Confluent Layered Drawings*

David Eppstein, Michael T. Goodrich, and Jeremy Yu Meng

School of Information and Computer Science,
University of California, Irvine,
Irvine, CA 92697, USA
{eppstein,goodrich,ymeng}@ics.uci.edu

Abstract. We combine the idea of confluent drawings with Sugiyama style drawings, in order to reduce the edge crossings in the resultant drawings. Furthermore, it is easier to understand the structures of graphs from the mixed style drawings. The basic idea is to cover a layered graph by complete bipartite subgraphs (bicliques), then replace bicliques with tree-like structures. The biclique cover problem is reduced to a special edge coloring problem and solved by heuristic coloring algorithms. Our method can be extended to obtain multi-depth confluent layered drawings.

1 Introduction

Layered drawings visualize hierarchical graphs in a way such that vertices are arranged in layers and edges are drawn as straight lines or curves connecting these layers. A common method was introduced by Sugiyama, Tagawa and Toda [25] and by Carpano [4]. Several closely related methods were proposed later (see e.g. [12, 19, 15, 22, 6, 20, 11].)

Crossing reduction is one of the most important objectives in layered drawings. But it is well known that for two-layer graphs the straight-line crossing minimization problem is NP-complete [14]. The problem remains NP-complete even when one layer is fixed. Jünger and Mutzel [16] present exact algorithms for this problem, and perform experimental comparison of their results with various heuristic methods. Recently new methods related to crossing reduction ([26, 1, 8, 23, 10]) have been proposed.

However when the given two-layer graph is dense, even in an optimum solution, there are still a large number of crossings. Then the resulting straight-line drawing will be hard to read, since edge-crossing minimization is one of the most important aesthetic criteria for visualizing graphs [24]. This give us a motivation for exploring new approaches to reduce the crossings in a drawing other than the traditional methods.

In addition, it is sometime of interest to find the bicliques between two layers. For example in the drawing of a call graph, it is interesting to find out which set

* Work by the first author is supported by NSF grant CCR-9912338. Work by the second and the third author is supported by NSF grants CCR-0098068, CCR-0225642, and DUE-0231467.

J. Pach (Ed.): GD 2004, LNCS 3383, pp. 184–194, 2004.

of modules are calling a common set of functions and what are those common functions. Call graphs are usually visualized as layered drawings. However it is hard to learn this information from layered drawings by traditional Sugiyama-style approaches, especially when the input graphs are dense.

Our previous work [5] introduces the concept of confluent drawings. In [5] we talk about the confluent drawability of several classes of graphs and give a heuristic for finding confluent drawings of graphs with bounded arboricity. In this paper we experiment with an implementation of confluent drawings for the layered graphs. However we relax the constraint of planarity and allow crossings in the drawings, while it is not allowed to have crossings in a confluent drawing in our previous definitions.

We are aware of the Edge Concentration method by Newbery [21]. Edge Concentration and our method share a same idea of covering by bicliques. But in Newbery's method, dummy nodes (edge concentrators) are explicit in the drawing and treated equally as original nodes, which causes the nodes' original levels to change. In our method dummy nodes are implicit in the curve representation of edges and the original levels are preserved. Furthermore, our method uses a very different algorithm to compute the biclique covers.

2 Definitions

In this section we give definitions for confluent layered drawings. The definitions almost remain the same as in our previous confluent drawing paper, except that the planarity constraints are dropped. Fig. 1 gives an idea of confluent layered drawings. Edges in the drawing are represented as smooth curves.

A curve is *locally monotone* if it contains no sharp turns, that is, it contains no point with left and right tangents that forms a angle less than or equal to 90

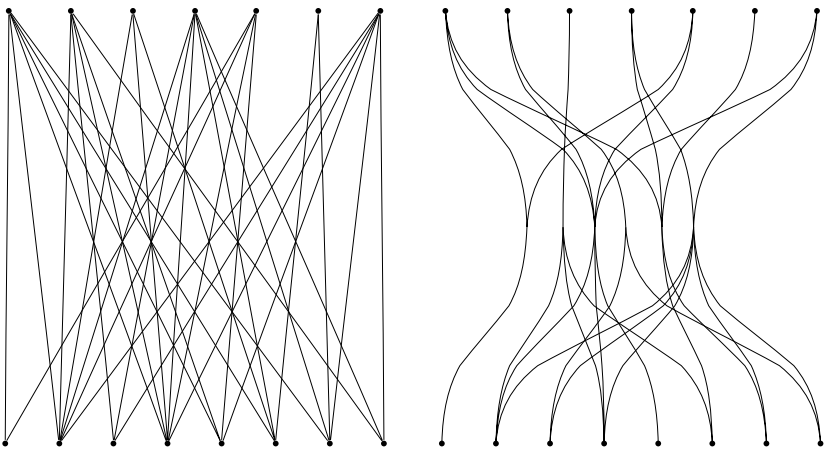


Fig. 1. An example confluent layered drawing.

degrees. Intuitively, a locally-monotone curve is like a single train track, which can make no sharp turns. *Tracks* are the union of locally-monotone curves. They are formed by merging edges together.

A drawing A formed by a collection of tracks on the plane is called a *confluent drawing* for an undirected graph G if and only if

- There is a one-to-one mapping between the vertices in G and A, so that, for each vertex $v \in V(G)$, there is a corresponding vertex $v' \in A$, and all vertices of G are assigned to distinct points in the plane.
- There is an edge (v_i, v_j) in $E(G)$ if and only if there is a locally-monotone curve e' connecting v'_i and v'_j in along tracks in A.

The directed version of a confluent drawing is defined similarly, except that in such a drawing the locally-monotone curves are directed and in every track formed by the union of directed curves, the curves must be oriented consistently.

Self loops and parallel edges of G are not allowed in our definitions, although multiple ways of realizing the same edge are allowed. Namely, for an edge in the original graph, there could be more than one locally monotone path in the drawing corresponding to this edge.

We apply the idea of confluent drawings on layered graphs. Particularly, in the resultant confluent drawing, we replace bicliques in a biclique cover of a two-layer graph $G = (U, L, E)$ by tree-like structures and draw them with smooth curves. As we can see in Fig. 1, our method can greatly reduce the crossings in the drawings of dense bipartite graphs. Additionally, nodes of a biclique can be easily identified by following the smooth curve paths.

Since it is valid to have more than one confluent path between two nodes u and l in the confluent drawing when $(u, l) \in E$, as defined above, it is straightforward that a confluent layered drawing can be obtained by computing a biclique cover C of G, then visualizing each biclique in C as a tree-like structure. We show how to compute a biclique cover of G in the next section.

3 Computing Biclique Cover of Bipartite Graphs

Fishburn and Hammer [9] show that the biclique cover problem is equivalent to a restricted edge coloring problem. This coloring is not much useful for general graphs. However, it has a nice result for triangle-free graphs, and since bipartite graphs belong to the class of triangle-free graphs, an immediate result is that this type of edge coloring can be used to find a biclique cover of a bipartite graph. This result is useful in layered drawing because the edges between any two layers in such a drawing induce a bipartite subgraph.

An edge coloring $c\colon E \leftarrow \{1, 2, \ldots, k\}$ for $G = (V, E)$ is *simply-restricted* if no induced K_3 is monochromatic and the vertex-disjoint edges in an induced P_4 or C_4^c have different colors. Fig. 2 shows the conditions that such induced subgraphs of a simply-restricted edge coloring must satisfy.

Let $d(G)$ denote the bipartite dimension of G, which is the minimum cardinality of a biclique cover of G. Let $\chi_s(G)$ be the chromatic number of a simply-restricted edge coloring of G. $\chi_s(G)$ is 0 if $E = \emptyset$; otherwise, it is the minimum k

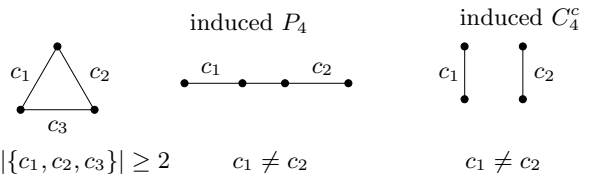


Fig. 2. The required conditions of induced subgraphs of a simply edge coloring.

for which G has a simply-restricted coloring $c: E \leftarrow \{1, 2, \ldots, k\}$. The following theorem states the equivalence of $d(G)$ and $\chi_s(G)$ for triangle-free graphs.

Theorem 1 ***Fishburn and Hammer [9])***
$d(G) = \chi_s(G)$ for every triangle-free graph.

Let E_j be the set of edges with color j in a simply-restricted edge coloring for a triangle-free graph G. As we can see in the second part of the proof of Theorem 1 (omitted here, included in the full version of this paper), E_j is included in the edge set of a biclique subgraph of G. Therefore, every edge set of a single color induces a biclique subgraph of G. By computing a simply restricted edge coloring we can get a biclique cover of G.

Because it is known that the problem of Covering by Complete Bipartite Subgraphs is NP-hard (Garey and Johnson [13] GT18), it is unlikely to have efficient optimization algorithms for finding the minimum biclique cover of a bipartite graph. Thus we only focus on fast heuristics for computing a near-optimal biclique cover.

The simply-restricted edge coloring problem can be transformed into a vertex coloring problem. So, instead of devising a special algorithm for the simply-restricted edge coloring, we can choose to use one of the existing vertex coloring algorithms. Well known heuristic algorithms for vertex coloring include Recursive Largest First (RLF) algorithm of Leighton [18], DSATUR algorithm of Brélaz [2]. For more about heuristics on graph coloring, see Campers et al. [3].

The above method of computing a biclique cover by coloring doesn't distinguish between two kinds of bicliques: $K_{p,1}$ and $K_{1,r}$, where $p, q, r > 1$. So if we are more interested in finding out the set of common callers and callees, we would need to give higher priority to $K_{p,q}$ than $K_{1,r}$ when covering the edges.

After the biclique cover of the two-layer bipartite graph is computed, each biclique in the cover is drawn as a tree-like structure in the final drawing. Doing this repeatedly for every two adjacent layers, we can get the drawings for multi-layer graphs.

The time complexity of the algorithm depends on the coloring heuristic subroutines. For a graph with a set of vertices V, both the RLF algorithm and the DSATUR algorithm run in worst case $O(|V|^3)$ time. There are some other faster coloring heuristics with $O(|V|^2)$ time, but their output qualities are worse. Suppose we have a two-layer bipartite graph $G = (V, E)$. The transformation from the simply-restricted edge coloring into vertex coloring version takes $O(|E|^2)$ time. Using RLF or DSATUR costs $O(|E|^3)$, thus the total time is $O(|E|^3)$.

4 Layout of the Bicliques

We described how to compute a biclique cover of a two-layer bipartite graph in the previous section. Now it is time to show how the bicliques are laid out. In the confluent layered drawings, each biclique in the biclique cover is visualized as a tree-like structure, as in Fig. 1. Now here are the questions. What are the best positions to place the centers of the tree-like structures? How to arrange the curves so that they form confluent tracks defined in Section 2?

4.1 Barycenter Method to Place Centers

In the case where the positions of nodes in the upper level and lower level are fixed, one would like to put the center of a tree to the center of the nodes belonging to the corresponding biclique. For example, in Fig. 3, the drawing on the left is visually better than the drawing on the right. Firstly it has better angular resolution and better edge separation. Secondly it is easier for people to see the biclique as a whole. Then the next question is: what does the center of those nodes mean? In our method, the natural candidate position for a center of the tree-like structure is the barycenter, i.e., the average position, of all the nodes in this biclique.

It looks bad too if these tree centers stay very close to each other. So we need to specify a minimum separation between two centers.

The above requirements can be formulated into constraints:

1. A tree center stays within the range of its leaves.
$$\min_j x_{ij} \leq x_i \leq \max_j x_{ij},$$
where x_i is the x-coordinate of the i^{th} tree center c_i, and x_{ij} is the x-coordinate of the j^{th} leaf of c_i.
2. The distance between any two centers is greater than or equal to the minimum separation.
$$\forall\, i \neq j, \qquad |x_i - x_j| \geq \delta$$
where δ is some pre-specified minimum distance.

Under these constraints, we want a tree center to stay as close as possible to the barycenter of all its leaf nodes. i.e., we want to minimize $\sum_i (x_i - \text{avg}_j(x_{ij}))^2$,

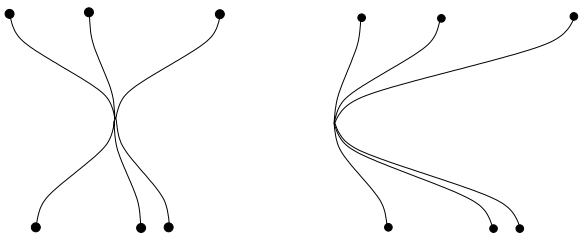


Fig. 3. Good-looking tree and bad-looking tree with centers placed differently.

subject to the above constraints. This is a Quadratic Programming problem, and unfortunately it is NP-hard (Garey and Johnson [13], MP2).

Since it is unlikely to have efficient algorithms for solving this optimization problem, and a small deviation of a tree center from the perfect position won't cause too much displeasure, we use instead a very simple heuristic method to place the tree centers. We first assign to each tree center the x-coordinate of the barycenter of its leaves. Then we sort tree centers by their x-value. The third step is to try to place these tree centers at their x-coordinates one by one. Assume we have k centers to place. Start from the j^{th} center, where $j = \lfloor \frac{k}{2} \rfloor$. Place center j at its barycenter, then try to place centers one by one in the following order: $j-1, j-2, \ldots, 1$. If constraint 2 is violated, the violating center is placed the minimum distance away from the previous placed center. Tree centers to the right of center j are placed similarly in the order of $j+1, j+2, \ldots, k$. It is easy to see that the running time of the barycenter method is dominated by the sorting of the tree centers.

4.2 Placing Tree Centers to Reduce Crossings

Alternatively, one might want to place these centers on positions such that the total number of edge crossings is as few as possible, especially in the case where nodes of upper level and lower level are not fixed. If this is the main concern, we can place the tree centers in another way in order to reduce the edge crossings.

After the biclique cover of a two-layer graph $G = (U, L, E)$ is computed, we construct a new three-layer graph G'. We treat these tree centers as nodes of a middle layer. The set of vertices includes three levels: an upper layer $U' = U$, a middle layer M consisting of tree centers, and a lower layer $L' = L$. The edges of G' are added as follows: for each biclique B_i in the biclique cover, add one edge between the tree center node m_i and each node $u \in U$ that belongs to B_i. Similarly add one edge between m_i and each node $l \in L$ that belongs to B_i.

Now a two-layer graph of the original problem is transformed into a three-layer graph G'. Straight-line crossing reduction algorithms can be applied on G'. After the crossing reduction, we obtain the ordering of nodes in each of the three layers. The orderings will be used to compute the positions of nodes and tree centers in the final drawings. Note that when crossing reduction method is used to place tree centers, it is not always true that a tree center always stays within the x-range of its leaves, i.e., bad centers like the one in Fig. 3 could appear.

Here we are using straight-line edge crossing reduction algorithms for our confluent layered drawings with curve edges. Readers may suspect the equality of the crossing number in the straight-line drawing for the new three-layer graph G' and the crossing number of our curve edge drawings. We will confirm this equality after we describe the generation of curves in the next section.

4.3 Curves

After the positions of tree centers (and the positions of nodes if not given) are computed, we are now ready to place the confluent tracks for the edges.

We use Bézier curves to draw the curve edges in confluent drawings. Given a set of control points $P_1, P_2, \ldots, P_n$, the corresponding Bézier curve is given by

$$C(u) = \sum_{k=0}^{n} P_k \, B_{k,n}(u) \qquad 0 \le u \le 1\,, \tag{1}$$

where $B_{k,n}(u)$ is a Bernstein polynomial

$$B_{k,n}(u) = \frac{n!}{k!\,(n-k)!}\, u^k (1-u)^{n-k}\,. \tag{2}$$

Bézier curves have some nice properties that are suitable for our confluent tracks. The first property is that a Bézier curve always passes its first and last control point. The second is that a Bézier curve always stays within the convex hull formed by its control points. In addition, the tangents of a Bézier curve at the endpoints are $P_1 - P_0$ and $P_n - P_{n-1}$. Thus it is easy to connect two Bézier curves while still maintaining the first order continuity: just let $P_n = P'_0$ and let the control points P_{n-1},$P_n = P'_0$,and P'_1 co-linear.

The confluent track between each node and the tree center is drawn as a Bézier curve. In our program we use cubic Bézier curves ($n = 4$ in Equation 2). Each such a curve has four control points, chosen as shown in Fig. 4.

More formally, assume we are given the following input for a biclique B_i: y_u,y_l, and y_c are the y-coordinates of the upper, lower, and tree center levels, respectively. x_i is the x-coordinate of the tree center for B_i. x_{ij}'s are the x-coordinates of nodes in biclique B_i. Let Δy be a distance parameter that controls the shape of the curve edges. When node j is in the upper level, the four control points are $P_0 = (x_{ij}, y_u)$, $P_1 = (x_{ij}, y_u + \Delta y)$, $P_2 = (x_i, y_c - \Delta y)$, and $P_3 = (x_i, y_c)$. When node j is in the lower level, the four control points are $P_0 = (x_i, y_c)$, $P_1 = (x_i, y_c + \Delta y)$, $P_2 = (x_{ij}, y_l - \Delta y)$, and $P_3 = (x_{ij}, y_l)$.

From Equation 1, it is not hard to verify that in a confluent layered drawing, two Bézier curves cross each other if and only if the corresponding straight-line

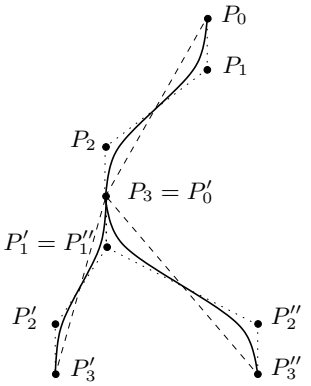


Fig. 4. Bézier curves.

edges (dashed lines in Fig. 4) of the bicliques cross each other, given that the control points are chosen as above. This should clear the doubt that appears at the end of Section 4.2.

5 Multi-depth Confluent Layered Drawings

So far we have introduced the method of confluent layered drawings: replacing subsets of edges in a two-layer graph by tree-like structures. This method can be extended to obtain drawings that display richer information. The extended drawings are called *multi-depth confluent layered drawings*.

The idea is as follows: after the biclique cover for a two-layer graph $G = (U, L, E)$ is computed, the tree center nodes are viewed as a middle layer M, and a new three layer graph $G' = (U, M, L, E')$ is constructed as in Section 4.2. The same biclique cover algorithm is then applied to G' twice, once for the subgraph induced by $U \cup M$; once for the subgraph induced by $M \cup L$. By applying this approach recursively, we get biclique covers at different depth. In the final drawing, only biclique covers at the largest depth are replaced by sets of tree-like structures. The final drawing is a multi-depth confluent layered drawing. The drawings discussed before this section are all *depth-one (confluent layered) drawings*.

In a depth-one drawing, we compute a biclique cover and lay out the biclique cover. In general, for a depth-i drawing, we need to compute $2^i - 1$ biclique covers and 2^{i-1} biclique covers are laid out.

An example drawing of depth-two is shown in Fig. 5.

Because the control points for the Bézier curves are chosen in a way such that the tangents at the endpoints of the Bézier curves are all vertical, it is guaranteed that all segments of a path are connected seamlessly and smoothly in

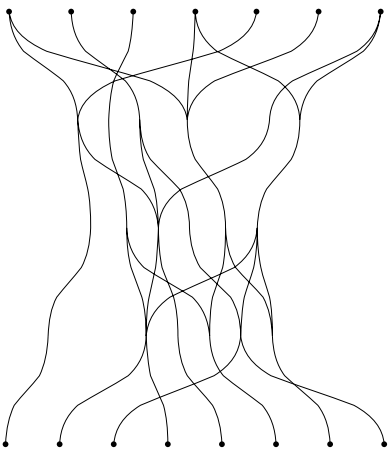


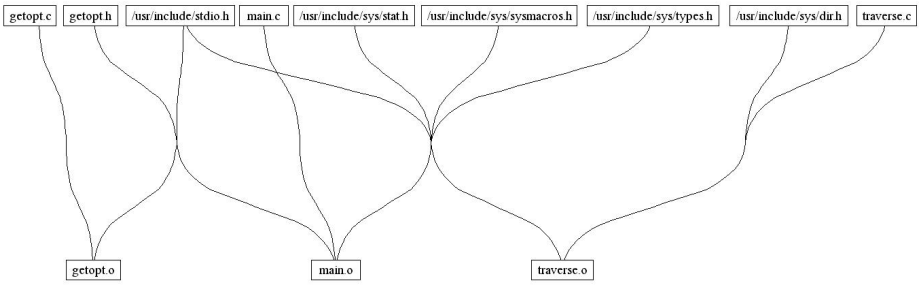
Fig. 5. Depth-two confluent drawing (on same input as the drawing of Fig. 1).

multi-depth drawings. Readers probably have already noticed some wavy edges in the drawing of Fig. 5. It is because a single edge biclique ($K_{1,1}$) is also drawn as two Bézier curves. We offer an option in our program to do a simple treatment for these single edge bicliques: draw them as a single Bézier curves instead of two. But after this special treatment is applied, the crossing property is not preserved any more. That means two curve segments could have crossing(s), even though their corresponding edges in G' don't cross each other in a straight-line drawing.

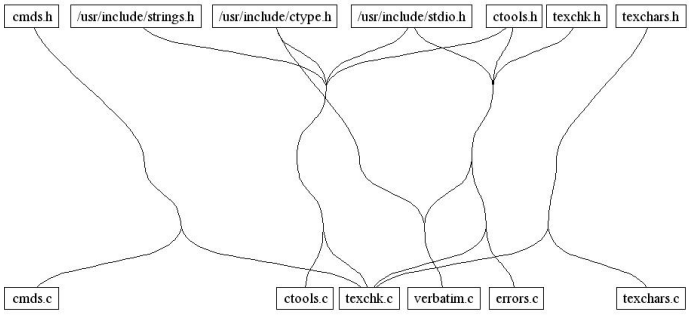
Multi-depth drawings may further reduce the number of crossings. They also show a richer structure than the depth-one drawings, which only display bicliques. For example we can observe relationships between bicliques in a depth-two confluent layered drawing. However higher depth requires more computations of biclique covers, and generates more dummy centers. The former leads to the increasing of time and space complexity, while the latter could result in a more complicate confluent drawing. We feel that drawings with depth higher than two are not very practically useful.

6 Real-World Examples

We list two example drawings of real-world graphs in Fig. 6. We implemented the algorithm of computing biclique cover using the RLF heuristic. For the center placement we implemented the barycenter method. We assume that besides the



(a) "Derives" relation for the Shar program



(b) "Includes" relation for the Texchk program

Fig. 6. Confluent drawing for examples of Newbery [21].

two-layer graph, the input also includes the positions of (fixed) nodes in upper and lower levels (possibly output by other algorithms that take labels and other information into account.) The result drawing is written into a file of DOT format [17]. The `neato` program in the Graphviz package [7] is then used to generate the graphic file in a desired format. Fig. 6 (a) is a depth-one drawing. Fig. 6 (b) is a depth-two drawing with the special smoothing treatment applied.

7 Conclusions and Acknowledgments

In this paper we introduce a new method – confluent layered drawing, for visualizing layered graphs. It combines the layered drawing technique with the relaxed confluent drawing approach. There are still interesting open problems, e.g., how to test whether a layered graph has a crossing-free confluent layered drawing? How to minimize the crossing of the drawing among all possible biclique covers? It is also useful to investigate better ways for visualizing confluent tracks.

We would like to thank anonymous referees for their helpful comments.

References

1. W. Barth, M. Jünger, and P. Mutzel. Simple and efficient bilayer crossing counting. In M. Goodrich and S. Kobourov, editors, *Graph Drawing (Proc. GD '02)*, volume 2528 of *Lecture Notes Comput. Sci.*, pages 130–141. Springer-Verlag, 2002.
2. Brélaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22(4):251–256, 1979.
3. G. Campers, O. Henkes, and J. P. Leclerq. Graph coloring heuristics: A survey, some new propositions and computational experiences on random and "Leighton's" graphs. In G. K. Rand, editor, *Operational research '87 (Buenos Aires, 1987)*, pages 917–932. North-Holland Publishing Co, 1988.
4. M. J. Carpano. Automatic display of hierarchized graphs for computer aided decision analysis. *IEEE Trans. Syst. Man Cybern.*, SMC-10(11):705–715, 1980.
5. M. Dickerson, D. Eppstein, M. T. Goodrich, and J. Y. Meng. Confluent drawing: Visualizing nonplanar diagrams in a planar way. In G. Liotta, editor, *Graph Drawing (Proc. GD '03)*, volume 2912 of *Lecture Notes Comput. Sci.*, pages 1–12. Springer-Verlag, 2003.
6. P. Eades and K. Sugiyama. How to draw a directed graph. *J. Inform. Process.*, 13:424–437, 1991.
7. J. Ellson, E. Gansner, E. Koutsofios, and S. North. Graphviz. URL: `http://www.research.att.com/sw/tools/graphviz`.
8. T. Eschbach, W. Günther, R. Drechsler, and B. Becker. Crossing reduction by windows optimization. In M. Goodrich and S. Kobourov, editors, *Graph Drawing (Proc. GD '02)*, volume 2528 of *Lecture Notes Comput. Sci.*, pages 285–294. Springer-Verlag, 2002.
9. P. C. Fishburn and P. L. Hammer. Bipartite dimensions and bipartite degree of graphs. *Discrete Math.*, 160:127–148, 1996.
10. M. Forster. Applying crossing reduction strategies to layered compound graphs. In M. Goodrich and S. Kobourov, editors, *Graph Drawing (Proc. GD '02)*, volume 2528 of *Lecture Notes Comput. Sci.*, pages 276–284. Springer-Verlag, 2002.

11. E. R. Gansner, E. Koutsofios, S. C. North, and K. P. Vo. A technique for drawing directed graphs. *IEEE Trans. Softw. Eng.*, 19:214–230, 1993.
12. E. R. Gansner, S. C. North, and K. P. Vo. DAG – A program that draws directed graphs. *Softw. – Pract. Exp.*, 18(11):1047–1062, 1988.
13. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, 1979.
14. M. R. Garey and D. S. Johnson. Crossing number is NP-complete. *SIAM J. Algebraic Discrete Methods*, 4(3):312–316, 1983.
15. D. J. Gschwind and T. P. Murtagh. A recursive algorithm for drawing hierarchical directed graphs. Technical Report CS-89-02, Department of Computer Science, Williams College, 1989.
16. M. Jünger and P. Mutzel. 2-layer straightline crossing minimization: Performance of exact and heuristic algorithms. *J. Graph Algorithms Appl.*, 1(1):1–25, 1997.
17. E. Koutsofios and S. North. Drawing graphs with *dot*. Technical report, AT&T Bell Laboratories, Murray Hill, NJ., 1995. Available from `http://www.research.bell-labs.com/dist/drawdag`.
18. F. T. Leighton. A graph coloring algorithm for large scheduling problems. *Journal of Research of National Bureau of Standard*, 84:489–506, 1979.
19. E. B. Messinger. Automatic layout of large directed graphs. Technical Report 88-07-08, Department of Computer Science, University of Washington, 1988.
20. E. B. Messinger, L. A. Rowe, and R. H. Henry. A divide-and-conquer algorithm for the automatic layout of large directed graphs. *IEEE Trans. Syst. Man Cybern.*, SMC-21(1):1–12, 1991.
21. F. J. Newbery. Edge concentration: A method for clustering directed graphs. In *Proc. 2nd Internat. Workshop on Software Configuration Management*, pages 76–85, 1989.
22. F. Newbery Paulisch and W. F. Tichy. EDGE: An extendible graph editor. *Softw. – Pract. Exp.*, 20(S1):63–88, 1990. also as Technical Report 8/88, Fakultat fur Informatik, Univ. of Karlsruhe, 1988.
23. M. Newton, O. Sýkora, and I. Vrt'o. Two new heuristics for two-sided bipartite graph drawing. In M. Goodrich and S. Kobourov, editors, *Graph Drawing (Proc. GD '02)*, volume 2528 of *Lecture Notes Comput. Sci.*, pages 312–319. Springer-Verlag, 2002.
24. H. Purchase. Which aesthetic has the greatest effect on human understanding? In G. Di Battista, editor, *Graph Drawing (Proc. GD '97)*, volume 1353 of *Lecture Notes Comput. Sci.*, pages 248–261. Springer-Verlag, 1997.
25. K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical systems. *IEEE Trans. Syst. Man Cybern.*, SMC-11(2):109–125, 1981.
26. V. Waddle and A. Malhotra. An E log E line crossing algorithm for levelled graphs. In J. Kratochvíl, editor, *Graph Drawing (Proc. GD '99)*, volume 1731 of *Lecture Notes Comput. Sci.*, pages 59–71. Springer-Verlag, 1999.