and τ. Now a $\sigma\tau$-numbering is computed.

The only thing missing now is that the auxiliary graphs are 2-connected. This is a consequence of the following lemma.

[1] In Eades et al. (1998), the time complexity is improved to $O(n^2)$.

[2] For the definition of st-numbering see Definition 2.9 on page 27.

Lemma 8.17. *For every non-root node ν of the inclusion tree of a connected c-planar hierarchical graph $C = (G, T)$ with triangulated G, the subgraph of G induced by $V \setminus V(\nu)$ is connected.*

Proof. Suppose that the subgraph of G induced by $V \setminus V(\nu)$ has $k \geq 2$ components denoted by $F_1, \ldots, F_k$. Since G is triangulated, it has a unique planar embedding. By Theorem 8.16, all vertices of $G - G(\nu)$ are in the same face of $G(\nu)$. Since G is connected, there is a face f of G such that its boundary contains an edge connecting $G(\nu)$ and F_i and also an edge connecting $G(\nu)$ and F_j for a $j \neq i$. Because G is triangulated, f also contains an edge connecting F_i and F_j, a contradiction.

Unfortunately, there are hierarchical clustered graphs such that any c-planar straight-line convex drawing strategy results in poor area requirement and angular resolution. Eades, Feng and Lin gave a family $C_n = (G_n, T_n)$ of clustered graphs which require area $\Omega(2^n)$ and have angles between two edges incident to a vertex in $O(1/n)$. A sketch of the recursive construction of the underlying graphs G_n can be seen in Figure 8.9. The root of T_n is adjacent to two nodes A and B, which have children $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$, respectively.

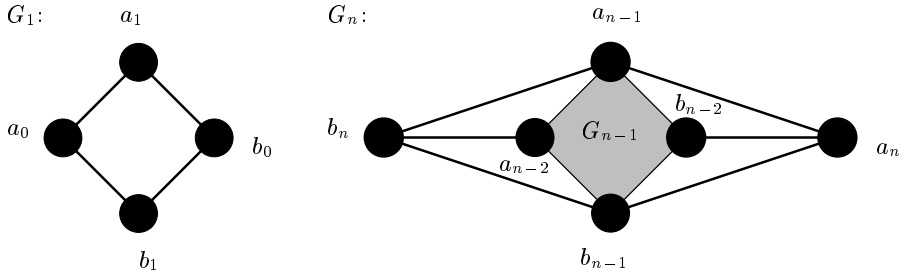


Fig. 8.9. Example for poor vertex and angular resolution.

8.3.2 Orthogonal Drawings with Rectangular Clusters

Eades and Feng (1997) gave an algorithm to construct a drawing of a hierarchical clustered graph $C = (G, T)$ with fixed embedding and degree at most 4 such that G is drawn orthogonal and the regions are rectangles. Using the constrained visibility representation, the algorithm takes time $O(n^2)$, the drawing space $O(n^2)$, and each edge has at most 3 bends.[3]

The algorithm works as follows: First triangulate G and compute a c-st-numbering as constructed in the previous section. Orient the edges from lower

[3] In Eades et al. (1999), the time complexity is improved to $O(n)$.

to higher numbers. Construct a directed graph G' from the oriented triangulation of G by adding four additional dummy vertices and replacing virtual edges for each cluster to ensure rectangular regions. Construct a constraint visibility representation of G' for a suitable set of non-intersecting paths. Construct an orthogonal drawing of G from the visibility representation of G' and finally reduce some bends.

How to construct a planar orthogonal drawing from a constraint visibility representation for non-clustered graphs is, for example, explained in Di Battista et al. (1999) Section 4.9 on page 130. A short introduction to this topic is also given in Section 6.4.2. So it remains to give the construction of G' and the additional constraints. Proceeding from the leaves to the root of T, for

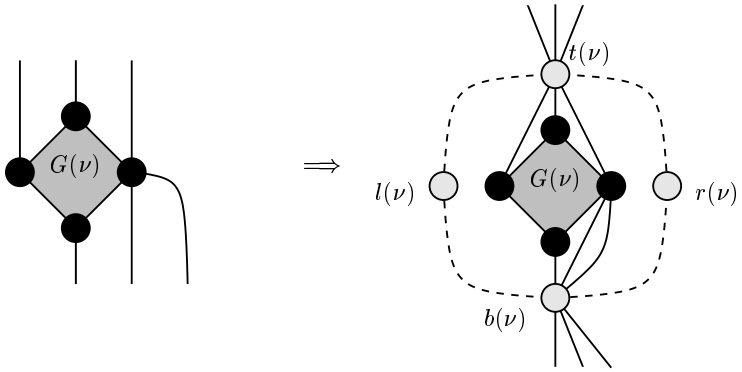


Fig. 8.10. Virtual edges are bunched together.

each non-leaf node ν of T add four dummy vertices denoted by $b(\nu)$ (bottom), $t(\nu)$ (top), $l(\nu)$ (left), and $r(\nu)$ (right) to $G(\nu)$ and split virtual edges (v, w) of $G(\nu)$ by a dummy vertex in the following way (illustrated in Figure 8.10):

- If $v \in V(\nu)$ replace (v, w) by $(v, t(\nu))$ and $(t(\nu), w)$.
- If $w \in V(\nu)$ replace (v, w) by $(v, b(\nu))$ and $(b(\nu), w)$.

Add edges $(b(\nu), r(\nu))$, $(r(\nu), t(\nu))$, $(b(\nu), l(\nu))$ and $(l(\nu), t(\nu))$ to $G(\nu)$.

For a node $\nu \neq s$ on the way from s to the root of T, let μ be the child of ν on this way. If $\mu \neq s$, add edge $(b(\nu), b(\mu))$, else add $(b(\nu), s)$. Similarly, for a node $\nu \neq t$ on the way from t to the root of T, let μ be the child of ν on this way. Add edge $(t(\mu), t(\nu))$ respectively $(t, t(\nu))$. By this construction, G' is a planar st-graph with $O(n)$ vertices and $O(n^2)$ edges.

Now, the alignment requirements in G' for the visibility representation, which is a set of paths ϕ, is specified. For each non-leaf node of T, the set ϕ contains the paths $(b(\nu), l(\nu), t(\nu))$ and $(b(\nu), r(\nu), t(\nu))$. Intercluster edges in G are replaced by paths in G'. These paths are also added to ϕ. Finally,

some paths containing edges incident to vertices of G are added to ϕ to avoid unnecessary bends like in the non-clustered version. Thus, ϕ is a set of non-intersecting paths in the sense defined below and a constraint visibility representation can be computed.

Definition 8.18 (Set of Non-Intersecting Paths). *Two paths p_1 and p_2 of a planar graph G with given embedding are said to be* non-intersecting *if they are edge disjoint and there is no vertex v of G with edges e_1, e_2, e_3, and e_4 incident to v in this clockwise order around v, such that e_1 and e_3 are in p_1 and e_2 and e_4 are in p_2. A set of pairwise non-intersecting paths of G is called a* set of non-intersecting paths.

For each non-leaf node ν, the rectangle bounded by the drawing of the corresponding vertices $b(\nu)$ and $t(\nu)$ and the drawing of the paths $(b(\nu), r(\nu), t(\nu))$ and $(b(\nu), l(\nu), t(\nu))$ is defined to be $\mathcal{D}(\nu)$.

Having at most three bends per edge is as good as it gets: Eades and Feng gave a family $C_n = (G_n, T_n)$ of examples for hierarchical clustered graphs such that in every c-planar orthogonal drawing with rectangular clusters, there are at least $O(n)$ edges that bent more than twice. G_n is a sequence

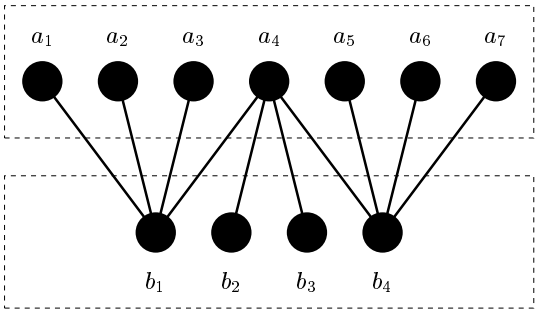


Fig. 8.11. Example for a lot of bends.

of n copies of the graph H shown in Figure 8.11 such that vertex a_7 of a previous copy of H serves as vertex a_1 of the next copy. It is partitioned into two clusters. Cluster A containing the a-vertices and cluster B containing the b-vertices. The embedding is as sketched in the figure. Each copy of H has at least one edge with more than two bends: At least one of the edges $\{a_4, b_1\}$ and $\{a_4, b_4\}$ has two or more bends in the cluster region of A. Suppose it's $\{a_4, b_1\}$. Then at least one of the edges $\{a_4, b_1\}$ and $\{a_1, b_1\}$ has three or more bends. Thus G_n has at least n such edges, but $10n + 1$ vertices.

8.3.3 Multilevel Visualization of Clustered Graphs

Eades and Feng (1996) show a way to represent both the *adjacency* and *inclusion* relations of a *clustered graph* in the same drawing. Here, the inclusion

relation is not only just drawn as simple regions containing the drawing of their corresponding vertices, but as a tree structure that also geometrically visualizes this relation. As graphs get larger and larger it is a common strategy to visualize them at multiple abstraction levels. If the graph has a recursive clustering it is a natural approach to take the clustering of the graph as abstraction levels, which provides the possibility to *zoom* in and out within the clustered structure of the graph. The method presented in Eades and Feng (1996) is a three dimensional representation of the clustered graph with each cluster level drawn at a different z-coordinate, and with the inclusion relation drawn as a tree in three dimensions. This kind of representation also keeps track of the abstractions from one level to the next.

Teminology. The *height* of a cluster v, denoted by $height(v)$, is defined as the depth of the subtree of T rooted at v.

For a clustered graph, its *view at level* i is a graph $G_i = (V_i, E_i)$ where V_i consists of the set of nodes of height i in T. There is an edge (μ, ν) in E_i if there exists an edge $(u, v) \in E$ where u belongs to cluster μ and v belongs to cluster ν; in other words, edge (μ, ν) is an abstraction of *all* edges between cluster μ and ν.

In a *plane drawing* of a clustered graph, the vertices are drawn as points and edges as curves in the plane as usual. Each cluster $\nu \in T$ is drawn as a simple closed region R that contains the drawing of $G(\nu)$, as defined in Definition 8.14. If a clustered graph has a c-planar representation (Definition 8.15), then it is *c-planar* (Figure 8.12).

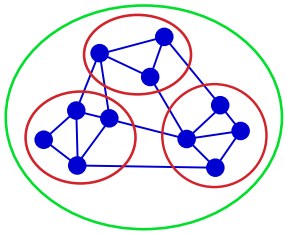


Fig. 8.12. A plane drawing of a clustered graph; the graph shown here is *c-planar*.

Multilevel Drawings. A *multilevel drawing* (Fig. 8.13) of a clustered graph $C = (G, T)$ consists of:

- a sequence of plane drawings of representations from the leaf level (level 0) to the root level of T, where the view of level i is drawn on the plane $z = i$.
- A three dimensional representation of T, with each node $\nu \in T$ of height i drawn as a point on the plane $z = i$, and within the region of ν in the drawing of a view of that level.

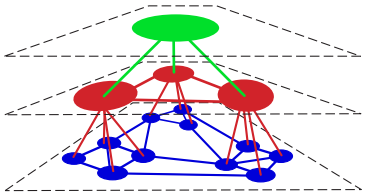


Fig. 8.13. Multilevel drawing.

For the plane drawing of a clustered graph several already presented automatic drawing algorithms can be used, such as the *straight-line convex* or *orthogonal rectangular* drawing methods (Section 8.3.1 and Section 8.3.2, see also Feng (1997)).

Here, the focus is only on the construction of the multilevel drawing which consists of two steps, the construction of view drawings for each level i, $i = 0, \ldots, height$ (root of T), and the drawing of the inclusion tree T.

1. View drawing for each level:

1. For each level i, $i = 0, \ldots, height$ (root of T), construct a plane drawing and translate it to the plane $z = i$, starting at the leaf level.
2. An edge (μ, ν) in level $i+1$ is an abstraction for all edges between cluster μ and ν in level i. Choose one of these edges (u, v) between cluster μ and ν as a representative edge, and derive the drawing of edge (μ, ν) in the view of level $i + 1$ from the drawing of edge (u, v) in the view of level i. In the two dimensional plane, cluster μ and ν are drawn as simple closed regions $R(\mu)$ and $R(\nu)$; the drawing of edge (u, v) intersects the boundaries of these regions at points x and y (Figure 8.14). To construct the drawing of edge (μ, ν) in the view of level $i + 1$, use the segment between x and y and translate it to the plane $z = i + 1$.

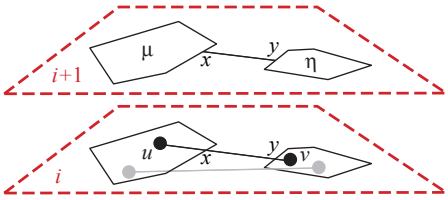


Fig. 8.14. Deriving a drawing for abstraction edges.

2. 3D drawing of the inclusion tree T:

General results on 3D drawings can be found in Chapter 7. To find a three dimensional drawing of the inclusion tree, every node $\mu \in T$ with height i has to be placed on the plane $z = i$ and it also must be positioned in the corresponding region $R(\mu)$. This is achieved as follows:

1. Compute the position of the nodes of the inclusion tree recursively:
   Level $i = 0$: (leaf level)
   Take the positions as computed in the plane drawing of level 0;
   Level $i > 0$:
   Let the position for node $\mu \in T$ be the average of all xy-coordinates of its children from level $i - 1$;
2. Route the inclusion edges as straight line segments between the corresponding nodes.

8.4 Hierarchical Representation of Compound Graphs

Sugiyama and Misue (1991) introduce an extension to the class of *clustered graphs*, the class of *compound digraphs*, and they also present an algorithm to produce an automatic hierarchical representation of compound digraphs. The main difference between these two graph classes is the use of the *inclusion relation*. A clustered graph is a graph with a partition of its vertex set into clusters. In the representation of a clustered graph, the cluster regions are drawn as simple closed regions that contain the drawing of all the vertices belonging to that cluster; the inclusion relation is restricted to these cluster regions, and there are no edges connecting them. In a compound digraph, the inclusion relation as well as the adjacency relation is defined on the *same* set of vertices.

Definition 8.19 (Compound Graph). *An* inclusion digraph *is a pair $D_c = (V, E)$ where E is a finite set of* inclusion edges *whose element $(u, v) \in E$ means that u includes v (Figure 8.15 (a)).*

An adjacency digraph *is a pair $D_a = (V, I)$ where F is a finite set of* adjacency edges *whose element $(u, v) \in E$ means that u is adjacent to v (Figure 8.15 (b)).*

A compound digraph *is defined as a triple $D = (V, E, I)$ obtained by compounding these two digraphs (Figure 8.15 (c)).*

In Sugiyama and Misue (1991), the inclusion digraph D_c is required to be a rooted tree and is also called the *inclusion tree* of D. The depth of a vertex $v \in V$ is the number of vertices on the path between v and the *root* of D_c and is denoted by $depth(v)$, with $depth(root) = 1$, where *root* denotes the root of D_c. The parent of a vertex $v \in D_c$ is denoted by $Parent(v)$.

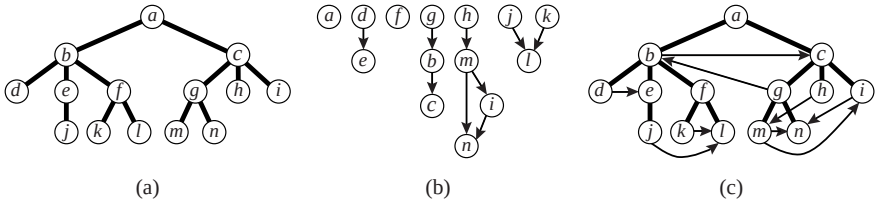


Fig. 8.15. (a) Inclusion tree D_c (b) Adjacency graph D_a (c) The compound digraph D obtained from (a) and (b).

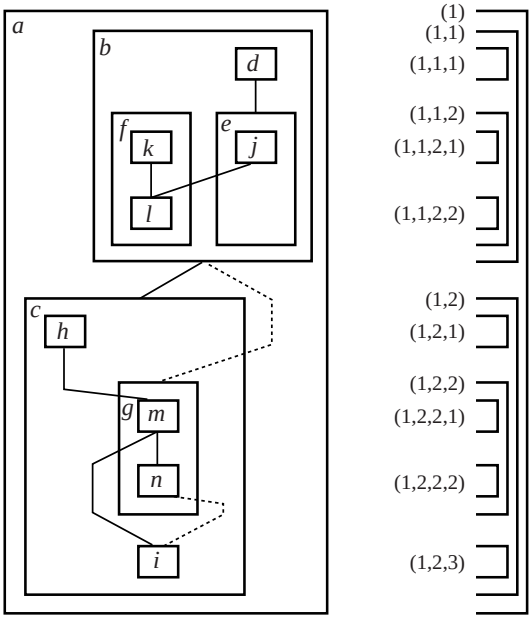


Fig. 8.16. The representation of a compound digraph from Figure 8.15 (c) and its compound levels.

Figure 8.16 shows a representation of the compound digraph and its compund levels of Figure 8.15 (c). The adjacency edges drawn with solid lines have downward orientation and edges drawn with broken lines upward. The vertices of the compound digraph are drawn as rectangles. The inclusion relation $(u, v) \in E$ is realized as the rectangle representing vertex u is inside the rectangle representing vertex v. The conventions for the drawing of compound digraphs are specified more precisely below.

8.4.1 Conventions

Drawing conventions.

C1 : *Vertex Shape:*
A vertex is drawn as a rectangle with horizontal and vertical sides.

C2 : *Inclusion:*
An inclusion edge (u, v) is drawn in such a way that the rectangle corresponding to u includes geometrically the rectangle corresponding to v.

C3 : *Hierarchy:*
Vertices are laid out hierarchically in terms of both inclusive and adjacent relations on parallel-nested horizontal bands, called *compound levels*.

C4 : *Down-Arrow:*
An adjacency edge (u, v) is drawn as a downward arrow with possible bends, originating from the bottom side of the rectangle corresponding to u and terminating on the top side of the rectangle corresponding to v.

Drawing Rules. To enhance the readability and the aesthetic of the drawing, the following objectives should be satisfied as much as possible.

R1 : *Closeness:*
Connected vertices are laid out as close as possible to each other.

R2 : *Edge Crossings:*
The number of crossings between adjacency edges is reduced as much as possible.

R3 : *Edge-Rectangle Crossings:*
The number of crossings between adjacency edges and the vertex rectangles is reduced as much as possible.

R4 : *Line-Straightness:*
One-span adjacency edges, i.e., edges between adjacent levels, are drawn as straight lines, whereas long span adjacency edges are drawn as polygonal lines with as few bends as possible.

R5 : *Balancing:*
Edges originating from and ending at a vertex rectangle are laid out in a balanced form.

The above rules specify topological and metrical layout properties; their priority is top-down.

8.4.2 The Layout Algorithm

The algorithm consists of the following four steps that are similar to those of an algorithm by Sugiyama et al. (1981) and Sugiyama (1987) for the layered representation of a general digraph. Due to the complex structure of a compound digraph, these steps must be modified and extended, in particular to display the inclusion relations. A brief description of the four steps follows, but the focus will be mainly on the ideas and the modifications that have to be done to the original Sugiyama algorithm (see Chapter 5). A detailed description of this algorithm is given in Sugiyama and Misue (1991).

Step 1: *Hierarchization*

Due to the two kinds of relations existing in a compound graph, this step is different to the one in Sugiyama's original algorithm for general graphs, so it will be explained in more detail.

A. Compound Level Assignment

In this step, the vertices of the compound digraph are assigned to *compound levels* to satisfy the drawing conventions. This level assignment places the vertices on parallel-nested horizontal bands. As shown in Figure 8.16, the compound levels can be expressed by the assignment of a sequence of positive integers to every vertex $v \in V$.

Let $\Sigma = 1, 2, 3, \ldots$ and $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$, and suppose that a lexicographical ordering is introduced for elements of Σ^+, e.g., $(1,1,2) < (1,2) < (1,2,1) < (1,2,2)$. Then the problem of assigning compound levels to the vertices of D is to find a mapping $c\text{-}level : V \to \Sigma^+$ satisfying the *Inclusion* and *Down-Arrow* conventions (C2, C4).

The *Inclusion* convention (C2) can be expressed as follows:

I1 : $\forall\, v \in V : c\text{-}level\,(v) \in \Sigma^{depth\,(v)}$
I2 : For any inclusion edge $(v, w) \in E : c\text{-}level\,(w) = append\,(c\text{-}level\,(v), s)$; $s \in \Sigma$ and *append* is a function that appends a component to a sequence.

The *Down-Arrow* convention (C4) is more complicated: For any adjacency edge $(v, w) \in F$ there is a unique path P from v to w in the inclusion tree D_c:

$$P : \quad (v = p_m, p_{m-1}, \ldots, p_1, t, q_1, q_2, \cdots, q_{n-1}, q_n = w),$$

where t is the *top vertex*, i.e., t has minimal depth. P originates from the rectangle of v, goes out across $p_{m-1}, \ldots, p_1$, passes t, goes in across $q_1, q_2, \cdots, q_{n-1}$, and terminates on w. To formulate the *Down-Arrow* convention, the order among each pair (p_i, q_i) of vertices that have the same depth for any adjacency edge $(v, w) \in F$ must be specified as follows:

D1 : if $depth\,(v) > depth\,(w)$ (or $m > n$),
(a) $c\text{-}level\,(p_i) \leq c\text{-}level\,(q_i),\ i = 1, \ldots, n-1$
(b) $c\text{-}level\,(p_n) < c\text{-}level\,(w)$

D2 : if $depth(v) \leq depth(w)$ (or $m \leq n$),

(a) $c\text{-}level(p_i) \leq c\text{-}level(q_i),\ i = 1, \ldots, m-1$

(b) $c\text{-}level(v) < c\text{-}level(q_m)$.

For example, in Figure 8.15, the path corresponding to adjacency edge (j, l) is j, e, b, f, l where b is the top vertex. Since $depth(j) = depth(l)$, we have $c\text{-}level(e) \leq c\text{-}level(f)$ and $c\text{-}level(j) < c\text{-}level(l)$ from D2.

A compound digraph has a *compound level assignment* if and only if there exists a mapping $c\text{-}level : V \to \Sigma^+$ satisfying I1, I2, D1, and D2.

B. Hierarchization Algorithm

A hierarchical map of the graph can not always be determined because the digraph might be cyclic. If there are cycles in D, some of the adjacency edges need to be reversed in order to obtain a hierarchization. Because the problem of finding this minimum set of feedback adjacency edges is $\mathcal{NP}$-complete (Garey and Johnson, 1991; Lempel and Cederbaum, 1966), heuristics are introduced for determining the edges that have to be reversed (see also Section 5.2).

In order to meet the requirements D1 and D2, every adjacency edge of D is replaced with one of the two following types of adjacency edges, $\rightarrow$ and $\Rightarrow$, which represent the relations $<$ and $\leq$, respectively, in D1 and D2. If edges between the same pair of vertices are duplicated during the replacement, reducing rules such as $\rightarrow = \rightarrow\ +\ \rightarrow$, $\Rightarrow = \Rightarrow\ +\ \Rightarrow$, and $\rightarrow = \rightarrow\ +\ \Rightarrow$ are applied to determine the resulting edge type. The graph derived by this edge replacement is called the *derived graph* of D. An adjacency edge in the derived graph is called an *original edge*, if $e \in F$, *derived edge* otherwise (Figure 8.17). Note that in the derived graph of D, every adjacency edge connects two vertices with identical depth. The derived graph of $D = (V, E, I)$ is denoted by $DD = (V, E, ID, type)$, where ID is the derived set of adjacency edges and $type : ID \to \{\rightarrow, \Rightarrow\}$.

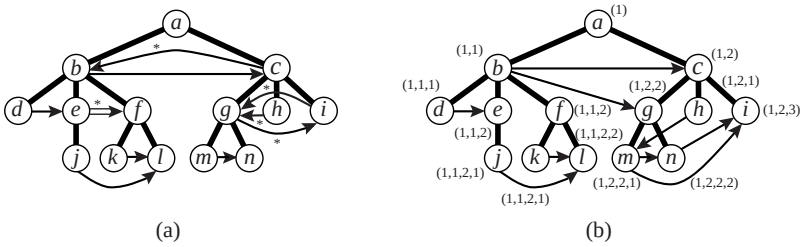


Fig. 8.17. The *derived graph* (a) and the *assigned compound digraph* (b) obtained from the compound graph of Figure 8.15 (c); edges in (a) marked with asterisk ($\star$) are derived edges.

Now, the *Down-Arrow* convention is established and the compound levels can be assigned to the vertices. If DD is not cycle-free, the cycles have to be resolved. To do this, the strongly connected components of DD are investigated, and the cycles are destroyed by either contracting strongly connected components into a so-called *proxy vertex* or by deleting some of the adjacency edges. This procedure leads to a cycle-free, hierarchical graph to which one finally can assign the compound levels. Of course, the deleted edges have to be put back in again, as well as the proxy vertices must be replaced again later on. All vertices of the component of DD, which have been contracted to a proxy vertex, are assigned the same compound level. Each adjacency edge (v, w) of the compound digraph $D = (V, E, I)$ is now checked, whether $\mathit{c\text{-}level}(v) < \mathit{c\text{-}level}(w)$. If this does not hold, the direction of that edge is reversed. The result is an *assigned compound digraph* $DA = (V, E, IA, \mathit{c\text{-}level})$ (Figure 8.17 (b)).

Step 2: *Normalization*

In an assigned compound digraph DA, an adjacency edge $(v, w) \in IA$ is said to be *proper* if and only if $\mathit{c\text{-}level}(\mathit{Parent}(v)) = \mathit{c\text{-}level}(\mathit{Parent}(w))$ and $\mathit{tail}(\mathit{c\text{-}level}(v)) = \mathit{tail}(\mathit{c\text{-}level}(w)) - 1$, where *tail* is a function, that returns the last number of the *c-level*-string as an integer. The assigned compound digraph DA is now transformed into a *proper* compound digraph by replacing every non-proper adjacency edge with appropriate dummy vertices, dummy inclusion edges and dummy proper adjacency edges (cf. Sugiyama and Misue (1991) for more detail).

Step 3: *Vertex Ordering*

The idea of this step is similar as in Sugiyama's original algorithm for general graphs. The horizontal order of the vertices per level is determined by permuting their order on each level in such a way that the drawing rules *Closeness*, *Edge Crossings*, and *Edge-Rectangle Crossings* (R1 – R3) are satisfied as much as possible. The problem of minimizing edge crossings is $\mathcal{NP}$-complete even for only two levels (Garey and Johnson, 1991). Minimizing of edge-rectangle crossings, which is equivalent to the *linear arrangement* problem, is also $\mathcal{NP}$-complete (Garey and Johnson, 1991). Hence, heuristics, i.e., *barycentric ordering* as in Sugiyama's algorithm, are used to accomplish these tasks. In a compound digraph there exist also *local hierarchies* due to the inclusion relation, so vertex ordering must also be applied to these local hierarchies, i.e., subtrees of D_c. This step leads to an *ordered compound digraph*.

Step 4: *Metrical Layout*

In this last step, the positions of vertices (i.e., horizontal and vertical positions, widths and heights of rectangles) are determined by attaining the *Closeness*, *Line-Straightness*, and *Balancing* rules (R1, R4, R5) as much as possible. This problem can be expressed as a quadratic programming problem; a heuristic called the *priority layout method* is also developed to solve the

problem. Once the vertex positions are determined, a routing for the edges can easily be achieved. The orientation of the reversed edges is changed back again, and all inserted dummy vertices and dummy edges are deleted and their corresponding originals are rearranged. This step finally leads to an automatic drawing of the original compound digraph.

8.5 Force-Directed Methods for Clustered Graphs

Force-directed graph drawing methods (cf. Chapter 4) can also be adopted to support and show the structure of a clustered graph. In the following sections we will see different ways of adaptation with different design goals.

One possibility to receive a more structured layout for clustered graphs is to decide between different spring forces. In Sections 8.5.2 and 8.5.3 we show two different approaches for an expanded force model for clustered graphs.

8.5.1 Inserting Dummy Vertices

The easiest way to achieve clustering is to insert *dummy vertices* as follows (Figure 8.18):

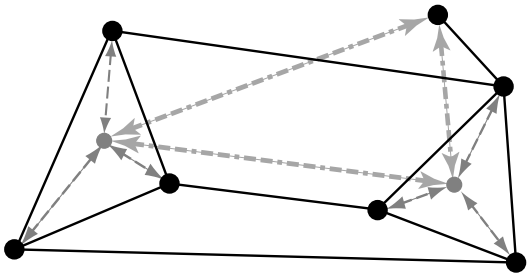


Fig. 8.18. Realizing clustering constraints in a force-directed approach by inserting dummy *attractors* (shaded vertices) in each cluster.

1. Let $G = (V, E)$ be a graph with a partition $(C_1, C_2, \ldots, C_k)$ on the vertex set V. For each C_i, $1 \leq i \leq k$, add a dummy *attractor* vertex c_i to the graph.
2. Add *attractive* forces between an attractor c_i and each vertex of the corresponding cluster C_i.
3. Add *repulsive* forces between pairs of attractors and between attractors and vertices not belonging to any cluster (i.e., if $\bigcup_{i=1}^{k} C_i \subset V$).

In this approach, no new forces have to be added. After inserting these attractors the vertices within a cluster will be closer to each other than before, and the distance between the clusters will grow.

8.5.2 Interactive Clustering

Huang and Eades (1998a) describe an animated interactive system for clustering and navigating huge graphs, called DA-TU, where they use the following expanded force model consisting of three different spring forces (Figure 8.19):

- *internal-spring*
  A spring force between a pair of vertices that belong to the same cluster.
- *external-spring*
  A spring force between a pair of vertices that belong to different clusters.
- *virtual-spring*
  In each cluster there is a virtual vertex (black vertices in Figure 8.19) that is connected to all vertices belonging to the same cluster by virtual edges; this is a similar approach to the concept of *attractors* described above. A virtual spring force exists between a vertex and a virtual vertex along a virtual edge.

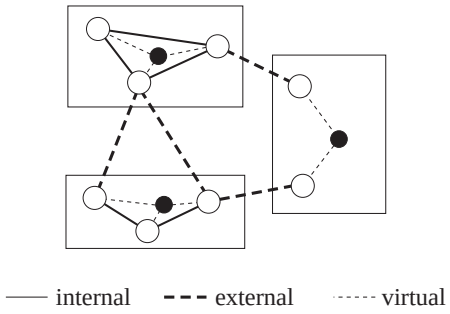


Fig. 8.19. Expanded spring model.

Additionally there is a gravitational repulsion force between each pair of vertices. All forces are applied additively to each vertex.

Some of the features of the DA-TU system are worth to be mentioned here because they show a possible application for clustering:

- The user can interactively change the graph.
- The user can interactively change the clustering of the graph.
- All transitions from one state of the graph to another are animated.
- Clusters can be interactively *contracted* and *expanded*, respectively. This is in particular useful for large graphs that do not fit on the screen or are too large to comprehend, so the clustered structure is used to navigate through the graph. If some of the clusters are contracted, DA-TU draws a so-called *abridgement* of the given graph.

Definition 8.20 (Abridgement, Ancestor Tree). *A clustered graph* $C' = (G', T')$ *is an* abridgement *of the clustered graph* $C = (G, T)$ *if* T' *is an* ancestor tree *of* T *with respect to a set* U *of nodes of* T *and there is an edge between two distinct nodes* u *and* v *of* G' *if and only if there is an edge in* G *between a descendant of* u *and a descendant of* v.

In other words, the ancestor tree T' is a subtree of T consisting of all nodes and edges on paths between elements of U and the root. Figure 8.20 shows such an ancestor tree for the set U of black nodes as the shaded area of the original inclusion tree.

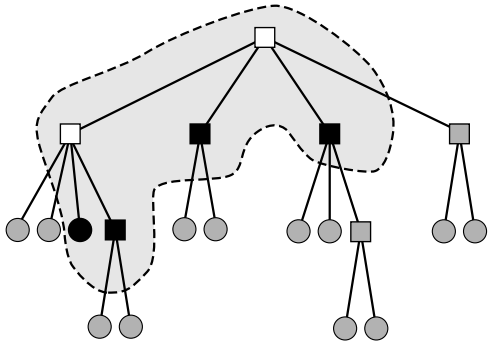


Fig. 8.20. The shaded area is the ancestor tree of the set of black nodes.

8.5.3 Meta Layouts

A similar approach is taken by Wang and Miyamoto (1995). They also use three forces but in a slightly different way. Instead of inserting virtual vertices as attractors in each cluster they use the concept of a *meta-graph*.

First, the edge set of the given graph is divided into *intra-edges*, i.e., edges between vertices belonging to the same cluster, and *inter-edges*, i.e., edges between vertices belonging to different clusters.

The forces in the force-directed placement are also divided into two categories:

- *intra-force*
  A spring force between a pair of vertices that belong to the same cluster.
- *inter-force*
  A spring force between a pair of vertices that belong to different clusters.

A force-directed placement is constructed by applying the intra- and inter-forces. An undirected graph G_{meta} is then constructed by collapsing the clusters of G into *meta-vertices* and transforming inter-edges of G existing between a pair of clusters into one *meta-edge* each. This sounds similar to the concept of a *quotient graph* (Definition 8.3), but goes further than that. A layout for G_{meta} is called *meta-layout* of G and can be obtained from the force-directed placement where the dimensions and center of each meta-vertex are set to the dimensions and center of the underlying subgraph, respectively (Figure 8.21). To calculate the forces between the meta-vertices, an improved force-directed placement is used that takes the different vertex-sizes into account (Wang and Miyamoto, 1995). The net force on a meta-vertex is defined as the *meta-force* on all vertices contained in the cluster represented by that meta-vertex.

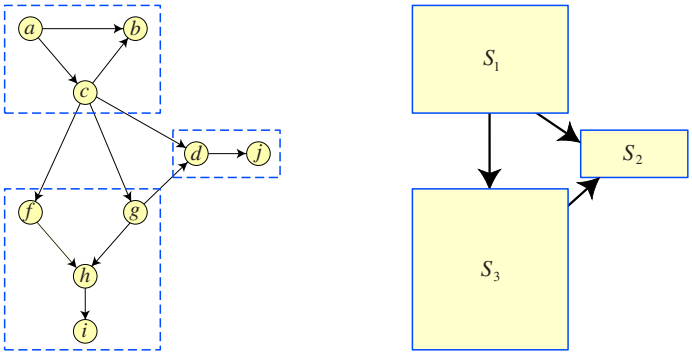


Fig. 8.21. Meta-graph and meta-layout.

In Figure 8.21 a force-directed layout and a partition are given on the left side, the corresponding meta-layout is shown on the right. The forces that are applied to vertex c are:

- The intra-force on c is the sum of forces between c, a and c, b
- The inter-force on c is the sum of forces between c and the vertices of subgraphs S_2 and S_3
- The meta-force on c is the net force on meta-vertex S_1 in the meta layout.

To finally compute a drawing of the clustered graph, Wang and Miyamoto (1995) propose a divide-and-conquer approach.

Divide-and-Conquer Drawing Approach. A divide-and-conquer drawing algorithm would draw a clustered graph in the following three steps:

1. divide the graph into subgraphs (cluster);

2. draw the subgraphs;
3. compose the subgraph layouts together to form the resulting layout.

The problem with this approach is that inter-edges are not taken into account which may result in a drawing with many crossings between inter-edges or long inter-edges.

In Wang and Miyamoto (1995), the last two steps of this divide-and-conquer approach are combined within one force-directed placement algorithm by using the following composite force F_{comp} to position a vertex:

$$F_{comp} = F_{intra} + S(t)F_{inter} + \big(1 - S(t)\big)F_{meta}$$

where F_{intra}, F_{inter}, F_{meta} are the intra-, inter- and meta-force on a vertex, respectively, and $S(t) \in [0,1]$ is a function of layout time t such that $S(t)$ decreases as t increases after a threshold t' and reaches 0 at another threshold $t'' > t'$.

By applying this composite force, the force-directed placement can be divided into three phases:

1. *Between time* 0 *and time* t'*:* $S(t) = 1 \implies F_{comp} = F_{intra} + F_{inter}$.
   The force-directed placement leads to a layout with uniform edge lengths and a small number of edge crossings, as shown in Figure 8.22 (a).
2. *Between time* t' *and time* t''*:* $S(t)$ decreases:
   The strength of the inter-forces is reduced while the strength of the meta-forces is increased at the same time.
3. *at time* t''*:* $S(t) = 0 \implies F_{comp} = F_{intra} + F_{meta}$.
   Inter-forces do not count anymore; the intra-forces keep the vertices contained in the same cluster close together while the meta-forces fix the final positions of the clusters and eliminate possible overlaps between clusters. The resulting structured layout is shown in Figure 8.22 (b).

Wang and Miyamoto (1995) also present a way to add layout constraints to their force-directed placement algorithm by integrating a constraint solver.

Integration of a Constraint Solver. Layout constraints of the three following types can occur:

- *absolute constraints*, to fix an absolute vertex position
- *relative constraints*, to constrain the position of a vertex in relation to other vertices
- *cluster constraints*, to cluster several vertices into a subgraph that can be processed as a whole.

While trying to solve given constraints, some vertices may block others from reaching their optimal positions calculated by the force-directed placement algorithm. This may lead to a poor layout.

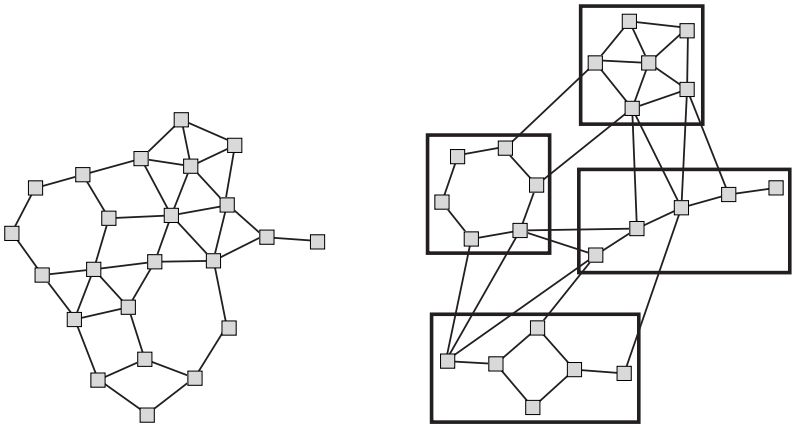


Fig. 8.22. Layout created at time t' (*left*) and resulting layout (*right*).

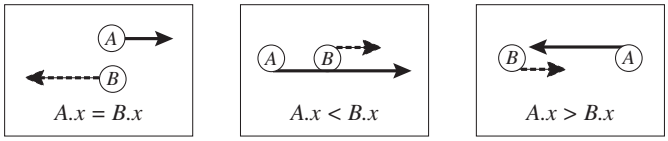


Fig. 8.23. Examples for constraints that become barriers.

If a constraint for two vertices A and B is given that prevents vertex A from reaching its optimal position, then vertex B is called a *barrier* for vertex A. Figure 8.23 shows several examples of barriers; the arrows indicate the direction and the strength of the forces calculated by the layout algorithm, the corresponding constraint shown below would be violated if these movements would be performed. In Figure 8.23 (a) no movement of either vertex is possible without violating the corresponding constraint. In the other two examples, the vertices could at least be partially moved until one of the vertices becomes a barrier for the other.

To avoid barriers while at the same time improving the layout, the vertices could be moved together without changing their relative positions. This is done by introducing *rigid sticks* to represent constraints in the force-directed placement. If vertex v_1 becomes a barrier for vertex v_2, a rigid stick is introduced between them so that they have to move like one rigid object. The movements of v_1 and v_2 are determined by the weighted average of the forces that are working on them:

$$f = \frac{w_1 f_1 + w_2 f_2}{w_1 + w_2}$$

where f is the new resulting force on v_1 and v_2, f_1 and f_2 are the old forces on v_1 and v_2, respectively, and w_1 and w_2 are weights of v_1 and v_2, respectively.

The layout algorithm and the solver cooperate to solve the given constraints as much as possible while at the same time keeping a good layout resolution. This cooperation works in an iteration of the following four steps:

Step 1: Calculate forces;

Step 2: Introduce sticks and distribute forces;

Step 3: Calculate new positions;

Step 4: Satisfy constraints.

Steps 1 and 3 are performed by the layout algorithm, whereas the other two steps are performed by the solver. Figure 8.24 shows an example of applying constraints to the graph of Figure 8.22.

```
d3.y = d5.y

abs(a4.x - a2.x) <= 120
abs(a1.x - a5.x) <= 70
a5.y = a1.y = a2.y - 64
a4.y = a3.y = a2.y

c2.y = c5.y = c4.y = c1.y
```

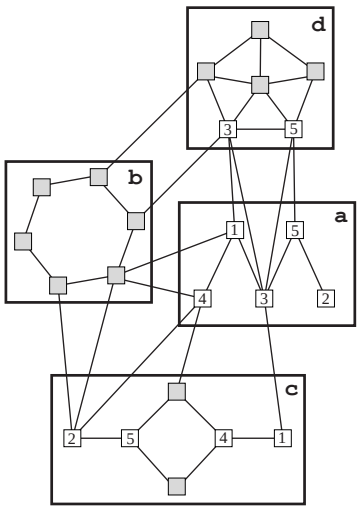


Fig. 8.24. Resulting layout of the graph of Figure 8.22 after adding the constraints on the left side.

8.6 Online Graph Drawing of Huge Graphs – A Case Study

Traditional graph drawing algorithms assume that the given graph can be laid out in a readable and understandable way on the screen or on paper. But there are important situations where this assumption does not hold. Suppose for example the graph displaying parts of the WWW or graphs arising in information retrieval. These graphs can be very large and there is no way to fit them in a readable way on a display medium.

Most graph drawing systems approach the layout of huge graphs in the following way:

1. Layout the graph on a virtual and very large page.
2. Provide a smaller window with scroll bars to show the part of interest, and to allow the user to navigate through the graph.

However, some problems are involved in this approach:

- The whole graph may not be known, e.g., in distributed systems, where a local vertex only knows part of the graph.
- To explore the graph, the user can only move geometrically through the graph by the means of the scroll bars. But the user might want to explore the graph in a logical way, in particular, if the graph contains relational data or hyperlinks. Moreover, long edges that do not fit on the screen are hard to follow. A more user-oriented approach would be better; the user should be able to control the logical content of the display.
- There is no *mental map* (Eades et al., 1991) that helps the user to keep track of his exploration so far. Even worse, the user can not see the whole graph and might get lost in empty areas.
- Besides that, it costs a lot of memory to store and display the large virtual screen.

To deal with these problems, several techniques have been proposed (see Sarkar and Brown (1994); Eades et al. (1991); Nielsen (1990); Mukherjea et al. (1994); Robertson et al. (1993)). For example, in Sarkar and Brown (1994) the *fish-eye* view technique is described where a detailed picture of a subgraph is shown along with the so called context of that subgraph. This kind of view provides the user with more information about the position of the subgraph within the whole graph. Another approach are three-dimensional methods, such as *cone trees* (Robertson et al., 1993) which lead to an increase in density of information on the screen.

These techniques work effectively for graphs of moderately large size, but they can not be applied when the graph is not completely known. Moreover, they still predefine the geometry of the graph.

The aim of *Online Graph Drawing* is the visualization of huge graphs which may be partially unknown. At any time, a tiny but non-empty subgraph called the *logical frame* is known and displayed on the screen. The user can explore the huge graph by changing the logical frame.

The layout of such a logical frame has to satisfy the usual aesthetic criteria for drawing graphs, e.g., minimization of edge crossings and uniform vertex distribution. Additionally, the transition from one picture of a logical frame to the next should preserve the user's *mental map* (Eades et al., 1991), i.e., successive drawings should not differ much, so that the user can easily follow the change in the drawing and does not loose orientation in a completely different layout when changing the focus.

Eades et al. (1997b) describe a model of Online Graph Drawing as well as an instantiation of that model in a system for *Online Force-Directed Animated Visualization* (OFDAV) for assisting web navigation (see also Eades et al. (1997a)). An interesting part of that model is a new force-directed drawing algorithm, that can be used to produce a continuous sequence of layouts according to the above mentioned criteria.

In OFDAV, the view of the user is focused on a small subgraph, the logical frame, that is defined by a *focus vertex* v. A force-directed graph drawing algorithm is used to draw this subgraph as well as its logical neighborhood. The user can change focus by selecting another vertex within the displayed frame which then becomes the new focus node, and the view changes according to this selection. Multiple animation steps are used to guide the user through the change of view and to preserve the mental map. A linear *history* is also kept by lining up a certain number of previously visited focus vertices.

The focus vertices together with their neighborhoods form a *clustering* of the graph.

The Online Graph Model. To explore a huge, partially unknown graph $G = (V, E)$, a sequence of *logical frames* $F_1 = (G_1, Q_1), F_2 = (G_2, Q_2), \ldots$ is used (Figure 8.25):

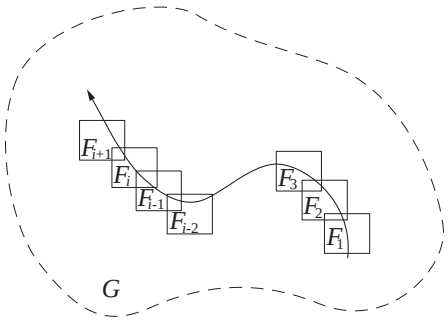


Fig. 8.25. The path of exploration of a huge graph G by a sequence of logical frames.

Each logical frame $F_i = (G_i, Q_i)$ consists of a connected subgraph $G_i = (V_i, E_i)$ of G and a queue Q_i of *focus vertices*. Successive frames differ only by a few vertices. The sequence of logical frames represents the sequence of subgraphs that are viewed by the user of the system and is determined by the interaction of the user who can change the focus and thereby decides which new logical frame has to be displayed. To define the logical frame more precisely, we need to explain the concept of neighborhood.

Suppose that $G = (V, E)$ is a graph, $v \in V$, and d is a non-negative integer. Then, the *distance-d neighborhood* $N_d(v)$ of v is the subgraph of G

induced by the set of vertices whose graph-theoretic distance from v is at most d; note that $v \in N_d(v)$. In OFDAV only distance-1 neighborhoods are used, so we write $N(v)$ instead of $N_1(v)$ in the following, and call it the *neighborhood* of v.

Definition 8.21 (Logical Frame, Focus Vertex). *Given a queue $Q = (v_1, v_2, \ldots, v_s)$ of vertices of graph $G = (V, E)$, the subgraph of G induced by the union of $N(v_1), N(v_2), \ldots, N(v_s)$ is called a* logical frame *$F = (G', Q)$ (Figure 8.26 (a)), with $G' = (V', E')$ and*

$$V' = \bigcup_{i=1}^{s} N(v_i) \qquad E' = \{(u, v) \in E \mid u, v \in V'\}.$$

The vertices of the queue Q are the focus vertices *of the logical frame F.*

Clustering. Suppose that $Q = (v_1, v_2, \ldots, v_s)$ is the queue of focus vertices in G. Each neighborhood $N(v)$, for $v \in Q$, can be divided into two parts, the *common part* $C(v)$, and the *local part* $P(v)$, defined as follows:

- $C(v)$ is the part of the neighborhood $N(v)$ that also occurs in the neighborhood of other focus vertices $v' \neq v$, that is:

$$C(v_j) = \bigcup_{i=1, i \neq j}^{s} N(v_j) \cap N(v_i).$$

- $P(v)$ is the part of the neighborhood $N(v)$ that does not occur in the neighborhood of any other focus vertex $v' \neq v$ (Figure 8.26 (b)), that is:

$$P(v_j) = N(v_j) - \bigcup_{i=1, i \neq j}^{s} N(v_i).$$

The user can explore the graph by changing the focus vertex, and this exploration is visualized by a sequence of logical frames. In practice, only a small number of vertices can be displayed on the screen at a time, in particular if the vertices are labeled, e.g., by the names of the html-pages as in OFDAV. Here, a global constant B is introduced as an upper bound for the length of the focus queue Q. For www-graphs, small values of B between 7 and 10 ensure that there are about 20 to 60 vertices on the screen at a time.

The transition from one logical frame to the next is obtained by adding the new focus vertex with its local neighborhood. If the length of Q was already B before, then the least recently used focus vertex and its local neighborhood are deleted (FIFO policy).

The local neighborhoods of the focus vertices in each frame can be viewed as the *clusters* of this frame. This will become clearer in the next section where the force model is explained.

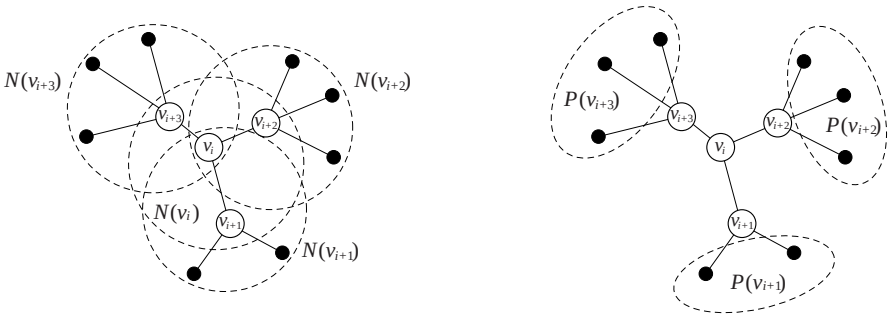


Fig. 8.26. Neighborhood of the focus vertices (*left)* local part neighborhood (*right*).

The Force Model. The force model is based on Eades (1984) and consists of a combination of Hooke's law springs and Newtonian gravitational forces. In order to address the specific criteria of this online drawing approach, extra Newtonian gravitational forces among the neighborhoods, $N(v_i), N(v_{i+1}), \ldots, N(v_{i+B})$, of the focus vertices are added. These forces are used to separate the neighborhoods so that the user can visually identify the changes induced by changing the focus. This leads to a clustering of the displayed subgraph.

The total force applied to a vertex v of a logical frame $F_i = (G_i, Q_i)$, with $G_i = (V_i, E_i)$ is

$$f = \sum_{u \in N(v)} f_{uv} + \sum_{u \in V_i} g_{uv} + \sum_{u \in Q_i} h_{uv}$$

where f_{uv} is the force exerted on v by the spring between u and v, and g_{uv} and h_{uv} are the gravitational repulsions exerted on v by one of the other vertices u in F_i (Figure 8.27).

The details of the modified force-directed drawing algorithm are given in Eades et al. (1997b). Here, we will only explain the idea of this approach.

The modified force model aims to satisfy the following four aesthetic criteria:

1. The spring force f_{uv} between adjacent vertices is aimed to ensure that the distance between vertices u and v is approximately equal to the zero energy spring length.
2. The gravitational force g_{uv} ensures that the vertices are not too close together and distributed evenly.
3. The extra gravitational force h_{uv} aims to minimize the overlaps among the (local) neighborhoods within a logical frame. This also ensures that the next vertices that have to disappear are placed close together which makes the identifying of the deleted objects easier for the user.

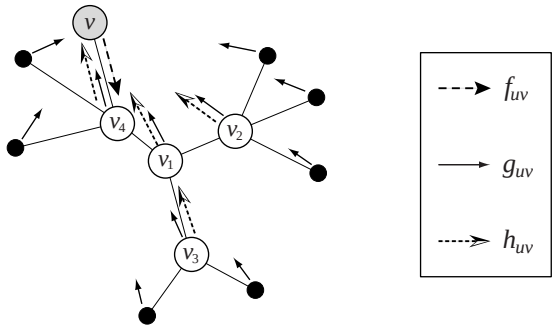


Fig. 8.27. The modified force model applied to the logical frame F_i.

4. h_{uv} also aims to keep the layout of the queue of focus vertices close to a straight line; new vertices appear in one end of that line while old vertices disappear at the other end. This helps the user in understanding the direction of exploration of the huge graph.

8.7 Summary

For a graph that does not have a natural cluster-structure it is not at first glance clear what a good clustering strategy is. Often, the vertices are gathered together with respect to graph connectivity. Two main heuristics to do this are introduced in Section 8.2. One is to integrate cut size and cluster size balance within a single objective function, like the ratio cut partition, and to optimize them with a suitable procedure. Another one makes use of the eigenvalues of the Laplacian matrix of the graph. Besides connectivity, other graph properties like similarity of neighborhoods can also be of interest.

Once a graph has a cluster-structure, the question arises, how to make this structure visible. If we want to draw a really large graph, it seems to be a good method to draw only the quotient graph. But sometimes one is interested to also see what happens within a cluster. In Section 8.3 and Section 8.4 two methods are presented that draw clusters as shapes which include the corresponding vertices. Additionally, crossings between edges and borders of shapes are avoided. Especially in the planar case in Section 8.3, such a crossing is only allowed if one endpoint of the edge is within the corresponding cluster and the other one is outside of it.

Another way to show the cluster-structure is to draw vertices that belong to the same cluster closer together than such that are in different clusters. Using force-directed methods, one can achieve this by adding a dummy vertex to each cluster or by regarding clusters as big vertices. In this case the force-function must respect the different size of the vertices.

Bibliography

Abbott, K. R., and Sarin, S. K. (1994). Experiences with workflow management: Issues for the next generation. In *Proceedings of ACM Conference on Computer-Supported Cooperative Work, Workflow and Information Sharing (CSCW'94)*, pages 113–120.

Agarwal, P. K., and Erickson, J. (1997). Geometric range searching and its relatives. Technical Report CS 1997-11, Department of Computer Science, Duke.

Agarwal, P. K., van Kreveld, M., and Suri, S. (1998). Label placement by maximum independent set in rectangles. *Computational Geometry: Theory and Applications*, 11(3-4):209–218.

Aho, A., Hopcroft, J., and Ullman, J. (1974). *The Design and Analysis of Computer Algorithms*. Addison-Wesley.

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall.

Alpert, C. J., and Kahng, A. B. (1995). Recent directions in netlist partitioning: a survey. *INTEGRATION, the VLSI Journal*, 19:1–81.

Amtrup, H. H. J., and Jost, U. (1996). What's in a word graph? Evaluation and enhancement of word lattices. Technical Report Verbmobil-Report 186, University Hamburg, Germany.

Andreev, E. M. (1970a). On convex polyhedra in Lobacevskii spaces. *Math. USSR-Sb.*, 10:413–440.

Andreev, E. M. (1970b). On convex polyhedra of finite volume in Lobacevskii space. *Math. USSR-Sb.*, 12:255–259.

Auslander, L., and Parter, S. V. (1961). On imbedding graphs in the plane. *Journal of Mathematics and Mechanics*, 10(3):517–523.

Baker, K. A., Fishburn, P. C., and Roberts, F. S. (1971). Partial orders of dimension 2. *Networks*, 2:11–28.

Batagelj, V., Kerzic, D., and Pisanski, T. (1992). Automatic clustering of languages. *Computational Linguistics*, 18(3):339–352.

Berge, C. (1993). *Graphs*. North Holland, Amsterdam, 3rd edition.

Berger, B., and Shor, P. (1990). Approximation algorithms for the maximum acyclic subgraph problem. In *Proceedings of the 1st ACM-SIAM Symposium on Discrete Algorithms (SODA'90)*, pages 236–243.

Bertolazzi, P., Cohen, R. F., Di Battista, G., Tamassia, R., and Tollis, I. G. (1994a). How to draw a series-parallel digraph. *International Journal of Computational Geometry and Applications*, 4:385–402.

Bertolazzi, P., Di Battista, G., and Didimo, W. (1997). Computing orthogonal drawings with the minimum number of bends. In *Proceedings of the 5th Workshop on Algorithms and Data Structures (WADS'97)*, Spinger LNCS 1272, pages 331–344.

Bertolazzi, P., Di Battista, G., Liotta, G., and Mannino, C. (1994b). Upward drawings of triconnected digraphs. *Algorithmica*, 6(12):476–497.

Bertolazzi, P., Di Battista, G., Mannino, C., and Tamassia, R. (1993). Optimal upward planarity testing of single-source digraphs. In *Proceedings of the 1st European Symposium on Algorithms (ESA'93)*, Springer LNCS 726, pages 37–48.

Bertsekas, D. P. (1998). *Network Optimization: Continuous and Discrete Models*. Athena Scientific.

Biedl, T., Shermer, T., Whitesides, S., and Wismath, S. (1999). Bounds for orthogonal 3D graph drawing. *Journal of Graph Algorithms and Applications*, 3(4):63–79.

Biedl, T. C. (1997). *Orthogonal Graph Visualization: The Three-Phase Method with Applications*. PhD thesis, Rutgers University.

Biedl, T. C. (1998). Three approaches to 3D-orthogonal box-drawings. In *Proceedings of the 6th International Symposium on Graph Drawing (GD'98)*. Springer LNCS 1547, pages 30–43.

Biedl, T. C., and Kant, G. (1994). A better heuristic for orthogonal graph drawing. In *Proceedings of the 2nd European Symposium on Algorithms (ESA'94)*, Springer LNCS 855, pages 24–35.

Biedl, T. C., and Kaufmann, M. (1997). Area-efficient static and incremental graph drawings. In *Proceedings of the 5th European Symposium on Algorithms (ESA'97)*, Springer LNCS 1284, pages 37–52.

Biedl, T. C., Madden, B. P., and Tollis, I. G. (1997a). The three-phase method: A unified approach to orthogonal graph drawing. In *Proceedings of the 5th International Symposium on Graph Drawing (GD'97)*. Springer LNCS 1353, pages 391–402.

Biedl, T. C., Shermer, T., Whitesides, S., and Wismath, S. (1997b). Orthogonal 3D graph drawing. In *Proceedings of the 5th International Symposium on Graph Drawing (GD'97)*. Springer LNCS 1353, pages 76–86.

Blythe, J., McGrath, C., and Krackhardt, D. (1996). The effect of graph layout on inference from social network data. In *Proceedings of the 3rd International Symposiom on Graph Drawing (GD'95)*. Springer LNCS 1027, pages 40–51.

Böhringer, K.-F., and Paulisch, F. N. (1990). Using constraints to achieve stability in automatic graph layout algorithms. In *Proceedings of the ACM Human Factors in Computing Systems Conference (CHI'90)*, pages 43–51.

Booth, K. S., and Lueker, G. S. (1976). Testing for the consecutive ones property, interval graphs and graph planarity using PQ-tree algorithms. *Journal of Computer and System Sciences*, 13:335–379.

Borgida, A., Brachman, R., McGuinness, D., and Resnick, L. (1989). CLASSIC: A structural data model for objects. In *Proceedings of the 1989 ACM-SIGMOD International Conference on Management of Data*, pages 59–67.

Bose, P., Gomez, F., Ramos, P., and Toussaint, G. (1996). Drawings nice projections of objects in space. In *Proceedings of the 3rd International Symposium on Graph Drawing (GD'95)*, Springer LNCS 1027, pages 52–63.

Brandenburg, F. J., Himsolt, M., and Rohrer, C. (1996). An experimental comparison of force-directed and randomized graph drawing algorithms. In *Proceedings of the 3rd International Symposium on Graph Drawing (GD'95)*, Springer LNCS 1027, pages 76–87.

Brandes, U. (1999). *Layout of Graph Visualizations*. PhD thesis, University of Konstanz. `http://www.ub.uni-konstanz/kops/volltexte/1999/255/`.

Brandes, U., Kenis, P., Raab, J., Schneider, V., and Wagner, D. (1999). Explorations into the visualization of policy networks. *Journal of Theoretical Politics*, 11(1):75–106.

Brandes, U., and Wagner, D. (1997). A Bayesian paradigm for dynamic graph layout. In *Proceedings of the 5th International Symposium on Graph Drawing (GD'97)*, Springer LNCS 1353, pages 236–247.

Brandes, U., and Wagner, D. (1998a). Dynamic grid embedding with few bends and changes. In *Proceedings of the 9th Annual International Symposium on Algorithms and Computation (ISAAC'98)*, Springer LNCS 1533, pages 89–98.

Brandes, U., and Wagner, D. (1998b). Using graph layout to visualize train interconnection data. In *Proceedings of the 6th International Symposium on Graph Drawing (GD'98)*, Springer LNCS 1547, pages 44–56.

Branke, J., Bucher, F., and Schmeck, H. (1997). A genetic algorithm for drawing undirected graphs. In *Proceedings of the 3rd Nordic Workshop on Genetic Algorithms and their Applications*, pages 193–206.

Bridgeman, S., Di Battista, G., Didimo, W., Liotta, G., Tamassia, R., and Vismara, L. (2000). Turn-regularity and optimal area drawings for orthogonal representations. *Computational Geometry: Theory and Applications*, 16(1):53–93.

Bridgeman, S., Fanto, J., Garg, A., Tamassia, R., and Vismara, L. (1997). InteractiveGiotto: An algorithm for interactive orthogonal graph drawing. In *Proceedings of the 5th International Symposium on Graph Drawing (GD'97)*, Springer LNCS 1353, pages 303–308.

Bridgeman, S., and Tamassia, R. (1998). Difference metrics for interactive orthogonal graph drawing algorithms. In *Proceedings of the 6th International Symposium on Graph Drawing (GD'98)*, Springer LNCS 1457, pages 57–71.

Bruß, I., and Frick, A. (1996). Fast interactive 3-D graph visualization. In *Proceedings of the 3rd International Symposium on Graph Drawing (GD'95)*, Springer LNCS 1027, pages 99–110.

Cai, J., Han, X., and Tarjan, R. E. (1993). An $O(m \log n)$-time algorithm for the maximal planar subgraph problem. *SIAM Journal on Computing*, 22:1142–1162.

Carpano, M. J. (1980b). Automatic display of hierarchized graphs for computer aided decision analysis. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-10(11):705–715.

Catarci, T. (1995). The assignment heuristic for crossing reduction. *IEEE Trans. Syst. Man Cybern.*, 25(3):515–521.

Chaiken, S., and Kleitman, D. J. (1978). Matrix tree theorems. *Journal of Combinatorial Theory, Series A*, 24:377–381.

Chan, T., Goodrich, M. T., Kosaraju, S. R., and Tamassia, R. (1996). Optimizing area and aspect ration in straight-line orthogonal tree drawings. In *Proceedings of the 4th International Symposium on Graph Drawing (GD'96)*. Springer LNCS 1190, pages 63–75.

Chan, T. M. (1999). A near-linear area bound for drawing binary trees. In *Proceedings of the 10th ACM-SIAM Symposium on Discrete Algorithms (SODA'99)*, pages 161–168.

Chiba, N., Nishizeki, T., Abe, S., and Ozawa, T. (1985). A linear time algorithm for embedding planar graphs using PQ-trees. *Journal of Computer and System Sciences*, 30:54–76.

Christensen, J., Friedman, S., Marks, J., and Shieber, S. (1997). Empirical testing of algorithms for variable-sized label placement. In *Proceedings of the 13th Annual ACM Symposium on Computational Geometry*, pages 415–417.

Christensen, J., Marks, J., and Shieber, S. (1993). Algorithms for cartographic label placement. In *Proceedings of the American Congress on Surveying and Mapping 1*, pages 75–89.

Christensen, J., Marks, J., and Shieber, S. (1995). An empirical study of algorithms for point-feature label placement. *ACM Transactions on Graphics*, 14(3):203–232.

Chrobak, M., and Kant, G. (1997). Convex grid drawings of 3-connected planar graphs. *International Journal of Computational Geometry and Applications*, 7(3):211–224.

Chvátal, V. (1983a). *Linear Programming*. W. H. Freeman.

Closson, M., Everett, H., Gartshore, S., and Wismath, S. (1998). Arrangepak, orthopak and vispak 2.0. Technical Report TR-CS-98, University of Lethbridge.

Coffman, E. G., and Graham, R. L. (1972). Optimal scheduling for two processor systems. *Acta Informatica*, 1:200–213.

Cohen, J. D. (1997). Drawing graphs to convey proximity: An incremental arrangement method. *ACM Transactions on Computer-Human Interaction*, 4(3):197–229.

Cohen, R. F., Di Battista, G., Tamassia, R., and Tollis, I. G. (1995). Dynamic graph drawings: Trees, series-parallel digraphs, and planar *st*-digraphs. *SIAM Journal on Computing*, 24(5):970–1001.

Cohen, R. F., Di Battista, G., Tamassia, R., Tollis, I. G., and Bertolazzi, P. (1992). A framework for dynamic graph drawing. In *Proceedings of the 8th ACM Annual Symposium on Computational Geometry (SCG'92)*, pages 261–270.

Colin de Verdière, Y. (1989). Empilements de cercles: convergence d'une methode de point fixe. *Forum Mathematicum*, 1:395–402.

Cormen, T., Leiserson, C., and Rivest, R. (1990). *Introduction to Algorithms*. The MIT Electrical Engineering and Computer Science Series. The MIT Press and McGraw-Hill Book Company.

Crescenzi, P., Di Battista, G., and Piperno, A. (1992). A note on optimal area algorithms for upward drawings of binary trees. *Computational Geometry: Theory and Applications*, 2:187–200.

Crescenzi, P., and Piperno, A. (1995). Optimal-area upward drawings of AVL-trees. In *Proceedings of the DIMACS International Workshop on Graph Drawing (GD'94)*. Springer LNCS 894, pages 307–317.

Cruz, I. F., and Twarog, J. P. (1996). 3D graph drawing with simulated annealing. In *Proceedings of the 3rd International Symposium on Graph Drawing (GD'95)*, Springer LNCS 1027, pages 162–165.

Cunningham, W. H. (1976). A network simplex method. *Mathematical Programming*, 11:105–116.

Czyzowicz, J. (1991). Lattice diagrams with few slopes. *Journal of Combinatorial Theory, Series A*, 56:96–108.

Czyzowicz, J., Pelc, A., and Rival, I. (1990). Drawing orders with few slopes. *Discrete Mathematics*, 82:233–250.

Dai, W. W.-M., and Kuh, E. S. (1987). Global spacing of building-block layout. In *Proceedings of the IFIP International Conference on Very Large Scale Integration (VLSI'87)*, pages 193–205.

Datta, A., Lenhof, H.-P., Schwarz, C., and Smid, M. H. M. (1993). Static and dynamic algorithms for k-point clustering problems. In *Proceedings of the 3rd Workshop on Algorithms and Data Structures (WADS'93)*, Springer LNCS 709, pages 265–276.

Davidson, R., and Harel, D. (1996). Drawing graphs nicely using simulated annealing. *ACM Transactions on Graphics*, 15(4):301–331.

de Fraysseix, H., Pach, J., and Pollack, R. (1990). How to draw a planar graph on a grid. *Combinatorica*, 10:41–51.

Di Battista, G., Eades, P., Tamassia, R., and Tollis, I. G. (1994). Algorithms for drawing graphs: An annotated bibliography. *Computational Geometry*, 4:235–282.

Di Battista, G., Eades, P., Tamassia, R., and Tollis, I. G. (1999). *Graph Drawing: Algorithms for the Visualization of Graphs.* Prentice Hall.

Di Battista, G., Liotta, G., and Vargiu, F. (1998a). Spirality and optimal orthogonal drawings. *SIAM Journal on Computing*, 27(6):1764–1811.

Di Battista, G., Liu, W. P., and Rival, I. (1990). Bipartite graphs, upward drawings, and planarity. *Information Processing Letters*, 36(6):317–322.

Di Battista, G., Patrignani, M., and Vargiu, F. (1998b). A split & push approach to 3-D orthogonal drawing. In *Proceedings of the 6th International Symposium on Graph Drawing (GD'98)*. Springer LNCS 1547, pages 87–101.

Di Battista, G., and Tamassia, R. (1988). Algorithms for plane representations of acyclic digraphs. *Theoretical Computer Science*, 61(2-3):175–198.

Di Battista, G., and Tamassia, R. (1989). Incremental planarity testing. In *Proceedings of the 30th Symposium on the Foundations of Computer Science (FOCS'89)*, pages 436–441.

Di Battista, G., and Tamassia, R. (1990). On-line graph algorithms with SPQR-trees. In *Proceedings of the 17th International Colloqium on Automata, Languages and Programming (ICALP'90)*, Springer LNCS 443, pages 598–611.

Di Battista, G., and Tamassia, R. (1996). On-line planarity testing. *SIAM Journal on Computing*, 25(5):956–997.

Di Battista, G., and Vismara, L. (1993). Angles of planar triangulated graphs. In *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing (STOC'93)*, pages 431–437.

Di Battista, G., and Vismara, L. (1996). Angles of planar triangulated graphs. *SIAM Journal on Discrete Mathematics*, 9(3):349–359.

Didimo, W., and Liotta, G. (1998). Computing orthogonal drawings in a variable embedding setting. In *Proceedings of the 9th Annual International Symposium on Algorithms and Computation (ISAAC'98)*, Springer LNCS 1533, pages 79–88.

Dietz, P. F., and Sleator, D. D. (1987). Two algorithms for maintaining order in a list. In *Proceedings of the 19th Annual ACM Symposium of Theory of Computing (STOC'87)*, pages 365–372.

Djidjev, H. N. (1995). A linear algorithm for the maximal planar subgraph problem. In *Proceedings of the 4th Workshop on Algorithms and Data Structures (WADS'95)*. Springer LNCS 955, pages 369–380.

Doddi, S., Mararthe, M. V., Mirzaian, A., Moret, B. M. E., and Zhu, B. (1999). Map labeling and its generalizations. Technical Report LA-UR-96-2411, Los Alamos National Labatory.

Doddi, S., Marathe, M. V., Mirzaian, A., Moret, B. M. E., and Zhu, B. (1997). Map labeling and its generalizations. In *Proceedings of the 8th ACM-SIAM Symposium on Discrete Algorithms (SODA'97)*, pages 148–157.

Dresbach, S. (1995). A new heuristic layout algorithm for directed acyclic graphs. In *Operations Research Proceedings 1994*, pages 121–126.

Duncan, C. A., Goodrich, M. T., and Kobourov, S. G. (1998). Balanced aspect ratio trees and their use for drawing very large graphs. In *Proceedings of the 6th International Symposium on Graph Drawing (GD'98)*. Springer LNCS 1547, pages 111–124.

Eades, P. (1984). A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160.

Eades, P., Cohen, R. F., and Huang, M. L. (1997a). Online animated graph drawing for Web navigation. In *Proceedings of the 5th International Symposium on Graph Drawing (GD'97)*, Springer LNCS 1353, pages 330–335.

Eades, P., and Feng, Q. W. (1996). Multilevel visualization of clustered graphs. In *Proceedings of the 4th International Symposium on Graph Drawing (GD'96)*. Springer LNCS 1190, pages 101–112.

Eades, P., and Feng, Q. W. (1997). Drawing clustered graphs on an orthogonal grid. In *Proceedings of the 5th International Symposium on Graph Drawing (GD'97)*. Springer LNCS 1353, pages 146–157.

Eades, P., Feng, Q. W., and Lin, X. (1996a). Straight-line drawing algorithms for hierarchical graphs and clustered graphs. In *Proceedings of the 4th International Symposium on Graph Drawing (GD'96)*. Springer LNCS 1190, pages 113–128.

Eades, P., Feng, Q., and Nagamochi, H. (1999). Drawing clustered graphs on an orthogonal grid. *Journal on Graph Algorithms and Applications*, 3(4):3–29.

Eades, P., Huang, M. L., and Wang, J. (1997b). Online animated graph drawing using a modified spring algorithm. Technical Report 97–05, Department of Computer Science and Software Engineering, University of Newcastle.

Eades, P., and Kelly, D. (1986). Heuristics for reducing crossings in 2-layered networks. *Ars Combinatorica*, 21.A:89–98.

Eades, P., Lai, W., Misue, K., and Sugiyama, K. (1991). Preserving the mental map of a diagram. In *Proceedings of Compugraphics '91*, pages 24–33.

Eades, P., and Lin, X. (1995). A new heuristic for the feedback arc set problem. *Australian Journal of Combinatorics*, 12:15–26.

Eades, P., Lin, X., and Smyth, W. F. (1993). A fast and effective heuristic for the feedback arc set problem. *Information Processing Letters*, 47:319–323.

Eades, P., and Marks, J. (1995). Graph drawing contest report. In *Proceedings of the DIMACS International Workshop on Graph Drawing (GD'94)*, Springer LNCS 894, pages 143–146.

Eades, P., and Marks, J. (1996). Graph-drawing contest report. In *Proceedings of the 3rd International Symposium on Graph Drawing (GD'95)*, Springer LNCS 1027, pages 224–233.

Eades, P., Marks, J., and North, S. C. (1996). Graph-drawing contest report. In *Proceedings of the 4th International Symposium on Graph Drawing (GD'96)*, Springer LNCS 1190, pages 129–138.

Eades, P., Marks, J., and North, S. C. (1997c). Graph-drawing contest report. In *Proceedings of the 5th International Symposium on Graph Drawing (GD'97)*, Springer LNCS 1353, pages 438–445.

Eades, P., Marks, J., Mutzel, P., and North, S. C. (1998). Graph drawing contest report. In *Proceedings of the 6th International Symposium on Graph Drawing (GD'98)*, Springer LNCS 1547, pages 423–435.

Eades, P., Nagamochi, H., and Feng, Q. (1998). Straight-line drawing algorithms for hierarchical graphs and clustered graphs. Technical Report 98-03, Department of Computer Science and Software Engineering, University of Newcastle, Australia. Available at `ftp://ftp.cs.newcastle.edu.au/pub/techreports/tr98-03.ps.Z`.

Eades, P., Stirk, C., and Whitesides, S. (1996). The techniques of Kolmogorov and Bardzin for three-dimensional orthogonal graph drawing. *Information Processing Letters*, 60(2):97–103. University.

Eades, P., and Sugiyama, K. (1990). How to draw a directed graph. *Journal of Information Processing*, 13:424–437.

Eades, P., Symvonis, A., and Whitesides, S. (1996b). Two algorithms for three dimensional orthogonal graph drawing. In *Proceedings of the 4th International Symposium on Graph Drawing (GD'96)*. Springer LNCS 1190, pages 139–154.

Eades, P., Symvonis, A., and Whitesides, S. (2000). Three-dimensional orthogonal graph drawing. *Discrete Applied Mathematics*, 103(1-3):55–87.

Eades, P., and Whitesides, S. (1994). Drawing graphs in two layers. *Theoretical Computer Science*, 131(2):361–374.

Eades, P., and Wormald, N. C. (1990). Fixed edge-length graph drawing is NP-hard. *Discrete Applied Mathematics*, 28:111–134.

Eades, P., and Wormald, N. C. (1994). Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11(4):379–403.

Edmondson, S., Christensen, J., Marks, J., and Shieber, S. (1997). A general cartographic labeling algorithm. *Cartographica*, 33(4):13–23.

Eiglsperger, M., Fößmeier, U., and Kaufmann, M. (2000). Orthogonal graph drawing with constraints. In *Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms (SODA 2000)*, pages 3–11.

Eppstein, D., and Erickson, J. (1994). Iterated nearest neighbors and finding minimal polytopes. *Discrete Computational Geometry*, 11:321–350.

Even, S. (1979). *Graph Algorithms*. Pitman.

Even, S., and Tarjan, R. E. (1976). Computing an st-numbering. *Theoretical Computer Science*, 2:436–441.

Faria, L., De Figueiredo, C. M. H., and Mendonca, C. F. X. (1998). Splitting number is $\mathcal{NP}$-complete. *Proceedings of the 24th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'98)*, Springer LNCS 1517, pages 285–297.

Fekete, S. P., and Meijer, H. (1999). Rectangle and box visibility graphs in 3D. *International Journal of Computational Geometry and Applications*, 9(1):1–27.

Feng, Q. (1997). *Algorithms for Drawing Clustered Graphs*. PhD thesis, University of Newcastle. `http://www.cs.newcastle.edu.au/Dept/theses.html`.

Feng, Q.-W., Cohen, R. F., and Eades, P. (1995). Planarity for clustered graphs. In *Proceedings of the 3rd European Symposium on Algorithms (ESA'95)*. Springer LNCS 979, pages 213–226.

Fialko, S., and Mutzel, P. (1998). A new approximation algorithm for the planar augmentation problem. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'98)*, pages 260–269.

Fisk, C. J., Caskey, D. L., and West, L. E. (1967). ACCEL: Automated circuit card etching layout. *Proceedings of the IEEE*, 55(11):1971–1982.

Foley, J. D., van Dam, A., Feiner, S. K., and Hughes, J. F. (1990). *Computer Graphics*, 2nd edition. Addison-Wesley.

Force, A. C. G. I. T. (1996). Application challenges to computational geometry. Technical Report TR-521-96, Princeton University.

Formann, M., Hagerup, T., Haralambides, J., Kaufmann, M., Leighton, F. T., Simvonis, A., Welzl, E., and Woeginger, G. (1990). Drawing graphs in the plane with high resolution. In *Proceedings of the 31st Symposium on the Foundations of Computer Science (FOCS'90)*, pages 86–95.

Formann, M., and Wagner, F. (1991). A packing problem with applications to lettering of maps. In *Proceedings of the 7th Annual Symposium on Computational Geometry (SCG '91)*, pages 281–288.

Formella, A., and Keller, J. (1995). Generalized fisheye views of graphs. In *Proceedings of the 3rd International Symposium on Graph Drawing (GD'95)*. Springer LNCS 1027, pages 242–253.

Fößmeier, U. (1997a). Interactive orthogonal graph drawing: Algorithms and bounds. In *Proceedings of the 5th International Symposium on Graph Drawing (GD'97)*. Springer LNCS 1353, pages 111–123.

Fößmeier, U. (1997b). *Orthogonale Visualisierungstechniken für Graphen*. PhD thesis, Eberhard-Karls-Universität zu Tübingen.

Fößmeier, U., Heß, C., and Kaufmann, M. (1998). On improving orthogonal drawings: The 4M-algorithm. In *Proceedings of the 6th International Symposium on Graph Drawing (GD'98)*. Springer LNCS 1547, pages 125–137.

Fößmeier, U., Kant, G., and Kaufmann, M. (1996). 2-visibility drawings of planar graphs. In *Proceedings of the 4th International Symposium on Graph Drawing (GD'96)*. Springer LNCS 1190, pages 155–168.

Fößmeier, U., and Kaufmann, M. (1995). Drawing high degree graphs with low bend numbers. In *Proceedings of the 3rd International Symposium on Graph Drawing (GD'95)*. Springer LNCS 1027, pages 254–266.

Foulds, L. R., Gibbons, P. B., and Giffin, J. W. (1985). Facilities layout adjacency determination: An experimental comparison of three graph theoretic heuristics *Operations Research*, 33:1091–1106.

Foulds, L. R., and Robinson, D. F. (1978). Graph theoretic heuristics for the plant layout problem. *International Journal of Production Research*, 16:27–37.

Fowler, R. J., Paterson, M. S., and Tanimoto, S. L. (1981). Optimal packing and covering in the plane are $\mathcal{NP}$-complete. *Information Processing Letters*, 12(3):133–137.

Freeman, L. C. (1999a). The social network graphics source. School of Social Science, University of California Irvine. `http://eclectic.ss.uci.edu/~lin/gallery.html`.

Freeman, L. C. (1999b). Using molecular modeling software in social network analysis: A practicum. School of Social Science, University of California Irvine. `http://eclectic.ss.uci.edu/~lin/chem.html`.

Freuder, E. C., and Wallace, R. J. (1992). Partial constraint satisfaction. *Artificial Intelligence*, 58(1-3):21–70.

Frick, A. (1997). Upper bounds on the number of hidden nodes in Sugiyama's algorithm. In *Proceedings of the 4th International Symposium on Graph Drawing (GD'96)*. Springer LNCS 1190, pages 169–183.

Frick, A., Ludwig, A., and Mehldau, H. (1995). A fast adaptive layout algorithm for undirected graphs. In *Proceedings of the DIMACS International Workshop on Graph Drawing (GD'94)*. Springer LNCS 894, pages 388–403.

Fruchterman, T. M. J., and Reingold, E. M. (1991). Graph-drawing by force-directed placement. *Software — Practice and Experience*, 21(11):1129–1164.

Gansner, E. R., Koutsofios, E., North, S. C., and Vo, K.-P. (1993). A technique for drawing directed graphs. *IEEE Transactions on Software Engineering*, 19(3):214–230.

Ganter, B., and Wille, R. (1999). *Formal Concept Analysis — Mathematical Foundations*. Springer.

Garey, M. R., and Johnson, D. S. (1983). Crossing number is $\mathcal{NP}$-complete. *SIAM Journal on Algebraic and Discrete Methods*, 4(3):312–316.

Garey, M. R., and Johnson, D. S. (1991). *Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$–Completeness*. W.H. Freeman & Co.

Garg, A., Goodrich, M. T., and Tamassia, R. (1996). Planar upward tree drawings with optimal area. *International Journal Computational Geometry and Applications*, 6:333–356.

Garg, A., and Tamassia, R. (1993). Efficient computation of planar straight-line upward drawings. In *Graph Drawing '93 (Proc. ALCOM Workshop on Graph Drawing)*.

Garg, A., and Tamassia, R. (1994). Planar drawings and angular resolution: Algorithms and bounds. In *Proceedings of the 2nd European Symposium on Algorithms (ESA'94)*. Springer LNCS 855, pages 12–23.

Garg, A., and Tamassia, R. (1995a). On the computational complexity of upward and rectilinear planarity testing. In *Proceedings of the DIMACS International Workshop on Graph Drawing (GD'94)*. Springer LNCS 894, pages 286–297.

Garg, A., and Tamassia, R. (1995b). Upward planarity testing. *Order*, 12:109–133.

Garg, A., and Tamassia, R. (1996a). GIOTTO3D: A system for visualizing hierarchical structures in 3D. In *Proceedings of the 4th International Symposium on Graph Drawing (GD'96)*. Springer LNCS 1190, pages 193–200.

Garg, A., and Tamassia, R. (1996b). A new minimum cost flow algorithm with applications to graph drawing. In *Proceedings of the 4th International Symposium on Graph Drawing (GD'96)*. Springer LNCS 1190, pages 201–216.

Garg, A., and Tamassia, R. (1997). A new minimum cost flow algorithm with applications to graph drawing. In *Proceedings of the 4th International Symposium on Graph Drawing (GD'96)*. Springer LNCS 1190, pages 201–216.

Georgakopoulos, D., Hornick, M., and Sheth, A. (1995). An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and Parallel Databases*, 3(2):119–153.

German Research Center for Artificial Intelligence GmbH (1999). The Verbmobil project. `http://www.dfki.de/verbmobil`.

Godehardt, E. (1988). *Graphs as Structural Models*, Advances in System Analysis 4. Vieweg.

Goldberg, A. V., and Kennedy, R. (1995). An efficient cost scaling algorithm for the assignment problem. *Mathematical Programming*, 71:153–178.

Goldstein, A. J. (1963). An efficient and constructive algorithm for testing whether a graph can be embedded in a plane. In *Graph and Combinatorics Conference, Contract No. NONR 1858-(21)*. Princeton University.

Grötschel, M., Jünger, M., and Reinelt, G. (1985). On the acyclic subgraph polytope. *Mathematical Programming*, 33(1):28–42.

Gutwenger, C., and Mutzel, P. (1998). Planar polyline drawings with good angular resolution. In *Proceedings of the 6th International Symposium on Graph Drawing (GD'98)*. Springer LNCS 1547, pages 167–182.

Hayashi, K., Inoue, M., Masuzawa, T., and Fujiwara, H. (1998). A layout adjustment problem for disjoint rectangles preserving orthogonal order. In *Proceedings of the 6th International Symposium on Graph Drawing*, number 1547 in LNCS, pages 183–197.

He, W., and Marriott, K. (1998). Constrained graph layout. *Constraints*, 3(4):289–314.

Herdeg, W., editor (1981). *Diagrams*. Graphis Press Corporation.

Hermansson, K., and Ojamae, L. (1994). MOVIEMOL — An easy-to-use molecular display and animation program. Technical Report UUIC-B19-500, Institute of Chemistry, University of Uppsala.

Hochbaum, D. S. (1995). *Approximation Algorithms for $\mathcal{NP}$-hard Problems.* PWS Publishing Company, Boston.

Hochbaum, D. S., and Maass, W. (1985). Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM*, 32(1):130–136.

Hong, S.-H., Eades, P., Quigley, A., and Lee, S.-H. (1998). Drawing algorithms for series-parallel digraphs in two and three dimensions. In *Proceedings of the 6th International Symposium on Graph Drawing (GD'98).* Springer LNCS 1547, pages 198–209.

Hong, S.-H., Eades, P., Quigley, A., and Lee, S.-H. (1999a). Drawing series-parallel digraphs symmetrically. To appear in *International Journal of Computational Geometry and Applications.*

Hong, S.-H., Eades, P., Quigley, A., and Lee, S.-H. (1999b). A three dimensional drawing algorithm for series-parallel graphs. Manuscript.

Hopcroft, J., and Tarjan, R. E. (1974). Efficient planarity testing. *Journal of the ACM*, 21:549–568.

Hopcroft, J. E., and Tarjan, R. E. (1973). Dividing a graph into triconnected components. *SIAM Journal on Computing*, 2(3):135–158.

Hsu, W.-L. (1995). A linear time algorithm for finding maximal planar subgraphs. In *Proceedings of the 6th International Symposium on Algorithms and Computation (ISAAC'95).* Springer LNCS 1004, pages 352–361.

Huang, M. L., and Eades, P. (1998a). A fully animated interactive system for clustering and navigating huge graphs. In *Proceedings of the 6th International Symposium on Graph Drawing (GD'98)*, Springer LNCS 1547, pages 374–383.

Hughes, J. G. (1993). *Object-Oriented Databases.* International Series in Computer Science. Prentice-Hall.

Humphrey, W., Dalke, A., and Schulten, K. (1996). VMD — Visual molecular dynamics. *Journal of Molecular Graphics*, 14(1):33–38.

Hutton, M. D., and Lubiw, A. (1991). Upward planar drawing of single source acyclic digraphs. In *Proceedings of the 2nd ACM-SIAM Symposium on Discrete Algorithms (SODA'91)*, pages 203–211.

Imai, H., and Asano, T. (1983). Finding the connected components and a maximum clique of an intersection graph of rectangles in the plane. *Journal of Algorithms*, 4:310–323.

Imai, H., and Asano, T. (1986). Efficient algorithms for geometric graph search problems. *SIAM Journal on Computing*, 15(2):478–494.

Imhof, E. (1962). Die Anordnung der Namen in der Karte. *International Yearbook of Cartography*, 2:93–129.

Imhof, E. (1975). Positioning names on maps. *The American Cartographer*, 2(2):128–144.

Indermark, K., Thomas, W., Huch, F., Leucker, M., and Noll, T. (1999). Various texts about the TRUTH system for modelling concurrent systems,

Lehrstuhl für Informatik II, RWTH Aachen. `http://www-i2.informatik.rwth-aachen.de/Forschung/MCS/`.

Isoda, S., Shimomura, T., and Ono, Y. (1987). VIPS: A visual debugger. *IEEE Software*, 4(3):8–19.

Iturriaga, C., and Lubiw, A. (1997). $\mathcal{NP}$-hardness of some map labeling problems. Technical Report CS-97-18, University of Waterloo.

Jain, A. K., and Dubes, R. C. (1988). *Algorithms for Clustering Data*. Prentice Hall.

Johnson, D. S. (1982). The $\mathcal{NP}$-completeness column: An ongoing guide. *Journal of Algorithms*, 3:89–99.

Jünger, M., Lee, E. K., Mutzel, P., and Odenthal, T. (1997). A polyhedral approach to the multi-layer crossing minimization problem. In *Proceedings of the 5th International Symposium on Graph Drawing (GD'97)*. Springer LNCS 1353, pages 13–24.

Jünger, M., and Mutzel, P. (1996). Maximum planar subgraphs and nice embeddings: Practical layout tools. *Algorithmica*, 16:33–59.

Jünger, M., and Mutzel, P. (1997). 2-Layer straightline crossing minimization: Performance of exact and heuristic algorithms. *Journal on Graph Algorithms and Applications*, 1(1):1–25.

Kakoulis, K. G., and Tollis, I. G. (1996). On the edge label placement problem. In *Proceedings of the 4th International Symposium on Graph Drawing (GD'96)*. Springer LNCS 1190, pages 241–256.

Kakoulis, K. G., and Tollis, I. G. (1997). An algorithm for labeling edges of hierarchical drawings. In *Proceedings of the 5th International Symposium on Graph Drawing (GD'97)*. Springer LNCS 1353, pages 169–180.

Kakoulis, K. G., and Tollis, I. G. (1998a). On the multiple label placement problem. In *Proceedings of the 10th Canadian Conference on Computational Geometry (CCCG'98)*, pages 66–67.

Kakoulis, K. G., and Tollis, I. G. (1998b). A unified approach to labeling graphical features. In *Proceedings of the 14th Annual ACM Symposium on Computional Geometry (SCG'98)*, pages 347–356.

Kamada, T., and Kawai, S. (1988). A simple method for computing general positions in displaying three-dimensional objects. *Computer Vision, Graphics and Image Processing*, 41:43–56.

Kamada, T., and Kawai, S. (1989). An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31:7–15.

Kant, G. (1996). Drawing planar graphs using the canonical ordering. *Algorithmica*, 16:4–32.

Kant, G., and Bodlaender, H. L. (1991). Planar graph augmentation problems. In *Proceedings of the 2nd Workshop on Algorithms and Data Structures (WADS'91)*, Springer LNCS 519, pages 286–298.

Karp, R. (1972). Reducibility among combinatorical problems. In *Complexity of Computer Computations*, pages 85–103. Plenum Press.

Kato, T., and Imai, H. (1988). The $\mathcal{NP}$-completeness of the character placement problem of 2 or 3 degrees of freedom. In *Record of Joint Conference of Electrical and Electronic Engineers in Kyushu*, page 1138.

Keahey, T. A., and Robertson, E. (1996). Techniques for non-linear magnifaction transformations. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis'96)*, pages 38–45.

Kedem, G., and Watanabe, H. (1984). Graph optimization techniques for IC-layout and compaction. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, CAD-3(1):12–20.

Kelly, D., and Rival, I. (1975). Planar lattices. *Canadian Journal of Mathematics*, 27:636–665.

Kenis, P. (1999). Analysing social network data by means of visualisation techniques. Paper presented at the *19th International Conference on Social Network Analysis (Sunbelt XIX)*, Charleston.

Kernighan, B. W., and Lin, S. (1970). An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2):291–307.

Kirchhoff, G. R. (1847). Über die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Verteilung galvanischer Ströme geführt wird. *Annalen der Physik und Chemie*, 72:497–508.

Klau, G. W., and Mutzel, P. (1998). Quasi-orthogonal drawing of planar graphs. Technical Report 98-1-013, Max-Planck-Institut für Informatik, Saarbrücken.

Klau, G. W., and Mutzel, P. (1999a). Combining graph labeling and compaction. In *Proceedings of the 7th International Symposium on Graph Drawing (GD'99)*. Springer LNCS 1731, pages 27–37.

Klau, G. W., and Mutzel, P. (1999b). Optimal compaction of orthogonal grid drawings. In *Integer Programming and Combinatorial Optimization (IPCO'99)*. Springer LNCS 1610, pages 304–319.

Knipping, L. (1998). Beschriftung von Linienzügen. Diplomarbeit, Fachbereich Mathematik und Informatik, Freie Universität Berlin.

Knuth, D. E., and Raghunathan, A. (1992). The problem of compatible representatives. *SIAM Journal on Discrete Mathematics*, 5(3):422–427.

Koebe, P. (1936). Kontaktprobleme auf der konformen Abbildung. *Berichte über die Verhandlungen der Sächsischen Akademie der Wissenschaften zu Leipzig, Mathematisch-Physikalische Klasse*, 88:141–164.

Kolmogorov, A. N., and Bardzin, Y. M. (1967). About realization of sets in 3-dimensional space. *Problems in Cybernetics*, pages 261–268.

Kosak, C., Marks, J., and Shieber, S. (1994). Automating the layout of network diagrams with specified visual organization. *IEEE Transactions on Systems, Man and Cybernetics*, 24(3):440–454.

Krackhardt, D., Blythe, J., and McGrath, C. (1994). KrackPlot 3.0: An improved network drawing program. *Connections*, 17(2):53–55.

Kruskal, J. B., and Wish, M. (1978). *Multidimensional Scaling.* Sage University Paper Series on Quantitative Applications in the Social Sciences 07-011.

Kucera, L., Mehlhorn, K., Preis, B., and Schwarzenecker, E. (1993). Exact algorithms for a geometric packing problem. *Proceedings of the 10th Symposium on the Theoretical Aspects of Computer Science (STACS'93).* Springer LNCS 665, pages 317–322.

Kumar, A., and Fowler, R. H. (1994). A spring modelling algorithm to position nodes of an undirected graph in three dimensions. Technical report, Department of Computer Science, University of Texas.

Laguna, M., and Martí, R. (1999). Grasp and path relinking for 2-layer straight line crossing minimization. *INFORMS Journal on Computing*, 11(1):44–52.

Laguna, M., Martí, R., and Valls, V. (1997). Arc crossing minimization in hierarchical digraphs with tabu search. *Computers and Operations Research*, 24(12):1175–1186.

Lam, S., and Sethi, R. (1977). Worst case analysis of two scheduling problems. *SIAM Journal on Computing*, 6:518–536.

LaPaugh, A. S. (1998). VLSI Layout Algorithms. In *Algorithms and Theory of Computation Handbook.* CRC Press.

Leiserson, C. E. (1980). Area-efficient graph layouts (for VLSI). In *Proceedings of the 21st Annual IEEE Symposium on Foundations of Computer Science (FOCS'80)*, pages 270–281.

Lempel, A., and Cederbaum, I. (1966). Minimum feedback arc and vertex sets of a directed graph. *IEEE Transactions on Circuit Theory*, CT–13(4):339–403.

Lempel, A., Even, S., and Cederbaum, I. (1967). An algorithm for planarity testing of graphs. In *Theory of Graphs: International Symposium (Rome 1966)*, pages 215–232. Gordon and Breach.

Lengauer, T. (1989). Hierarchical planarity testing algorithms. *Journal of the ACM*, 36:474–509.

Lengauer, T. (1990). *Combinatorial Algorithms for Integrated Circuit Layout.* Applicable Theory in Computer Science. B. G. Teubner and John Wiley & Sons.

Leung, J. (1992). A new graph-theoretic heuristic for facility layout. *Management Science*, 38(4):594–605.

Lewis, J. M., and Yannakakis, M. (1980). The node-deletion problem for hereditary properties is $\mathcal{NP}$-complete. *Journal of Computer and System Sciences*, 20(2):219–230.

Liebers, A. (1996). Methods for planarizing graphs - A survey and annotated bibliography. Technical Report Konstanzer Schriften in Mathematik und Informatik Nr. 12, Fakultät für Mathematik und Informatik, Universität Konstanz. ISSN 1430-3558. To appear in *Journal on Graph Algorithms and Applications.*

Lin, X. (1992). *Analysis of Algorithms for Drawing Graphs.* PhD thesis, University of Queensland.

Lino, P., Martí, R., and Valls, V. (1996). A branch and bound algorithm for minimizing the number of crossing arcs in bipartite graphs. *Journal of Operational Research*, 90:303–319.

Lipton, R. J., Rose, D. J., and Tarjan, R. E. (1979). Generalized nested dissection. *SIAM Journal on Numerical Analysis*, 16:346–358.

Lipton, R. J., and Tarjan, R. E. (1970). A seperator theorem for planar graphs. In *Proceedings of the Conference on Theoretical Computer Science*, pages 1–10.

Liu, P. C., and Geldmacher, R. C. (1977). On the deletion of nonplanar edges of a graph. In *Proceedings of the 10th Southeastern Conference on Combinatorics, Graph Theory, and Computing*, pages 727–738.

Lyons, K. A. (1992). Cluster busting in anchored graph drawing. In *Proceedings of the '92 CAS Conference (CASCON'92)*, pages 7–17.

Lyons, K. A., Meijer, H., and Rappaport, D. (1998). Algorithms for cluster busting in anchored graph drawing. *Journal on Graph Algorithms and Applications*, 2(1):1–24.

Mackworth, A. K., and Freuder, E. C. (1985). The complexity of some polynomial network consistency algorithms for constraint satisfaction problem. *Artificial Intelligence*, 25(1):65–74.

Mäkinen, E. (1990). Experiments on drawing 2-level hierarchical graphs. *International Journal of Computer and Mathematics*, 36:175–181.

Mäkinen, E., and Sieranta, M. (1994). Genetic algorithms for drawing bipartite graphs. *Internatonal Journal of Computer Mathematics*, 53:157–166.

Malitz, S., and Papakostas, A. (1992). On the angular resolution of planar graphs. In *Proceedings of the 24th Annual ACM Symposium on the Theory of Computing (STOC'92)*, pages 527–538.

Malitz, S., and Papakostas, A. (1994). On the angular resolution of planar graphs. *SIAM Journal on Discrete Mathematics*, 7(2):172–183.

Manning, J. (1990). *Geometric Symmetry in Graphs.* PhD thesis, Purdue University.

Marks, J., and Shieber, S. (1991). The computational complexity of cartographic label placement. Technical Report TR-05-91, Harvard University Computer Science.

Masuda, S., Kimura, S., Kashiwabara, T., and Fujisawa, T. (1983). On the Manhattan wiring problem. Technical Report CAS 83-20, Institute of Electronics and Communication Engineers of Japan.

Masui, T. (1992). Graphic object layout with interactive genetic algorithms. In *Proceedings of the 1992 IEEE Workshop on Visual Languages (VL'92)*, pages 74–87.

Matuszewski, C., Schönfeld, R., and Molitor, P. (1999). Using sifting for k-layer straightline crossing minimization. *Proceedings of the 7th Symposium on Graph Drawing (GD'99).* Springer LNCS 1731, pages 217–224.

McGrath, C., Blythe, J., and Krackhardt, D. (1996). Seeing groups in graph layouts. *Connections*, 19(2):22–29.

McGrath, C., and Borgatti, S. P. (1999). *The International Network for Social Network Analysis Homepage.* `http://www.heinz.cmu.edu/project/INSNA/`.

Mehlhorn, K. (1984). *Data Structures and Algorithms. Volume 2: Graph Algorithms and $\mathcal{NP}$-Completeness.* EATCS Monographs on Theoretical Computer Science. Springer.

Mehlhorn, K., and Näher, S. (1999). *The* LEDA *Platform of Combinatorial and Geometric Computing.* Cambridge University Press. Project home page at `http://www.mpi-sb.mpg.de/LEDA/`.

Messinger, E. B., Rowe, L. A., and Henry, R. H. (1991). A divide-and-conquer algorithm for the automatic layout of large directed graphs. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-21(1):1–12.

Miriyala, K., Hornik, S. W., and Tamassia, R. (1993). An incremental approach to aesthetic graph layout. In *Proceedings of the 6th International Workshop on Computer-Aided Software Engineering (CASE'93)*, pages 297–308.

Misue, K., Eades, P., Lai, W., and Sugiyama, K. (1995). Layout adjustment and the mental map. *Journal of Visual Languages and Computing*, 6:183–210.

Moen, S. (1990). Drawing dynamic trees. *IEEE Software*, 7:21–28.

Monien, B., Ramme, F., and Salmen, H. (1995). A parallel simulated annealing algorithm for generating 3D layouts of undirected graphs. In *Proceedings of the 3rd International Symposium on Graph Drawing (GD'95).* Springer LNCS 1027, pages 396–408.

Monien, B., Ramme, F., and Salmen, H. (1996). A parallel simulated annealing algorithm for generating 3d layouts of undirected graphs. In *Proceedings of the 3rd International Symposium on Graph Drawing (GD'95).* Springer LNCS 1027, pages 396–408.

MTA (1999). MTA New York City subway map. `http://www.mta.nyc.ny.us/nyct/images/sub1a.gif` and `http://www.mta.nyc.ny.us/nyct/images/sub2a.gif`.

Mukherjea, S., Foley, J., and Hudson, S. (1994). Interactive clustering for navigating in hypermedia systems. In *Proceedings of the ACM European Conference on Hypermedia Tehcnologie.*

Mutzel, P. (1994). *The Maximum Planar Subgraph Problem.* PhD thesis, Universität zu Köln.

Mutzel, P. (1995). A polyhedral approach to planar augmentation and related problems. In *Proceedings of the 3rd European Symposium on Algorithms (ESA'95).* Springer LNCS 979, pages 494–507.

Mutzel, P. (1997). An alternative method to crossing minimization on hierarchical graphs. In *Proceedings of the 4th International Symposium on Graph Drawing (GD'96).* Springer LNCS 1190, pages 318–333.

Nakano, S., Rahman, M. S., and Nishizeki, T. (1997). A linear-time algorithm for four-partitioning four-connected planar graphs. In *Proceedings of the 5th International Symposium on Graph Drawing (GD'97)*. Springer LNCS 1353, pages 334 – 344.

Nemhauser, G. L., and Sigismondi, G. (1992). A strong cutting plane/branch-and-bound algorithm for node packing. *Journal of the Operational Research Society*, 43:443–457.

Nielsen, J. (1990). The art of navigating throuh hypertext. *Communications of the ACM*, 33(3):296–310.

Nishizeki, T., and Chiba, N. (1988). *Planar Graphs: Theory and Algorithms*. North-Holland Mathematics Studies 140/32.

North, S. C. (1996). Incremental layout with DynaDag. In *Proceedings of the 3rd International Symposium on Graph Drawing (GD'95)*. Springer LNCS 1027, pages 409–418.

Oerder, M., and Ney, H. (1993). Word graphs: An efficient interface between continuous-speech recognition and language understanding. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP'93)*, volume II, pages 119–122.

Ostry, D. (1996). Some three-dimensional graph drawing algorithms. Master's thesis, University of Newcastle.

Otten, R. H. J. M., and van Wijk, J. G. (1978). Graph representation in interactive layout design. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, pages 914–918.

Papakostas, A. (1995). Upward planarity testing of outerplanar dags. In *Proceedings of the DIMACS International Workshop on Graph Drawing (GD'94)*. Springer LNCS 894, pages 298–306.

Papakostas, A., Six, J. M., and Tollis, I. G. (1996). Experimental and theoretical results in interactive orthogonal graph drawing. In *Proceedings of the 4th International Symposium on Graph Drawing (GD'96)*. Springer LNCS 1190, pages 371–386.

Papakostas, A., and Tollis, I. G. (1997a). Incremental orthogonal graph drawing in three dimensions. In *Proceedings of the 5th International Symposium on Graph Drawing (GD'97)*. Springer LNCS 1353, pages 52–63.

Papakostas, A., and Tollis, I. G. (1997b). Incremental orthogonal graph drawing in three-dimensions. Technical Report UTDCS-02-97, Dept. of Computer Sciencs, University of Texas at Dallas.

Papakostas, A., and Tollis, I. G. (1997c). Orthogonal drawing of high degree graphs with small area and few bends. In *Proceedings of the 5th Workshop on Algorithms and Data Structures (WADS'97)*. Springer LNCS 1272, pages 354–367.

Papakostas, A., and Tollis, I. G. (1997d). A pairing technique for area-efficient orthogonal drawings. In *Proceedings of the 4th International Symposium on Graph Drawing (GD'96)*. Springer LNCS 1190, pages 354–370.

Papakostas, A., and Tollis, I. G. (1998). Interactive orthogonal graph drawing. *IEEE Transactions on Computers*, 47(11):1297–1309.

Patrignani, M. (1999a). On the complexity of orthogonal compaction. Technical Report RT–DIA–39–99, Dipartimento di Informatica e Automazione, Università degli Studi di Roma Tre.

Patrignani, M. (1999b). On the complexity of orthogonal compaction. *Proceedings of the 6th Workshop on Algorithms and Data Structures (WADS'99)*. Springer LNCS 1663, pages 56–61.

Patrignani, M., and Vargiu, F. (1997). 3DCube: A tool for the three dimensional graph drawing. In *Proceedings of the 5th International Symposium on Graph Drawing (GD'97)*. Springer LNCS 1353, pages 284–290.

Paulish, F. N. (1993). *The Design of an Extendible Graph Editor*. Springer LNCS 704.

Platt, C. (1976). Planar lattices and planar graphs. *Journal of Combinatorial Theory, Series B*, 21:30–39.

Poon, C. K., Zhu, B., and Chin, F. (1998). A polynomial time solution for labeling a rectilinear map. *Information Processing Letters*, 65:201–207.

Poutré, J. A. L. (1994). Alpha-algorithms for incremental planarity testing. In *Proceedings of the 26th Annual ACM Symposium on the Theory of Computation (STOC'94)*, pages 706–715.

Purchase, H. C. (1997). Which aesthetic has the greatest effect on human understanding? In *Proceedings of the 5th International Symposium on Graph Drawing (GD'97)*. Springer LNCS 1353, pages 248–261.

Purchase, H. C., Cohen, R. F., and James, M. (1996). Validating graph drawing aesthetics. In *Proceedings of the 3rd International Symposium on Graph Drawing (GD'95)*. Springer LNCS 1027, pages 435–446.

Purchase, H. C., Cohen, R. F., and James, M. (1997). An experimental study of the basis for graph drawing algorithms. *ACM Journal of Experimental Algorithmics*, 2(4).

Quinn, N. R., and Breuer, M. A. (1979). A force directed component placement procedure for printed circuit boards. *IEEE Transactions on Circuits and Systems*, 26(6):377–388.

Reeves, C. M. (1995). *Modern Heuristic Techniques for Combinatorial Problems*. McGraw-Hill.

Reggiani, M. G., and Marchetti, F. E. (1988). A proposed method for representing hierarchies. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1):2–8.

Reinelt, G. (1985). *The linear ordering problem: algorithms and applications*. Research and Exposition in Mathematics 8, Heldermann.

Reingold, E. M., and Tilford, J. S. (1981). Tidier drawings of trees. *IEEE Transactions on Software Engineering*, 7(2):223–228.

Rival, I. (1985). The diagram. In *Graphs and Order*, NATO ASI Series, pages 103–133. Reidel Publishing.

Robertson, G. G., Mackinlay, J. D., and Card, S. K. (1993). Cone trees: Animated 3*d* visualizations of hierarchical information. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 189–193.

Rosenstiehl, P., and Tarjan, R. E. (1986). Rectilinear planar layouts of planar graphs and bipolar orientations. *Discrete & Computational Geometry*, 1(4):342–351.

Roxborough, T., and Sen, A. (1997). Graph clustering using multiway ratio cut. In *Proceedings of the 5th International Symposium on Graph Drawing (GD'97)*. Springer LNCS 1353, pages 291–296.

Rudell, R. (1993). Dynamic variable ordering for ordered binary decision diagrams. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD'93)*, pages 42–47.

Sablowski, R., and Frick, A. (1996). Automatic graph clustering. In *Proceedings of the 4th International Symposium on Graph Drawing (GD'96)*. Springer LNCS 1190, pages 395–400.

Sander, G. (1994). Graph layout through the VCG tool. Technical Report A03/94, Universität des Saarlandes.

Sander, G. (1996a). A fast heuristic for hierarchical Manhattan layout. In *Proceedings of the 3rd International Symposium on Graph Drawing (GD'95)*. Springer LNCS 1027, pages 447–458.

Sander, G. (1996b). Graph layout for applications in compiler construction. Technical Report A/01/96, FB 14 Informatik, Universität des Saarlandes.

Sarkar, M., and Brown, M. H. (1994). Graphical fisheye views. *Communications of the ACM*, 37(12):73–84.

Schlag, M., Liao, Y.-Z., and Wong, C. K. (1983). An algorithm for optimal two-dimensional compaction of VLSI layouts. *Integration, the VLSI Journal*, 1:179–209.

Schnyder, W. (1990). Embedding planar graphs on the grid. In *Proceedings of the 1st ACM-SIAM Symposium on Discrete Algorithms (SODA'90)*, pages 138–148.

Schrijver, A. (1986). *Theory of Linear and Integer Programming*. Wiley-Interscience.

Sedgewick, R. (1988). *Algorithms*, pages 438–441. Addison–Wesley, 2nd edition.

Shiloach, Y. (1976). *Arrangements of planar graphs on the planar lattice*. PhD thesis, Weizmann Institute of Science.

Sim, S. (1996). Automatic graph drawing algorithms. Manuscript, available at `http://www.cs.toronto.edu/~simsuz/papers/grafdraw.ps.gz`.

Six, J. M., Kakoulis, K. G., and Tollis, I. G. (1998). Refinement of orthogonal graph drawings. In *Proceedings of the 6th International Symposium on Graph Drawing (GD'98)*. Springer LNCS 1547, pages 302–315.

Strijk, T., and van Kreveld, M. (1999). Labeling a rectilinear map more efficiently. *Information Processing Letters*, 69(1):25–30.

Strijk, T., and Wolf, A. (1999). Labeling points with circles. Technical Report TR-99-08, Institut für Informatik, Freie Universität Berlin.

Stumme, G., and Wille, R. (1995). A geometrical heuristic for drawing concept lattices. In *Proceedings of the DIMACS International Workshop on Graph Drawing (GD'94)*. Springer LNCS 894, pages 452–460.

Sugiyama, K. (1987). A cognitive approach for graph drawing. *Cybernetic Systems*, 18(6):447–488.

Sugiyama, K., and Misue, K. (1991). Visualisation of structural information: Automatic drawing of compound digraphs. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(4):876–892.

Sugiyama, K., and Misue, K. (1995). A simple and unified method for drawing graphs: Magnetic-spring algorithm. In *Proceedings of the DIMACS International Workshop on Graph Drawing (GD'94)*. Springer LNCS 894, pages 364–375.

Sugiyama, K., Tagawa, S., and Toda, M. (1981). Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(2):109–125.

Supowit, K. J., and Reingold, E. M. (1983). The complexity of drawing trees nicely. *Acta Informatica*, 18:377–392.

Tamassia, R. (1987). On embedding a graph in the grid with the minimum number of bends. *SIAM Journal on Computing*, 16(3):421–444.

Tamassia, R. (1998). Constraints in graph drawing algorithms. *Constraints*, 3(1):87–120.

Tamassia, R., Di Battista, G., and Batini, C. (1988). Automatic graph drawing and readability of diagrams. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1):61–79.

Tamassia, R., and Tollis, I. G. (1986). A unified approach to visibility representations of planar graphs. *Discrete & Computational Geometry*, 1(4):321–341.

Tamassia, R., and Tollis, I. G. (1989). Planar grid embedding in linear time. *IEEE Transactions on Circuits and Systems*, 36(9):1230–1234.

Tamassia, R., Tollis, I. G., and Vitter, J. S. (1991). Lower bounds for planar orthogonal drawings of graphs. *Information Processing Letters*, 39(1):35–40.

Tanenbaum, A. S. (1995). *Distributed Operating Systems*. Prentice Hall.

Tarjan, R. E. (1983). *Data structures and network algorithms* CBMS-NSF Regional Conference Series in Applied Mathematics 44, SIAM.

Thomassen, C. (1980). Planarity and duality of finite and infinite planar graphs. *Journal of Combinatorial Theory, Series B*, 29:244–271.

Thompson, C. D. (1980). *A Complexity Theory for VLSI*. PhD thesis, Carnegie Mellon University.

Tunkelang, D. (1994). A practical approach to drawing undirected graphs. Technical Report CMU-CS-94-161, School of Computer Science, Carnegie Mellon University.

Tutte, W. T. (1960). Convex representations of graphs. *Proceedings of the London Mathematical Society, Third Series*, 10:304–320.

Tutte, W. T. (1963). How to draw a graph. *Proceedings of the London Mathematical Society, Third Series*, 13:743–768.

Ullman, J. (1989). *Principles of Database and Knowledgebase Systems*, volume 1. Computer Science Press.

Utech, J., Branke, J., Schmeck, H., and Eades, P. (1998). An evolutionary algorithm for drawing directed graphs. In *Proceedings of the International Conference on Imaging Science, Systems, and Technology*, pages 154–160.

Valdes, J., Tarjan, R. E., and Lawler, E. L. (1982). The recognition of series parallel digraphs. *SIAM Journal on Computing*, 11:298–313.

Valiant, L. (1981). Universality considerations in VLSI circuits. *IEEE Transactions on Computers*, C-30(2):135–140.

van Kreveld, M., Strijk, T., and Wolff, A. (1998). Point set labeling with sliding labels. In *Proceedings of the 14th Annual ACM Symposium on Computational Geometry (SCG'98)*, pages 337–346.

Verweij, B., and Aardal, K. (1999). An optimisation algorithm for maximum independent set with applications in map labelling. In *Proceedings of the 7th European Symposium on Algorithms (ESA'99)*. Springer LNCS 1643, pages 426–437.

Vogt, F. (1996). *Formale Begriffsanalyse mit C++: Datenstrukturen und Algorithmen*. Springer.

Vogt, F., and Wille, R. (1995). TOSCANA — a graphical tool for analyzing and exploring data. In *Proceedings of the DIMACS International Workshop on Graph Drawing (GD'94)*. Springer LNCS 894, pages 226–233.

Vossen, G. (1991). *Datenbankmodelle, Datenbanksprachen und Datenbankmanagement-Systeme*. Addison-Wesley.

Wagner, F. (1994). Approximate map labeling is in $\Omega(n \log n)$. *Information Processing Letters*, 52(3):161–165.

Wagner, F., and Wolff, A. (1995a). An efficient and effective approximation algorithm for the map labeling problem. In *Proceedings of the 3rd European Symposium on Algorithms (ESA'95)*. Springer LNCS 979, pages 420–433.

Wagner, F., and Wolff, A. (1995b). Map labeling heuristics: Provably good and practically useful. In *Proceedings of the 11th Annual ACM Symposium on Computational Geometry (SCG'95)*, pages 109–118.

Wagner, F., and Wolff, A. (1997). A practical map labeling algorithm. *Computational Geometry: Theory and Applications*, 7:387–404.

Wagner, F., and Wolff, A. (1998). A combinatorial framework for map labeling. In *Proceedings of the 6th International Symposium on Graph Drawing (GD'98)*. Springer LNCS 1547, pages 316–331.

Wang, X., and Miyamoto, I. (1995). Generating cunstomized layouts. In *Proceedings of the 3rd International Symposium on Graph Drawing (GD'95)*. Springer LNCS 1027, pages 504–515.

Wang, X., and Miyamoto, I. (1996). Generating customized layouts. In *Proceedings of the 3rd International Symposium on Graph Drawing (GD'95)*. Springer LNCS 1027, pages 504–515.

Warfield, J. (1977). Crossing theory and hierarchy mapping. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-7(7):502–523.

Warnke, V., Kompe, R., Niemann, H., and Nöth, E. (1997). Integrated dialog act segmentation and classification using prosodic features and language models. Technical Report Verbmobil-Report 218, Lehrstuhl für Mustererkennung 5, Universität Erlangen-Nürnberg.

Wasserman, S., and Faust, K. (1994). *Social Network Analysis: Methods and Applications*. Cambridge University Press.

Watanabe, H. (1984). *IC Layout Generation and Compaction Using Mathematical Optimization*. PhD thesis, University of Rochester.

Watanabe, T., Ae, T., and Nakamura, A. (1983). On the $\mathcal{NP}$-hardness of edge-deletion and -contraction problems. *Discrete Applied Mathematics*, 6:63–78.

Webber, R. (1997). Finding the best viewpoints for three-dimensional graph drawings. In *Proceedings of the 5th International Symposium on Graph Drawing (GD'97)*. Springer LNCS 1353, pages 87–98.

Webber, R. (1998). *Finding the Best Viewpoint for Three-Dimensional Graph Drawings*. PhD thesis, University of Newcastle. `http://www.cs.mu.oz.au/~rwebber/research/thesis/`.

Wei, Y.-C., and Cheng, C.-K. (1991). Ratio cut partitioning for hierarchical designs. *IEEE Transactions on Computer-Aided Design*, 10(7):911–921.

West, D. (1996). *Introduction to Graph Theory*. Prentice Hall.

White, D. (1999). Pgraph of Canaan genealogy made by Pajek program. Manuscript. `http://eclectic.ss.uci.edu/~drwhite/pgraph/p-graphs.html`.

Wiese, R., and Kaufmann, M. (1998). Adding constraints to an algorithm for orthogonal graph drawing. In *Proceedings of the 6th International Symposium on Graph Drawing (GD'98)*. Springer LNCS 1547, pages 462–463.

Wille, R. (1989). Lattices in data analysis: How to draw them with a computer. In *Algorithms and Order*, NATO ASI Series, pages 33–58. Kluwer Academic Publishers.

Wille, R. (1997). Introduction to formal concept analysis. In *Modelli e modellizzazione. Models and modelling*. Consiglio Nazionale delle Ricerche, Instituto di Studi sulli Ricerca e Documentazione Scientifica, Roma, pages 39–51.

Winter, A., and Schürr, A. (1997). Modules and updatable graph views for programmed graph rewriting systems. Technical Report AIB 97-3, Lehrstuhl für Informatik III, RWTH Aachen.

Wolff, A. (1999). *Map Labeling in Theory and Practice*. PhD thesis, Freie Universität Berlin.

Wolff, A., Knipping, L., van Kreveld, M., Strijk, T., and Agarwal, P. K. (1999). A simple and efficient algorithm for high-quality line labeling. In *Proceedings of GISRUK'99*.

Wood, D. (1998a). An algorithm for three-dimensional orthogonal graph drawing. In *Proceedings of the 6th International Symposium on Graph Drawing (GD'98)*. Springer LNCS 1547, pages 332–346.

Wood, D. (1998b). Two-bend three-dimensional orthogonal grid drawing of maximum degree five graphs. Technical Report 98/03, Monash University.

Wood, D. R. (1999a). Multi-dimensional orthogonal graph drawing in the general position model. Technical Report 99/38, Monash University.

Wood, D. R. (1999b). A new algorithm and open problems in three-dimensional orthogonal graph drawing. In *Proceedings of the 10th Australasian Workshop on Combinatorical Algorithms (AWOCA'99)*, pages 157–167.

Wood, D. R. (2000). Three-Dimensional Orthogonal Graph Drawing. PhD thesis, Monash University.

Yannakakis, M. (1978). Node- and edge-deletion $\mathcal{NP}$-complete problems. In *Proceedings 10th Annual ACM Symposium on the Theory of Computing (STOC'78)*, pages 253–264.

Yoeli, P. (1972). The logic of automated map lettering. *The Cartographic Journal*, 9:99–108.

Zeller, A., and Lütkehaus, D. (1996). DDD — A free graphical front-end for UNIX debuggers. *ACM SIGPLAN Notices*, 31(1):22–27.

Zoraster, S. (1986). Integer programming applied to the map label placement problem. *Cartographica*, 23(3):16–27.

Zoraster, S. (1990). The solution of large 0-1 integer programming problems encountered in automated cartography. *Operations Research*, 38(5):752–759.