Algorithms for Drawing Graphs: an Annotated Bibliography*

Giuseppe Di Battista
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria, 113 — Roma
00198 Italy
`dibattista@iasi.rm.cnr.it`

Peter Eades
Department of Computer Science
University of Newcastle
University Drive — Callaghan
NSW 2308, Australia
`eades@cs.newcastle.edu.au`

Roberto Tamassia
Department of Computer Science
Brown University
Providence, RI 02912–1910
USA
`rt@cs.brown.edu`

Ioannis G. Tollis
Department of Computer Science
The University of Texas at Dallas
Richardson, Texas 75083–0688
USA
`tollis@utdallas.edu`

June 1993

Abstract

Several data presentation problems involve drawing graphs so that they are easy to read and understand. Examples include circuit schematics and diagrams for information systems analysis and design. In this paper we present a bibliographic survey on algorithms whose goal is to produce aesthetically pleasing drawings of graphs. Research on this topic is spread over the broad spectrum of Computer Science. This bibliography constitutes an attempt to encompass both theoretical and application oriented papers from disparate areas.

*** Please send comments and updates to the authors. ***

*This document is available via anonymous `ftp` from `wilma.cs.brown.edu (128.148.33.66)`, files `/pub/gdbiblio.tex.Z` and `/pub/gdbiblio.ps.Z`.

Contents

1 Introduction

A number of data presentation problems involve the drawing of a graph on a 2-dimensional surface. Examples include circuit schematics, algorithm animation, and diagrams for information systems analysis and design. In this paper we present a bibliographic survey on algorithms whose goal is to produce clear and readable drawings of graphs.

Various *graphic standards* have been proposed for the representation of graphs in the plane. Usually, the vertices are represented by symbols such as circles or boxes, and each edge (u, v) is represented by a simple open curve between the symbols associated with the vertices u and v.

A drawing such that each edge is represented by a polygonal chain is a *polyline* drawing (see Fig. 1). There are two common special cases of this standard. A *straight-line* drawing maps each edge into a straight-line segment (see Fig. 2). This standard is commonly adopted in graph theory texts. An *orthogonal* drawing maps each edge into a chain of horizontal and vertical segments (see Fig. 3). Entity Relationship diagrams in data base design are usually drawn according to this standard. Note that polyline drawings can be modified to give drawings with nicely curved edges. A drawing is *planar* if no two edges intersect. A polyline drawing is a *grid* drawing if the vertices and the bends of the edges have integer coordinates.

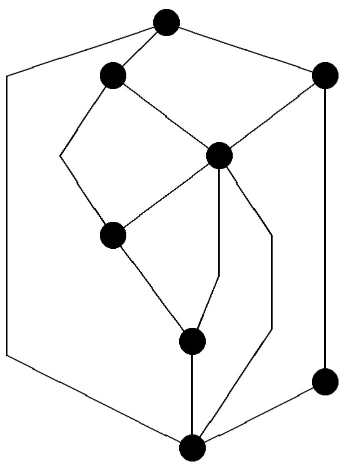


Figure 1: Polyline drawing.

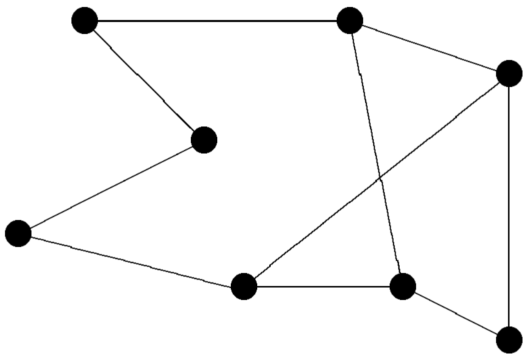


Figure 2: Straight-line drawing.

A *graph drawing algorithm* reads as input a combinatorial description of a graph G, and produces as output a drawing of G according to a given graphic standard. The drawing is described in terms of graphics primitives such as *DRAW_LINE* and *FILL_CIRCLE*, which can be interpreted on a physical graphics device.

Within a graphic standard, a graph has infinitely many different drawings. However, in almost all data presentation applications, the usefulness of a drawing of a graph depends on its readability, that is, the capability of conveying the meaning of the diagram quickly and clearly. Readability issues are expressed by means of *aesthetics*, which can be formulated as optimization goals for the drawing algorithms. In general, the aesthetics depend on the graphic standard adopted and the particular class of graphs of interest. A fundamental and classical aesthetic is the minimization of crossings between edges. In polyline drawings it is desirable to avoid bends in edges. In grid drawings, the area of the smallest rectangle covering the drawing should be minimal. In all graphic standards, the display of symmetries is desirable. It should be noted that aesthetics are subjective and may need to be tailored to suit personal preferences, traditions and culture. For example, although the cube graph is planar, it is traditionally drawn with crossing edges, as shown in Fig. 4.

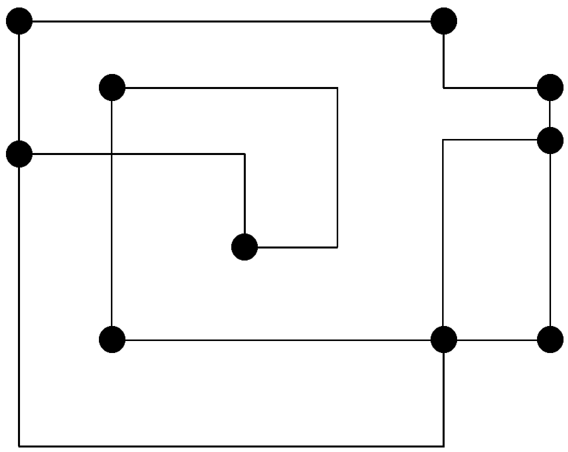


Figure 3: Orthogonal drawing.

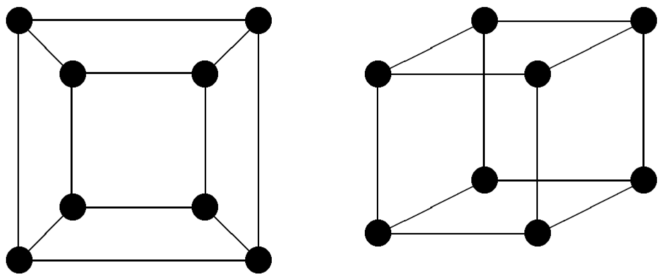


Figure 4: Two drawings of the cube graph.

Research on graph drawing algorithms is spread over the broad spectrum of Computer Science, from VLSI to data base design. This bibliography constitutes a first attempt to encompass both theoretical and application oriented papers from disparate areas. However, we do not consider layout algorithms (such as some VLSI layout techniques) that have no impact on the problem of producing aesthetically pleasing drawings. As indicated in the title, this bibliography concentrates on *algorithms* for drawing graphs. It is written from a Computer Science viewpoint, and does not deal with other aspects of the problem of drawing graphs. Namely, we do not attempt to cover the large literature on the mathematical theory of embeddings of graphs, work on circuit and facilities layout, or psychological and philosophical issues of aesthetically pleasing drawings. We have omitted many papers which describe graphic user interfaces and visualization systems; although these often use graph drawings, few currently have automatic layout facilities. However, introductory textbooks on graphs and algorithms, and a few significant papers from related areas have been included for the reader's convenience.

In Section 2 we mention background reference material for graph drawing problems. Sections 3, 4, 5, and 6 consider in turn algorithms for drawing trees, general graphs, planar graphs and directed graphs. Literature on systems which use graph layout algorithms is outlined in Section 7. Papers on topics that do not fit the above classification are mentioned in Section 8. A list of significant open problems is given in Section 9. An Index of authors is provided at the end of the paper.

Throughout the paper n and m denote the number of vertices and edges of the graph currently being considered.

2 Background

For elementary graph theory, the following textbooks may be consulted

1. J.A. Bondy and U.S.R. Murty, Graph Theory with Applications, North Holland, Amsterdam, 1976.

2. F. Harary, Graph Theory, Addison-Wesley, Reading, MA, 1969.

Fundamentals of data structures and algorithms are described in

3. E. Horowitz and S. Sahni, Fundamentals of Data Structures, Computer Science Press, 1983.

4. T.H. Cormen, C.E. Leiserson, and R.L. Rivest, Introduction to Algorithms, The MIT Press, 1990.

5. E.M. Reingold, J. Nievergelt, and N. Deo, Combinatorial Algorithms: Theory and Practice, Prentice Hall, 1977.

Algorithms for graph problems and applications are described in

6. S. Even, Graph Algorithms, Computer Science Press, Potomac, MD, 1979.

7. A. Gibbons, Algorithmic Graph Theory, Cambridge University Press, Cambridge, Great Britain, 1985.

8. J.A. McHugh, Algorithmic Graph Theory, Prentice Hall, Englewood Cliffs, NJ, 1990.

9. R.E. Tarjan, "Data Structures and Network Algorithms," CBMS-NSF Regional Conference Series in Applied Mathematics, vol. 44, Society for Industrial Applied Mathematics, 1983.

Algorithms for planar graphs are presented in:

10. T. Nishizeki and N. Chiba, Planar Graphs: Theory and Algorithms, Annals of Discrete Mathematics 32, North-Holland, 1988.

Concepts and applications of NP-completeness and complexity theory are described in

11. M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, 1979.

Basic concepts of computer graphics and computational geometry are given in

12. J.D. Foley and A. van Dam, Fundamentals of Interactive Computer Graphics, Addison-Wesley, 1982.

13. F.P. Preparata and M.I. Shamos, Computational Geometry, Springer-Verlag, New York, 1985.

Two previous versions of this bibliography have appeared as:

14. P. Eades and R. Tamassia, "Algorithms for Drawing Graphs: An Annotated Bibliography," Technical Report 82, Department of Computer Science, University of Queensland 1987.

15. P. Eades and R. Tamassia, "Algorithms for Drawing Graphs: An Annotated Bibliography," Technical Report CS-09-89, Department of Computer Science, Brown University 1989.

3 Trees

3.1 Rooted Trees

Rooted trees are often used to represent hierarchies such as family trees, organization charts, and search trees. Planar straight-line drawings and orthogonal polyline drawings are commonly used to represent rooted trees (see Fig. 5). The following additional aesthetics are often adopted.

- Vertices are placed along horizontal lines according to their level (graph-theoretic distance from the root).
- There is a minimum separation distance between two consecutive vertices on the same level.
- The width of the drawing is as small as possible.

Further, for ordered binary trees such as search trees, we require:

- The left and right children of each vertex v are placed to the left and right of v, respectively.

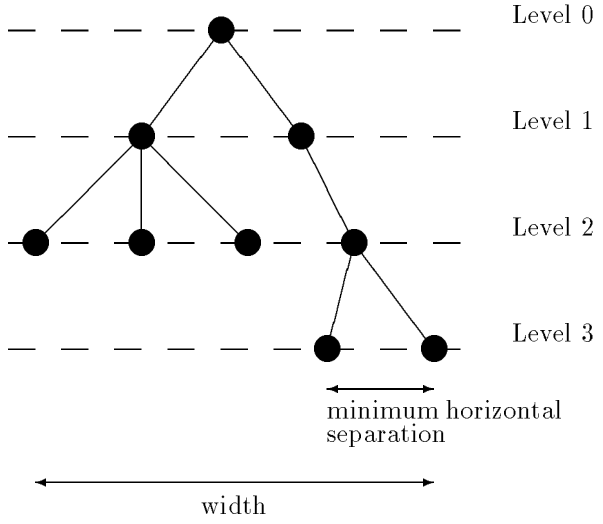


Figure 5: Drawings of a rooted tree in straight line and polyline orthogonal standards.

The following papers contain heuristics for drawing rooted trees that address the above aesthetics. Additional aesthetics, such as centering each parent upon its children, and generating congruent drawings for isomorphic subtrees, are also investigated.

16. R.E. Sweet, "Empirical Estimates of Program Entropy," Technical Report STAN-CS-78-698, Stanford Univ., Stanford, CA, 1978.

17. C. Wetherell and A. Shannon, "Tidy Drawing of Trees," IEEE Trans. on Software Engineering, vol. SE-5, no. 5, pp. 514-520, 1979.

18. J. Vaucher, "Pretty Printing of Trees," Software Practice and Experience, vol. 10, no. 7, pp. 553-561, 1980.

19. E. Reingold and J. Tilford, "Tidier Drawing of Trees," IEEE Trans. on Software Engineering, vol. SE-7, no. 2, pp. 223- 228, 1981.

20. J.S. Tilford, "Tree Drawing Algorithms," Technical Report UIUCDCS-R-81-1055, Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign, 1981.

21. J.Q. Walker II, "A Node-positioning Algorithm for General Trees," Software Practice and Experience, vol. 20, no. 7, pp. 685-705, 1990.

Implementation details of a variation of the algorithm by Reingold and Tilford [19] are discussed by Bruggemann-Klein and Wood; the paper presents a set of *TeX* macros to implement the algorithm.

22. A. Bruggemann-Klein and D. Wood, "Drawing Trees Nicely with TEX," Manuscript, Department of Computer Science, University of Waterloo, 1987.

The extension of these algorithms to rooted trees with arbitrary vertex degree is straightforward. The algorithms give aesthetically acceptable drawings. However, Supowit and Reingold show that they can produce drawings much wider than necessary.

23. K.J. Supowit and E.M. Reingold, "The Complexity of Drawing Trees Nicely," Acta Informatica, vol. 18, pp. 377-392, 1983.

This paper addresses the problem of constructing a minimum width drawing of a binary tree such that parents are centered upon their children and isomorphic subtrees are congruent. This problem is NP-complete if a grid drawing is required, but otherwise polynomially solvable by linear programming techniques.

The area requirement of straight-line and polyline grid drawings of binary and rooted trees is investigated in:

24. P. Crescenzi, G. Di Battista, and A. Piperno "A Note on Optimal Area Algorithms for Upward Drawings of Binary Trees," *Computational Geometry: Theory and Applications*, vol. 2, pp. 187-200, 1992.

25. A. Garg, M.T. Goodrich and R. Tamassia, "Area-Efficient Upward Tree Drawings," Proc. ACM Symp. on Computational Geometry, pp. 359-368, 1993.

Three drawing conventions that are appealing for their practical applicability are investigated in:

26. P. Eades, T. Lin, and X. Lin, "Two Tree Drawing Conventions," Technical Report No. 174, Key Centre for Software Technology, Department of Computer Science, The Univ. of Queensland, 1990. (to appear in *Computational Geometry and Applications*).

27. P. Eades, T. Lin, and X. Lin, "Minimum Size h-v Drawings," *Advanced Visual Interfaces* (Proceedings of AVI 92), World Scientific Series in Computer Science Volume 36, 386-394.

In the *inclusion* convention nodes are represented by boxes and parent-child relationships are represented by inclusion of one box in another. The *tip-over* convention is similar to the classical one, however, children of some nodes may be arranged vertically rather than horizontally. An h-v drawing is similar to a tip-over drawing. Examples of inclusion and tip-over drawings are in Fig. 6.

3.2 Free Trees

Surprisingly little work has been published on drawing "free" trees, that is, trees that do not represent hierarchies and have no specific root. However, the above algorithms for rooted trees can be modified to produce acceptable *radial* drawings of free trees by arranging the vertices of each level on a concentric circle about the graphtheoretic center of the tree.

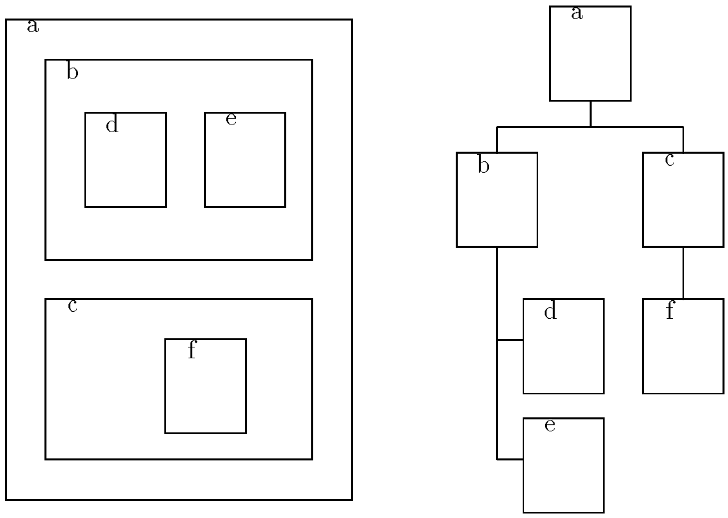
Folklore on radial and other simple drawings of free trees is summarized in

28. P.D. Eades, "Drawing Free Trees," Bulletin of the Institute for Combinatorics and its Applications, Vol. 5, 1992, 10-36.

Strategies for constructing radial drawings are described in

29. M.A. Bernard, "On the Automated Drawing of Graphs," Prc. 3rd Caribbean Conf. on Combinatorics and Computing, pp. 43-55, 1981.

30. T. Kamada, Visualizing Abstract Objects and Relations, World Scientific, 1989.



(a) (b)

Figure 6: Inclusion and tip-over conventions.

The following paper shows how to display symmetries in radial drawings.

31. J. Manning and M.J. Atallah, "Fast Detection and Display of Symmetry in Trees," Congressus Numerantium, vol. 64, pp. 159-169, 1988.

Bhatt and Cosmodakis show that it is NP-complete to construct an orthogonal grid drawing of a tree such that the maximum edge length is minimized:

32. S. Bhatt and S. Cosmadakis, "The Complexity of Minimizing Wire Lengths in VLSI Layouts," Information Processing Letters, vol. 25, pp. 263-267, 1987.

The techniques of Bhatt and Cosmodakis are refined and extended in the following:

33. F.J. Bradenburg, "Nice Drawings of Graphs and Trees are Computationally Hard," Technical Report MIP-8820, Fakultät für Mathematik und Informatik, Univ. Passau, 1988.

34. A. Gregori, "Unit Length Embedding of Binary Trees on a Square Grid," Information Processing Letters, vol. 31, pp. 167-172, 1989.

35. P. J. Idicula, "Drawing Trees in Grids," Masters Thesis, Department of Computer Science, University of Auckland 1990.

4 General Graphs

There are several aesthetics for obtaining attractive drawings of general undirected graphs. The main such aesthetics are:

- display symmetry;
- avoid edge crossings;
- avoid bends in edges;
- keep edge lengths uniform;

- distribute vertices uniformly.

In general, the optimization problems associated with these aesthetics are NP-hard. Several complexity results are reported in:

36. D.S. Johnson, "The NP-Completeness Column: an Ongoing Guide," J. of Algorithms, vol. 3, no. 1, pp. 89-99, 1982.

37. D.S. Johnson, "The NP-Completeness Column: an Ongoing Guide," J. of Algorithms, vol. 5, no. 2, pp. 147-160, 1984.

Many problems are NP-hard even for restricted classes of graphs, such as trees and planar graphs. Specific results are presented in [23, 32, 33, 34, 35, 112, 26] and:

38. M.R. Garey and D.S. Johnson, "Crossing Number is NP-Complete," SIAM J. Algebraic and Discrete Methods, vol. 4, no. 3, pp. 312-316, 1983.

39. M.R. Kramer and J. van Leeuwen, "The Complexity of Wire-Routing and Finding Minimum Area Layouts for Arbitrary VLSI Circuits", in Advances in Computing Research, vol. 2, ed. F.P. Preparata, pp. 129-146, JAI Press, Greenwich, CT, 1984.

40. Z. Miller and J.B. Orlin, "NP-Completeness for Minimizing Maximum Edge Length in Grid Embeddings", J. Algorithms, vol. 6, pp. 10-16, 1985.

Besides time complexity limitations, the above aesthetics are also "competitive" in that the optimality of one often prevents the optimality of others. Because of such difficulties, general approaches to graph drawing are usually heuristic.

4.1 Straight-Line Drawings

A model for measuring the symmetry of a straight-line drawing of a graph is given in:

41. R. Lipton, S. North, and J. Sandberg, "A Method for Drawing Graphs", Proc. ACM Symp. on Computational Geometry, pp. 153-160, 1985.

This paper also proposes an algorithm for constructing a straight-line drawing of a graph with as much symmetry as possible; however the algorithm requires the solution of the apparently intractable problem of computing the automorphism group of a graph. A completely different approach to symmetry display (which avoids computing automorphisms) is described in:

42. P. Eades, "A Heuristic for Graph Drawing," Congressus Numerantium, vol. 42, pp. 149-160, 1984.

This algorithm, called *spring embedder*, is a heuristic based on a physical model. The straight-line standard is adopted. The drawing process is to simulate a mechanical system, where vertices are replaced by rings, and edges are replaced by springs. The springs attract the rings if they are too far apart, and repel them if they are too close.

The algorithms of [86, 87, 91, 92] may be viewed as spring algorithms with the positions of some of the vertices fixed; although originally designed for planar graphs, they may be applied to nonplanar graphs with reasonable results.

Other algorithms of a similar "force directed" nature are described in [30] and

43. J.E. Cuny, D.A. Bayley, J.W. Hagerman, and A.A. Hough, "The Simple Simon Programming Environment: A Status Report", Technical Report 87-22, Dept. of Computer and Information Science, Univ. of Massachusetts, Amherst, MA, May 1987.

44. T. Kamada and S. Kawai, "Automatic Display of Network Structures for Human Understanding", Technical Report 88-007, Dept. of Information Science, Univ. of Tokyo, 1988.

45. T. Kamada, "On Visualization of Abstract Objects and Relations," Ph.D. dissertation, Dept. of Information Science, Univ. of Tokyo, 1988.

46. T. Kamada and S. Kawai, "An Algorithm for Drawing General Undirected Graphs", Information Processing Letters, vol. 31, pp. 7-15, 1989.

47. T. Kamada, "Symmetric Graph Drawing by a Spring Algorithm and its Applications to Radial Drawing", Manuscript, Dept. of Information Science, Univ. of Tokyo, 1989.

48. T. Fruchterman and E. Reingold, "Graph Drawing by Force-Directed Placement," Software-Practice and Experience, vol. 21, no. 11, pp. 1129-1164, 1991.

A general model for spring algorithms is defined in [175]; this thesis also attempts to explain mathematically the apparent connection between spring algorithms and symmetrical drawings.

An extension of the spring approach is presented by Davidson and Harel. An energy function is defined in terms of the desired aesthetics: for instance, the number of edge crossings plus the closeness of vertices. A layout of minimal energy (an thus maximal beauty according to the energy function) is obtained by simulated annealing.

49. R. Davidson and D. Harel, "Drawing Graphs Nicely Using Simulated Annealing", Technical Report CS 89–13, Department of Applied Mathematics and Computer Science, The Weizmann Institute of Science, Rehovot, 1989.

An algorithm based on multidimensional scaling (a standard statistical method) that finds a placement of vertices with euclidean distances that approximate the graph-theoretic distances is presented in

50. J.B. Kruskal and J.B. Seery, "Designing Network Diagrams," Proc. First General Conference on Social Graphics, U. S. Department of the Census, Washington, D.C., pp. 22-50, July 1980.

An algorithm that uses several heuristics to obtain "near-optimal" drawings is presented by Tunkelang. The heuristics improve on existing approaches by focusing on three aspects of the graph drawing problem: computation of the aesthetic cost of a drawing, order of node placement, and local optimization techniques. The algorithm and comparison with the techniques of [48] and [49] are described in:

51. D. Tunkelang, "An Aesthetic Layout Algorithm for Undirected Graphs," M.S. Thesis, Department of Electrical Engineering and Computer Science, MIT, 1992.

A simple heuristic for constructing straight-line drawings which adds one vertex at a time is described in:

52. H. Watanabe, "Heuristic Graph Displayer for G-BASE," Technical Report no. 17, Ricoh Software Research Center, Tokyo, Japan, 1988.

Mäkinen considers straight-line drawings with vertices placed along the circumference of a circle. He shows that several related optimization problems are NP-complete and gives a heuristic for reducing the maximum edge length.

53. E. Mäkinen, "On Circular Layouts," Int. Journal of Computer Mathematics, vol. 24, pp. 29-37, 1988.

4.2 Planarization

As discussed above, most of the techniques for drawing general undirected graphs are heuristics based on various types of simulation. Given the wealth of techniques available for drawing planar graphs, a sensible strategy for drawing a nonplanar graph is to first *planarize* the graph, and then apply a planar graph drawing algorithm. The term *planarization* is used for several related problems. In general, planarization seeks to transform a nonplanar graph into a planar graph with a small number of well defined operations.

The most common planarization operation is *edge deletion*: one must find a small number of edges whose deletion yields a planar graph. This is equivalent to finding a planar subgraph with a large number of edges. Finding a planar subgraph with a maximum number of edges is NP-hard. However, a maximal planar subgraph can be found efficiently:

54. J. Cai, X. Han, and R.E. Tarjan, "An $O(m \log n)$-time Algorithm for the Maximal Subgraph Problem," SIAM J. on Computing, 1993 (to appear).

Heuristics for finding a maximum planar subgraph and algorithms for finding a maximal planar subgraph are presented in:

55. M. Marek-Sadowska, "Planarization Algorithms for Integrated Circuits Engineering", Proc. IEEE Int. Symp. on Circuits and Systems, pp. 919-923, 1978.

56. N. Chiba, I. Nishioka, and I. Shirakawa, "An Algorithm of Maximal Planarization of Graphs", Proc. IEEE Int. Symp. on Circuits and Systems, pp. 649-652, 1979.

57. E. Nardelli and M. Talamo, "A Fast Algorithm for Planarization of Sparse Diagrams", Technical Report R.105, IASI-CNR, Rome, 1984.

58. T. Ozawa and H. Takahashi, "A Graph-planarization Algorithm and its Applications to Random Graphs", Graph Theory and Algorithms, Lecture Notes in Computer Science, vol. 108, pp. 95-107, 1981.

59. R. Jayakumar, K. Thulasiraman, and M.N.S. Swamy, "On Maximal Planarization of Nonplanar Graphs", IEEE Trans. Circuits and Systems, vol. CAS-33, no. 8, 843–854, 1986.

60. R. Jayakumar, K. Thulasiraman, and M.N.S Swamy, "$O(n^2)$ Algorithms for Graph Planarization," Technical Report CSD-88-01, Dept. Computer Science, Concordia Univ., 1988.

61. G. Kant, "An $O(n^2)$ Maximal Planarization Algorithm based on PQ-trees," Technical Report RUU-CS-92-03, Department of Computer Science, Utrecht Univ., The Netherlands, 1992.

62. L.R. Foulds, P.B. Gibbons, and J.W. Giffin, "Graph Theoretic Heuristics for the Facilities Layout Problem: An Experimental Comparison", Operations Research, 1985.

63. P. Eades, L. Foulds, and J. Giffin, "An Efficient Heuristic for Identifying a Maximal Weight Planar Subgraph", Combinatorial Mathematics IX, Lecture Notes in Mathematics, vol. 952, pp. 239-251, 1982.

Another planarization technique is to find a drawing with the minimum number of crossings. Again, this problem is NP-hard [38]. Heuristics for crossing minimization are given in:

64. D. Ferrari and L. Mezzalira, "On Drawing a Graph with the Minimum Number of Crossings", Technical Report n. 69-11, Istituto di Elettrotecnica ed Elettronica, Politecnico di Milano, 1969.

A new technique for planarization is "splitting". The *splitting* operation is to make two copies of a vertex and share the neighbors between the two copies. This technique is used in manual layout to simplify complex graphs. A *minimum splitting sequence* is a minimum length sequence of splittings which makes the graph planar. Heuristics for finding a minimum splitting sequence are discussed in

65. C.X. Mendonça, "Heuristics for Planarization by Vertex Splitting", manuscript, University of Newcastle 1992.

The topological equivalence among nonplanar drawings of a graph is studied in:

66. R.B. Eggleton, "Rectilinear Drawings of Graphs," Utilitas Mathematica, vol. 29, pp. 146-172, 1986.

There is an extensive mathematical literature on crossing numbers of graphs, see the following papers for references:

67. H. Harborth and I. Mengersen, "Edges Without Crossings in Drawings of Complete Graphs", J. Combinatorial Theory (B), vol. 17, no. 3, pp. 229-311, 1974.

68. R.K. Guy, "Crossing Numbers of Graphs," Graph Theory and Applications, Lecture Notes in Mathematics, vol. 303, pp. 111-124, 1972.

4.3 Polyline Drawings

A comprehensive approach to the construction of orthogonal grid drawings, based on a number of graph algorithms, is presented in:

69. C. Batini, E. Nardelli, M. Talamo, and R. Tamassia, "A Graphtheoretic Approach to Aesthetic Layout of Information Systems Diagrams", Proc. 10th Int. Workshop on Graphtheoretic Concepts in Computer Science (Berlin, June 1984), pp. 9-18, Trauner Verlag, 1984.

70. R. Tamassia, G. Di Battista, and C. Batini, "Automatic Graph Drawing and Readability of Diagrams", IEEE Transactions on Systems, Man and Cybernetics, vol. SMC-18, no. 1, pp. 61-79, 1988.

Within this approach the drawing is incrementally specified in three phases (see Fig. 7): The first phase, *planarization*, determines the topology of the drawing. The second phase, *orthogonalization*, computes an orthogonal shape for the drawing. The third phase, *compaction*, produces the final drawing. This approach allows homogeneous treatment of a wide range of diagrammatic representations, aesthetics and constraints.

Another approach to the construction of orthogonal grid drawings, based on the results of [79] and on visibility representations (Section 5.4), is presented in:

71. H. de Fraysseix and P. Rosenstiehl, "Structures Combinatoires pour des Traces Automatiques de Reseaux", Proc. 3rd European Conf. on CAD/CAM and Computer Graphics, Paris, pp. 332-337, Hermes, 1984.

An algorithm for constructing polyline grid drawings that allows the user to choose between a hierarchical drawing method and the orthogonal grid drawing technique of [70] is presented in.

72. H. Trickey, "Drag: A Graph Drawing System," Proc. Int. Conf. on Electronic Publishing, pp. 171-182, Cambridge University Press, 1988.

Orthogonal grid drawings of graphs whose vertices have preassigned locations in the plane are investigated in:

73. Y. Kajitani and H. Takahashi, "Rectilinear Drawing of a Graph on a Plane with the Minimum Number of Segments", Manuscript (presented at the 2nd Int. Catania Combinatorial Conf.), 1989.

5 Planar Graphs

A graph is *planar* if it admits a planar drawing. Planar graphs play an important role in graph theory [1, 2] and graph algorithms; see [6, 10], and:

74. R.E. Tarjan, "Algorithm Design," Communications ACM, vol. 30, no. 3, pp. 205-212, 1987.

Clearly, planar drawings are aesthetically desirable. Furthermore, as discussed in the previous section, algorithms for drawing nonplanar graphs often begin by *planarizing* the graph (see section 4.2), and then by applying a planar graph drawing algorithm.

5.1 Planarity Testing and Planar Representations

A *planar representation* is a data structure representing the combinatorial adjacencies between the faces of a planar drawing. Most planar graph drawing methods proceed as follows:

Step 1. Test planarity.

Step 2. (if the graph is planar) Construct a planar representation.

Step 3. Use the planar representation to draw the graph according to some graphic standard.

In this subsection we consider the first two steps.

Finding a linear time algorithm to test the planarity of a graph was an interesting challenge for early algorithms research. The first algorithm to succeed used a *path addition approach* and was presented in:

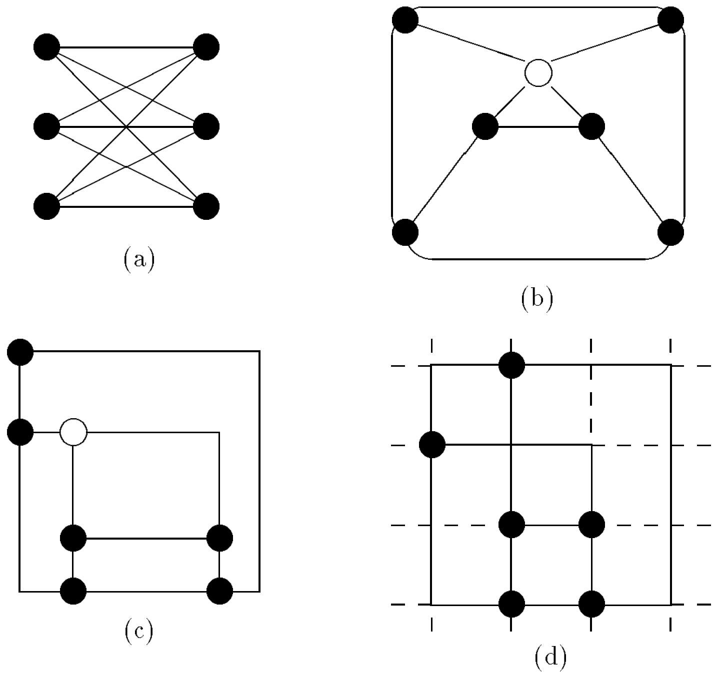


Figure 7: A general strategy for orthogonal grid drawings. (a) Given graph. (b) Planarization. (c) Orthogonalization. (d) Compaction.

75. J. Hopcroft and R.E. Tarjan, “Efficient Planarity Testing,” J. ACM, vol. 21, no. 4, pp. 549-568, 1974.

The *vertex addition approach* was developed to give a linear time algorithm in the following papers:

76. A. Lempel, S. Even, and I. Cederbaum, “An Algorithm for Planarity Testing of Graphs,” in Theory of Graphs, Int. Symposium (Rome, 1966), pp. 215-232, Gordon and Breach, New York, 1967.

77. S. Even and R.E. Tarjan, “Computing an st-Numbering,” Theoretical Computer Science, vol. 2, pp. 339-344, 1976.

78. K. Booth and G. Lueker, “Testing for the Consecutive Ones Property, Interval Graphs, and Graph Planarity Using PQ-Tree Algorithms,” J. of Computer and System Sciences, vol. 13, pp. 335-379, 1976.

Another approach is presented in:

79. H. de Fraysseix and P. Rosenstiehl, “A Depth-First-Search Characterization of Planarity”, Annals of Discrete Mathematics, vol. 13, pp. 75-80, 1982.

Minor errors of [75] are corrected in:

80. Narsingh Deo, “Note on Hopcroft and Tarjan’s Planarity Algorithm,” J. ACM, vol. 23, no. 1, pp. 74-75, 1976.

The aforementioned planarity testing algorithms can be modified to construct planar representations. The following paper extends the algorithm of [78] in this way.

81. N. Chiba, T. Nishizeki, S. Abe, and T. Ozawa, “A Linear Algorithm for Embedding Planar Graphs Using PQ-Trees”, J. of Computer and System Sciences, vol. 30, no. 1, pp. 54-76, 1985.

In the remainder of this Section we consider drawing algorithms that construct a planar drawing from a given planar representation.

5.2 Straight-Line Drawings

A classical result independently established by Wagner, Fary and Stein shows that every planar graph admits a planar straight-line drawing.

82. K. Wagner, "Bemerkungen zum Vierfarbenproblem," Jber. Deutsch. Math.-Verein, vol. 46, pp. 26-32, 1936.

83. I. Fary, "On Straight Lines Representation of Planar Graphs," Acta Sci. Math. Szeged, vol. 11, pp. 229-233, 1948.

84. S.K. Stein, "Convex Maps," Proc. Amer. Math. Soc., vol. 2, pp. 464-466, 1951.

This result also follows from Steinitz's theorem on convex polytopes in three dimensions.

85. E. Steinitz and H. Rademacher, Vorlesung über die Theorie der Polyeder, Springer, Berlin, 1934.

Convex drawings of planar graphs, that is, planar straight-line drawings where every face is drawn as a convex polygon (see Fig. 8) were first studied by Tutte.

86. W.T. Tutte, "Convex Representations of Graphs," Proc. London Math Soc., vol. 10, pp. 304-320, 1960.

87. W.T. Tutte, "How to Draw a Graph," Proc. London Math Soc., vol. 3, no. 13, pp. 743-768, 1963.

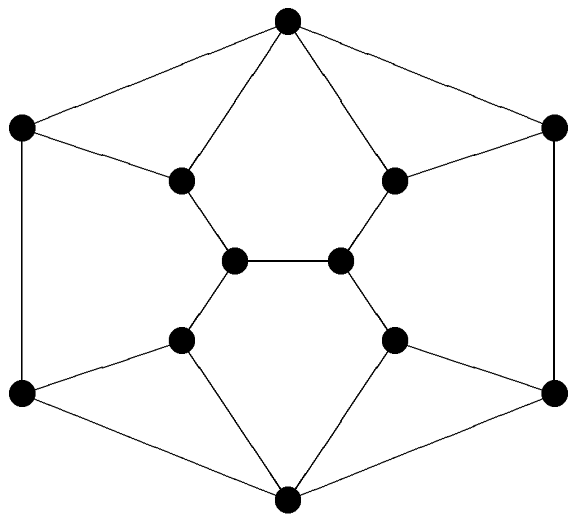


Figure 8: Convex drawing.

Tutte shows that a convex drawing of a *3-connected graph* (see [1]) can be obtained by solving a system of linear equations. Thomassen characterizes the class of graphs that admit a convex drawing.

88. C. Thomassen, "Planarity and Duality of Finite and Infinite Planar Graphs", J. Combinatorial Theory, Series B, vol. 29, pp. 244-271, 1980.

Chiba et al. show that Thomassen's result can be implemented as an algorithm for producing a convex drawing in linear time.

89. N. Chiba, T. Yamanouchi, and T. Nishizeki, "Linear Algorithms for Convex Drawings of Planar Graphs", in Progress in Graph Theory, ed. J.A. Bondy and U.S.R. Murty, pp. 153-173, Academic Press, 1984.

90. N. Chiba, K. Onoguchi, and T. Nishizeki, "Drawing Planar Graphs Nicely", Acta Informatica, vol. 22, pp. 187-201, 1985.

Becker et al. investigate the problem of minimizing the total edge length (according to several metrics, not including the Euclidean metric) in a planar straight-line drawing where the external face is a prescribed convex polygon. They show that the optimal drawing is unique and convex and can be obtained by standard numerical techniques.

91. B. Becker and G. Hotz, "On The Optimal Layout of Planar Graphs with Fixed Boundary", SIAM J. Computing, vol. 16, no. 5, pp. 946-972, 1987.

92. B. Becker and H.G. Osthof, "Layout with Wires of Balanced Length", Information and Computation, vol. 73, pp. 45-58, 1987.

Eades and Wormald show that the problem of constructing a planar straight-line drawing with prescribed edge lengths (according to the Euclidean metric) is NP-hard.

93. P. Eades and N. Wormald, "Fixed Edge Length Graph Drawing is NP- hard," Discrete Applied Mathematics, Vol. 28, pp. 111-134, 1990.

An elegant algorithm for constructing planar straight-line drawings has been given by Read. The algorithm uses linear times but quadratic storage.

94. R. Read, "New Methods for Drawing a Planar Graph Given the Cyclic Order of the Edges at Each Vertex", Congressus Numerantium, vol. 56, pp. 31-44, 1987.

Manning and Atallah give an algorithm for displaying symmetries in planar straight-line drawings of planar graphs.

95. J. Manning and M.J. Atallah, "Fast Detection and Display of Symmetry in Outerplanar Graphs", Technical Report CSD-TR- 606, Dept. of Computer Sciences, Purdue Univ., West Lafayette, IN, 1986.

96. M.J. Atallah and J. Manning, "Fast Detection and Display of Symmetry in Embedded Planar Graphs", Manuscript, Purdue Univ., 1988.

Schnyder and de Fraysseix et al. independently show that every planar graph admits a planar straight-line grid drawing with area $O(n^2)$.

97. H. de Fraysseix, J. Pach, and R. Pollack, "Small Sets Supporting Fary Embeddings of Planar Graphs", Proc. 20th ACM Symp. on Theory of Computing, pp. 426-433, 1988.

98. H. de Fraysseix, J. Pach, and R. Pollack, "How to Draw a Planar Graph on a Grid," Combinatorica, vol. 10, pp. 41-51, 1990.

99. W. Schnyder, "Embedding Planar Graphs on the Grid," Proc. ACM- SIAM Symp. on Discrete Algorithms, pp. 138-148, 1990.

Chrobak shows that the constructive proof of [97] can be modified to yield an $O(n)$-time drawing algorithm.

100. M. Chrobak and T.H. Payne, "A Linear Time Algorithm for Drawing a Planar Graph on a Grid", Technical Report UCR-CS-90-2, Dept. of Math. and Comput. Sci., Univ. California, Riverside, 1990.

The performance of the algorithms in [89], [100], [94], and [87] are compared in the following paper. These algorithms have been implemented and tested on randomly generated maximal planar graphs. The standard deviations in angle size, edge length, and face area are used to compare the quality of the planar straight-line drawings produced.

101. S. Jones, P. Eades, A. Moran, N. Ward, G. Delott, and R. Tamassia, "A Note on Planar Graph Drawing Algorithms," Technical Report 216, Dept. of Computer Science, Univ. of Queensland, 1991.

Kant presents an algorithm for constructing planar convex straight-line grid drawings with area $O(n^2)$. His technique has several other graph drawing applications.

102. G. Kant, "Drawing Planar Graphs Using the *lmc*-Ordering," Proc. IEEE Symp. on Foundations of Computer Science, pp. 101-110, 1992.

Several of the algorithms that produce planar straight-line drawings operate primarily on triangulations. Thus for this and other applications, algorithms for triangulating planar graphs are required. Such algorithms are presented in:

103. G. Kant and H.L. Bodlaender, "Triangulating Planar Graphs while Minimizing the Maximum Degree,"in Algorithm Theory (SWAT '92), (Proc. 3rd Scandinavian Workshop on Algorithm Theory, Helsinki, July 1992), ed. O. Nurmi E. Ukkonen, pp. 258-271, Lecture Notes in Computer Science, vol. 621, Springer-Verlag, 1992.

5.3 Orthogonal Grid Drawings

Investigations of planar orthogonal grid drawings were first motivated by problems in circuit layout. Within this graphic standard, minimizing the number of bends and the area is important for both diagram readability and VLSI applications (see Fig. 9).

Any planar graph of degree at most 4 admits a planar orthogonal grid drawing with area $O(n^2)$. Further, there are graphs which need quadratic area. These results are presented in:

104. Y. Shiloach, "Arrangements of Planar Graphs on the Planar Lattice", Ph.D. Thesis, Weizmann Institute of Science, Rehovot, Israel, 1976.

105. L. Valiant, "Universality Considerations in VLSI Circuits," IEEE Transactions on Computers, vol. C-30, no. 2, pp. 135-140, 1981.

Tamassia uses network flow techniques to give an $O(n^2 \log n)$-time algorithm for minimizing bends.

106. R. Tamassia, "On Embedding a Graph in the Grid with the Minimum Number of Bends", SIAM J. Computing, vol. 16, no. 3, pp. 421-444, 1987.

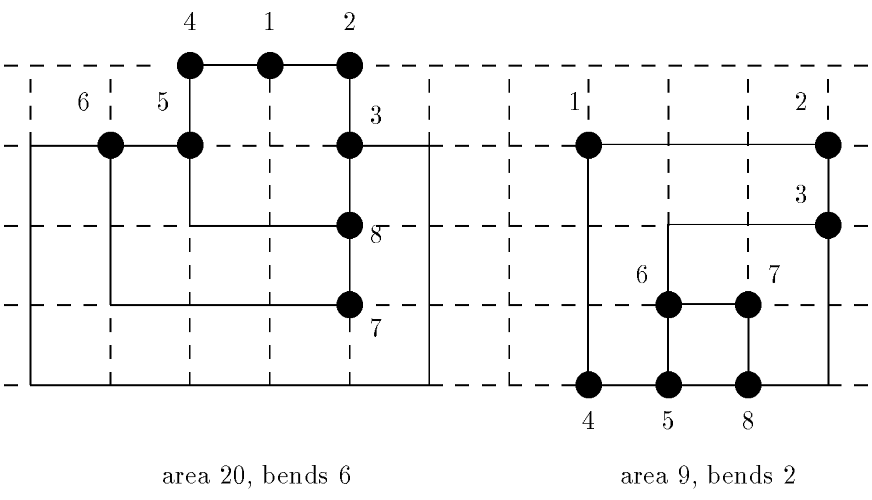


Figure 9: Examples of planar orthogonal grid drawings.

Storer gives three heuristics for constructing drawings with $O(n)$ bends.

107. J.A. Storer, "On Minimal Node-Cost Planar Embeddings," Networks, vol. 14, pp. 181-212, 1984.

Tamassia and Tollis present another heuristic for bend minimization which has the same performance bounds as the ones by Storer and runs in $O(n)$ time.

108. R. Tamassia and I.G. Tollis, "Efficient Embedding of Planar Graphs in Linear Time", Proc. IEEE Int. Symp. on Circuits and Systems, pp. 495-498, Philadelphia, 1987.

109. R. Tamassia and I.G. Tollis, "Planar Grid Embedding in Linear Time", IEEE Trans. on Circuits and Systems, vol. CAS-36, no. 9, pp. 1230-1234, 1989.

Lower bounds for planar orthogonal drawings of graphs, and parallel algorithms for achieving the same performance bounds as the ones by Storer and Tamassia and Tollis are described in

110. R. Tamassia, I.G. Tollis, and J.S. Vitter, "Lower Bounds for Planar Orthogonal Drawings of Graphs," *Information Processing Letters*, vol. 39, pp. 35-40 (1991).

111. R. Tamassia, I.G. Tollis, and J.S. Vitter, "Lower Bounds and Parallel Algorithms for Planar Orthogonal Grid Drawings," Proc. IEEE Symposium on Parallel and Distributed Processing, pp. 386-393 (1991).

NP-completeness results related to the minimization of area and total edge length in planar orthogonal grid drawings have been presented in [32, 33, 107, 34, 35] and:

112. D. Dolev, F.T. Leighton, and H. Trickey, "Planar Embedding of Planar Graphs", in Advances in Computing Research, vol. 2, ed. F.P. Preparata, pp. 147-161, JAI Press Inc., Greenwich, CT, 1984.

This paper also gives a heuristic for area minimization.

Orthogonal drawing algorithms are briefly surveyed in:

113. R. Tamassia, "Planar Orthogonal Drawings of Graphs", Proc. IEEE Int. Symp. on Circuits and Systems, 1990.

5.4 Visibility Representations

A *visibility representation* for a planar graph G consists of representing the vertices of G by horizontal segments, and the edges of G by vertical segments, so that the edge-segment associated with each edge (u, v) intersects exactly the vertex-segments associated with u and v, and no other vertex-segment (see Fig. 10).

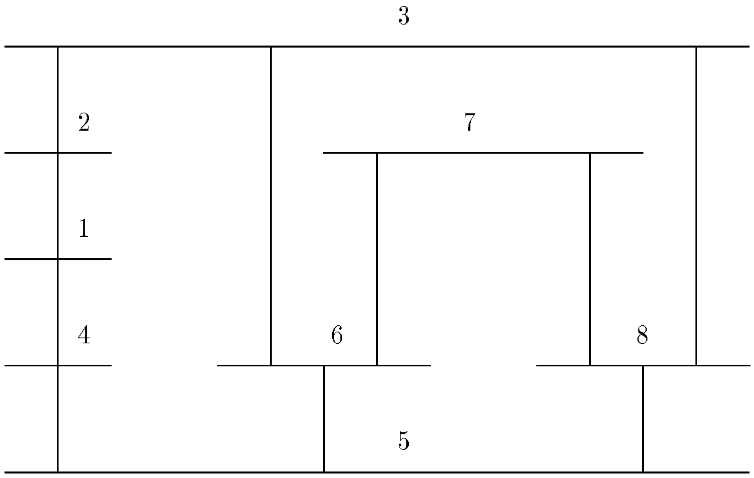


Figure 10: Visibility representation for the graph of Fig. 9.

The study of this graphic standard was originally motivated by VLSI layout and compaction problems because it gives regular and modular drawings.

114. M. Schlag, F. Luccio, P. Maestrini, D.T. Lee, and C.K. Wong, "A Visibility Problem in VLSI Layout Compaction", in Advances in Computing Research, vol. 2, ed. F.P. Preparata, pp. 259- 282, JAI Press Inc., Greenwich, CT, 1985.

Theoretical results characterizing visibility representations and variations of it appear in:

115. P. Duchet, Y. Hamidoune, M. Las Vergnas, and H. Meyniel, "Representing a Planar Graph by Vertical Lines Joining Different Levels", Discrete Mathematics, vol. 46, pp. 319- 321, 1983.

116. C. Thomassen, "Plane Representations of Graphs," in Progress in Graph Theory, ed. J.A. Bondy and U.S.R. Murty, pp. 43-69, Academic Press, 1984.

117. S.K. Wismath, "Characterizing Bar Line-of-Sight Graphs," Proc. ACM Symp. on Computational Geometry, pp. 147-152, 1985.

118. R. Tamassia and I.G. Tollis, "Representations of Graphs on a Cylinder," SIAM J. on Discrete Mathematics, vol. 4, no. 1, pp. 139-149, 1991.

119. F. Luccio, S. Mazzone, and C. Wong, "A Note on Visibility Graphs", Discrete Mathematics, vol. 63, pp. 105-110, 1987.

120. S.K. Wismath, "Weighted Visibility Graphs of Bars and Related Flow Problems", Algorithms and Data Structures (Proc. WADS'89), pp. 325-334, Lecture Notes in Computer Science, vol. 382, Springer-Verlag, 1989.

Algorithms that construct visibility representations in linear time are given in:

121. R.H.J.M. Otten and J.G. van Wijk, "Graph Representations in Interactive Layout Design", Proc. IEEE Int. Symp. on Circuits and Systems, pp. 914-918, New York, 1978.

122. P. Rosenstiehl and R.E. Tarjan, "Rectilinear Planar Layouts and Bipolar Orientations of Planar Graphs," Discrete & Computational Geometry, vol. 1, no. 4, pp. 343-353, 1986.

123. G. Kant, "A More Compact Visibility Representation," Proc. Int. Workshop on Graph-Theoretic Concepts in Computer Science (WG '93), Lecture Notes in Computer Science, 1993.

A complete combinatorial characterization of three classes of visibility representations and linear time drawing algorithms are presented in:

124. R. Tamassia and I.G. Tollis, "A Unified Approach to Visibility Representations of Planar Graphs", Discrete & Computational Geometry, vol. 1, no. 4, pp. 321-341, 1986.

An algorithm to construct *constrained* visibility representations (that is, representations where the edges of given paths are aligned) is presented in:

125. G. Di Battista, R. Tamassia, and I.G. Tollis, "Constrained Visibility Representations of Graphs", Information Processing Letters, vol. 41, pp. 1-7, 1992.

A bipolar orientation of an undirected graph consists of orienting the edges so that the resulting directed graph is acyclic and has exactly one source (vertex without incoming edges) and exactly one sink (vertex without outgoing edges). The creation of a bipolar orientation is often the first step for the generation of a visibility representation. The properties of bipolar orientations are systematically explored in terms of circuits, cocircuits, rank activities, Tutte polynomial, poset dimension, angle bipartition and max flow-min cut theorem in:

126. H. de Fraysseix, P.O. de Mendez, and P. Rosenstiehl, "Partial Orders and Orientation," Manuscript, 1992.

Efficient algorithms are described to list, generate or extend bipolar orientations for general graphs or plane ones, with or without constraints. The importance of the paper goes beyond visibility representations; in fact bipolar orientations are exploited in several drawing algorithms.

5.5 Other Graphic Standards

Algorithms for constructing planar polyline grid drawings are described in [104] and:

127. A.K. Hope, "A Planar Graph Drawing Program," Software Practice and Experience, vol. 1, pp. 83-91, 1971.

128. D. Woods, "Drawing Planar Graphs," Ph.D. dissertation (Technical Report STAN-CS-82-943), Computer Science Dept., Stanford Univ., 1982.

Another standard is proposed by Ozawa: vertices are placed on a horizontal line and edges are drawn as half-circles or smooth connections of half-circles.

129. T. Ozawa, "Planarity Testing for IC Layout with Constraints for Pin Order and Congestion Between Pins", IEEE Conf. Record of the 14th Asilomar Conf. on Circuits, Systems Computers, pp. 188-192, 1980.

Kant investigates representations of planar cubic graphs in the hexagonal grid, presenting a linear time algorithm:

130. G. Kant, "Hexagonal Grid Drawings," Technical Report RUU-CS-92-06, Department of Computer Science, Utrecht Univ., The Netherlands, 1992.

Representations of planar graphs by means of subdivisions of the plane into polygons (usually rectangles) have been motivated by problems in architectural design. Each vertex is represented by a polygon, and for each edge (u, v) the polygons associated with vertices u and v are geometrically adjacent. Essentially, this amounts to representing the graph by its dual. In most cases, the polygons are required to be rectangles; linear time algorithms for finding such dual representations are presented in:

131. J. Bhasker and S. Sahni, "A Linear Algorithm to Find a Rectangular Dual of a Planar Triangulated Graph", Algorithmica, vol. 3, no. 2, pp. 247-278, 1988.

132. X. He, "On Finding the Rectangular Duals of Planar Triangulated Graphs", SIAM J. Computing (to appear). Technical Report 90-24, Department of Computer Science, University of Buffalo 1990.

133. G. Kant and X. He, "Two Algorithms for Finding Rectangular Duals of Planar Graphs," Proc. Int. Workshop on Graph-Theoretic Concepts in Computer Science (WG '93), Lecture Notes in Computer Science, 1993.

Background to the architectural motivation can be found in:

134. J.P. Steadman, Architectural Morphology, Pion, London, 1983.

In a *tessellation representation*, each constituent (vertex, edge, and face) of an embedded planar graph is represented by a rectangle with horizontal and vertical sides, and incidencies between constituents correspond to geometric adjacencies between rectangles (see Fig. 11). These representations are investigated in:

135. R. Tamassia and I.G. Tollis, "Tessellation Representations of Planar Graphs", Proc. 27th Annual Allerton Conf., pp. 48-57, 1989.

An algorithm that maps vertices to grid points to facilitate the construction of a planar drawing is described in:

136. R. Jayakumar, K. Thulasiraman, and M.N.S Swamy, "Planar Embedding: Linear-Time Algorithms for Vertex Placement and Edge Ordering", IEEE Trans. on Circuits and Systems, vol. 35, no. 3, pp. 334-344, 1988.

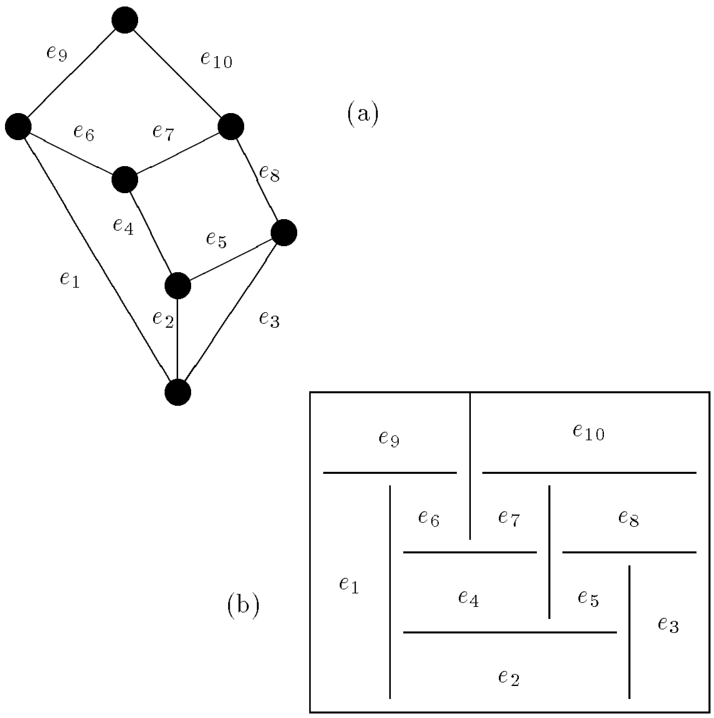


Figure 11: (a) A planar graph G. (b) Tessellation representation for G.

6 Directed Graphs

6.1 Acyclic Digraphs

Acyclic digraphs are widely used to display hierarchical structures. Examples include PERT diagrams, ISA hierarchies, and various dependency graphs. It is customary to represent these graphs so that the edges all flow in the same direction, e.g., from top to bottom, or from left to right (see Fig. 12). Namely, we say that a drawing of a digraph is *upward* if each arc is a curve monotonically increasing in the y-direction.

An important class of acyclic digraphs are covering digraphs of partially ordered sets. These digraphs are commonly represented by upward straight-line drawings, called *order diagrams*, *Hasse diagrams*, or simply *diagrams*.

A drawing algorithm for order diagrams is described in:

137. H. Jurgensen and J. Loewer, "Drawing Hasse Diagrams of Partially Ordered Sets", Technical Report TI 5/79, Institut für Theoretische Informatik, Technische Hochschule Darmstadt, Federal Republic of Germany, 1979.

Several issues in drawing order diagrams, such as the minimization of the number of slopes used for the arcs, are investigated in:

138. I. Rival and R. Wille, "Lattices Freely Generated by Partially Ordered Sets: Which can be Drawn?", J. für Reine und Angew. Math., vol. 310, pp. 56-80, 1979.

139. J. Czyzowicz, A. Pelc, and I. Rival, "Drawing Orders with Few Slopes", Technical Report TR-87-12, Dept. of Computer Science, Univ. of Ottawa, 1987.

140. J. Czyzowicz, A. Pelc, I. Rival, and J. Urrutia, "Crooked Diagrams with Few Slopes", Technical Report TR-87-26, Dept. of Computer Science, Univ. of Ottawa, 1987.

141. A. Pelc and I. Rival, "Orders with Level Diagrams," Technical Report TR-87-11, Dept. of Computer Science, Univ. of Ottawa, March 1987.

142. J. Czyzowicz, "Lattice Diagrams with Few Slopes," Technical Report #3, Département D'Informatique, Univ. of Quebec at Hull, 1987.

143. J. Czyzowicz, "Planar Lattices and the Slope Problem," Technical Report #4, Département D'Informatique, Univ. of Quebec at Hull, 1987.

144. J. Czyzowicz, A. Pelc, and I. Rival, "Planar Ordered Sets of Width Two", Technical Report TR-87-31, Dept. of Computer Science, Univ. of Ottawa, 1987.

Surveys on drawing techniques for order diagrams appear in:

145. I. Rival, "The Diagram," in Graphs and Orders, ed. I. Rival, pp. 103-133, Reidel Publishing, 1985.

146. I. Rival, "Graphical Data Structures for Ordered Sets," in Algorithms and Order, ed. I. Rival, pp. 3-31, Kluwer Academic Publishers, 1989.

6.1.1 Upward Planarity

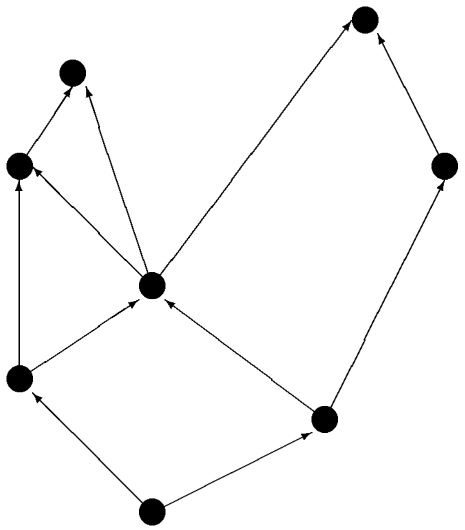


Figure 12: Upward drawing of an acyclic digraph.

The notion of planarity of undirected graphs has a corresponding notion of "upward planarity" for directed graphs. A drawing of a directed graph so that no pair of arcs cross and every arc is monotonically increasing in the y direction is an *upward drawing*. A graph is *upward planar* if it has an upward drawing. An upward planar drawing is in Fig. 12. Note that an upward planar graph must be acyclic, and its underlying undirected graph must be planar; however, there are planar acyclic digraphs which are not upward planar: see Fig. 13.

Various combinatorial characterizations of planar straight-line upward drawings are presented in:

147. D. Kelly and I. Rival, "Planar Lattices," Canadian J. Mathematics, vol. 27, no. 3, pp. 636-665, 1975.

148. C. Platt, "Planar Lattices and Planar Graphs," J. Combinatorial Theory, Series B, vol. 21, pp. 30-39, 1976.

149. D. Kelly, "Fundamentals of Planar Ordered Sets," Discrete Mathematics, vol. 63, pp. 197-216, 1987.

150. G. Di Battista and R. Tamassia, "Algorithms for Plane Representations of Acyclic Digraphs", Theoretical Computer Science, vol. 61, pp. 175-198, 1988.

151. C. Thomassen, "Planar Acyclic Oriented Graphs," Order, vol. 5, no. 4, pp. 349-361, 1989.

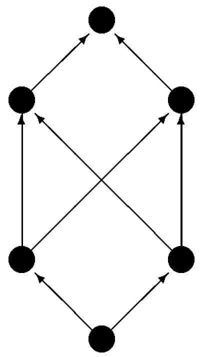


Figure 13: Planar acyclic digraph which is not upward planar.

Algorithms for upward drawings have three steps corresponding to the three phases for drawing general graphs as described in Subsection 4.3. However, the basic problem of algorithmically testing whether an acyclic digraph has an upward drawing is currently unsolved (see Section 9). For special classes of graphs, polynomial time algorithms have been found. These appear in:

152. G. Di Battista, W.P. Liu, and I. Rival, "Bipartite Graphs, Upward Drawings, and Planarity," Information Processing Letters, vol. 36, pp. 317-322, 1990.

153. M.D. Hutton and A. Lubiw, "Upward Planar Drawing of Single Source Acyclic Digraphs," Proc. 2nd ACM-SIAM Symp. on Discrete Algorithms, pp. 203-211, 1991.

154. P. Bertolazzi and G. Di Battista, "On Upward Drawing Testing of Triconnected Digraphs," Proc. 7th ACM Symp. on Computational Geometry, pp. 272-280, 1991.

155. P. Bertolazzi, G. Di Battista, G. Liotta, and C. Mannino "Upward Drawings of Triconnected Digraphs," Algorithmica (to appear).

156. P. Bertolazzi, G. Di Battista, C. Mannino, and R. Tamassia, "Optimal Upward Planarity Testing of Single-Source Digraphs," Proc. European Symp. on Algorithms, 1993.

If the topological structure (that is, a planar representation) of an upward planar digraph is known, then an upward drawing may be efficiently obtained; algorithms are given in [150]. For a survey see:

157. R. Tamassia, "Drawing Algorithms for Planar st-Graphs," *Australasian Journal of Combinatorics*, vol. 2, pp. 217-235 (1990).

In contrast to undirected graphs, upward planar straight line grid drawings may require exponential area. These results, as well as a discussion of symmetry display, may be found in:

158. G. Di Battista, R. Tamassia, and I.G. Tollis, "Area Requirement and Symmetry Display in Drawing Graphs", Proc. ACM Symp. on Computational Geometry, pp. 51-60, 1989.

159. G. Di Battista, R. Tamassia, and I.G. Tollis, "Area Requirement and Symmetry Display of Planar Upward Drawings," Discrete & Computational Geometry, vol. 7, pp. 381-401, 1992.

Lower bounds on area requirements and algorithms for constructing planar upward drawings of series-parallel digraphs are given in:

160. P. Bertolazzi, R.F. Cohen, G. Di Battista, R. Tamassia, and I.G. Tollis, "How to Draw a Series-Parallel Digraph," in Algorithm Theory (SWAT '92), (Proc. 3rd Scandinavian Workshop on Algorithm Theory, Helsinki, July 1992), ed. O. Nurmi E. Ukkonen, pp. 272-283, Lecture Notes in Computer Science, vol. 621, Springer-Verlag, 1992.

6.1.2 Hierarchical Drawings

A *hierarchical* drawing of an acyclic digraph is an upward polyline drawing where the vertices and bends are constrained to lie on a set of equally spaced horizontal lines, called *layers* (see Fig. 14). In some applications the assignment of vertices to layers is given, e.g., by the semantics of the graph. Such graphs are called *layered digraphs*, or *hierarchies*.

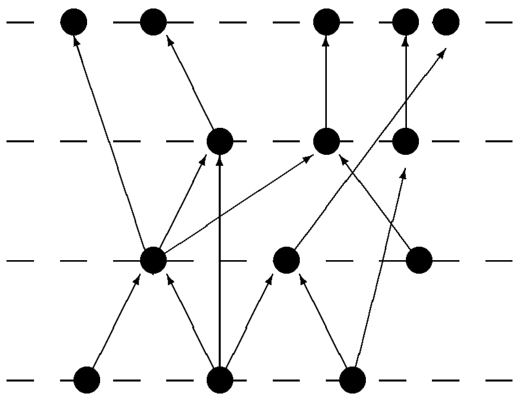


Figure 14: Hierarchical drawing.

Most of the rooted tree drawing algorithms of Section 3 may be used to draw trees as hierarchies. Sugiyama et al. present a comprehensive approach (see Fig. 15):

Step 1 Assign vertices to the layers so that vertices are distributed uniformly.

Step 2 Select a permutation of the vertices in each layer to reduce crossings.

Step 3 Adjust the position of the vertices in each layer to reduce the number of bends.

161. K. Sugiyama, S. Tagawa, and M. Toda, "Methods for Visual Understanding of Hierarchical Systems", IEEE Trans. on Systems, Man, and Cybernetics, vol. SMC-11, no. 2, pp. 109- 125, 1981.

162. K. Sugiyama and M. Toda, "Structuring Information for Understanding Complex Systems: A Basis for Decision Making", FUJITSU Scientific and Technical Journal, vol. 21, no. 2, pp. 144-164, 1985.

163. K. Sugiyama, "A Cognitive Approach for Graph Drawing," Cybernetics and Systems: An International Journal, vol. 18, pp. 447-488, 1987.

Variations and extensions of this approach are presented in:

164. M.J. Carpano, "Automatic Display of Hierarchized Graphs for Computer Aided Decision Analysis", IEEE Trans. on Systems, Man, and Cybernetics, vol. SMC-10, no. 11, pp. 705-715, 1980.

165. M.J. Carpano and M. Delarche, "Apport des Techniques Graphiques Interactives á l'Analyse Structurale de Systèmes. II- Exemples de Realization et d'Application", RAIRO Sept. Anal. Cont., June 1980.

166. M. May and P. Mennecke, "Layout of Schematic Drawings," Syst. Anal. Model. Simul., vol. 1, 4, pp. 307-338, 1984.

167. L.A. Rowe, M. Davis, E. Messinger, C. Meyer, C. Spirakis, and A. Tuan, "A Browser for Directed Graphs," Software Practice and Experience, vol. 17, no. 1, pp. 61-76, 1987.

168. E.B. Messinger, "Automatic Layout of Large Directed Graphs," Technical Report 88-07-08, Univ. of Washington, Dept. of Computer Science, 1988.

169. E.R. Gansner, S.C. North, and K.P. Vo, "DAG – A Program that Draws Directed Graphs", Software Practice and Experience, vol. 18, no. 11, pp. 1047-1062, 1988.

170. D. Jablonowski and V.A. Guarna, "GMB: A Tool for Manipulating and Animating Graph Data Structures", Software Practice and Experience, vol. 19, no. 3, pp. 283-301, 1989.

171. E.R. Gansner, E. Koutsofios, S.C. North, and K.P. Vo, "A Technique for Drawing Directed Graphs," IEEE Trans. on Software Engineering (to appear).

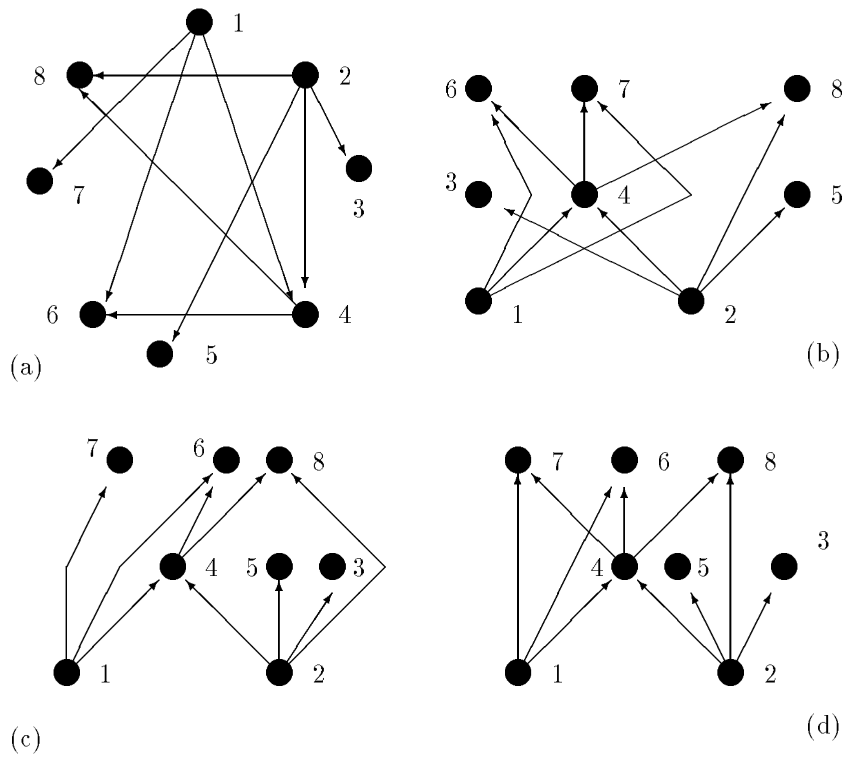


Figure 15: A general strategy for hierarchical drawings. (a) Given digraph. (b) Assignment of vertices to layers. (c) Crossing reduction. (d) Placement of vertices and bends.

Analyses of algorithms used at each of the three steps are presented in:

172. P. Eades and L. Xuemin, "How to Draw a Directed Graph," Proc. IEEE Workshop on Visual Languages (VL'89), pp. 13-17, 1989.

173. P. Eades and K. Sugiyama, "How to Draw a Directed Graph," Journal of Information Processing, vol. 14, no. 4, pp. 424–437, 1990.

174. P. Eades, "Complexity Issues in Drawing Directed Graphs", Proceedings of the International Workshop on Discrete Algorithms and Complexity, pp. 9–15, Fukuoka, Japan, Novermber 1989.

175. X. Lin, "Analysis of Algorithms for Drawing Graphs", PhD thesis, Department of Computer Science, University of Queensland, 1992.

Heuristics for the assignment of vertices to layers in Step 1 of the above technique are described in:

176. K. Sugiyama, "A Readability Requirement on Drawing Digraphs: Level Assignment and Edge Removal for Reducing the Total Length of Lines", Research Report No. 45, Int. Inst. for Advanced Study of Social Information Science, FUJITSU, Numazu, Japan, March 1984.

177. P. Eades and X. Lin,, "Notes on the Layer Assignment Problem for Drawing Directed Graphs", ACSC 14, Proceedings of the 14th Australian Computer Science Conference, University of New South Wales, pp. 26-10, 1991.

A divide-and-conquer algorithm for hierarchical drawings is proposed in:

178. 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. on Systems, Man, and Cybernetics, vol. SMC-21, no. 1, pp. 1-12, 1991.

A recursive algorithm for hierarchical drawings that partitions the original graph into subgraphs whose elements are closely related is presented in:

179. D.J. Gschwind and T.P. Murtagh, "A Recursive Algorithm for Drawing Hierarchical Directed Graphs", Technical Report CS-89-02, Dept. of Computer Science, Williams College, 1989.

A linear time algorithm for constructing hierarchical drawings is presented in:

180. G. Robins, "The ISI Grapher: A Portable Tool for Displaying Graphs Pictorially", Technical Report ISI/RS-87-196, Information Sciences Inst., Univ. of Southern California, 1987. (Also in Proc. Symboliikka '87, Helsinki, Finland, August 1987)

Orthogonal hierarchical drawings are investigated in:

181. J.E. Savage, "Heuristics for Level Graph Embeddings," Proc. Workshop on Graphtheoretic Concepts in Computer Science, pp. 307-318, Trauner Verlag, 1983.

Crossing reduction is a fundamental aesthetic for hierarchical drawings. An efficient algorithm to construct a planar hierarchical drawing of a layered digraph is given in:

182. G. Di Battista and E. Nardelli, "An Algorithm for Testing Planarity of Hierarchical Graphs", in Graph-Theoretic Concepts in Computer Science, (Proc. Int. Workshop WG '86, Bernierd, June 1986), ed. G. Tinhofer and G. Schmidt, pp. 277-289, Lecture Notes in Computer Science, vol. 246, Springer-Verlag, 1987.

183. G. Di Battista and E. Nardelli, "Hierarchies and Planarity Theory", IEEE Trans. on Systems, Man, and Cybernetics, 1988.

An algorithm which uses a technique adapted from [87] for hierarchical drawings is presented in:

184. P. Eades, X. Lin and R. Tamassia, An Algorithm for Drawing a Hierarchical Graph, manuscript 1992.

Minimizing crossings for layered digraphs is NP-hard even if there are only two layers [38], and even if the positions of the vertices on the first layer are fixed. This problem, as well as a discussion of heuristics, is presented in:

185. P. Eades, B. McKay, and N. Wormald, "On an Edge Crossing Problem", Proc. 9th Australian Computer Science Conf., pp. 327-334, Australian National University, 1986.

186. P. Eades and N. Wormald, "Edge Crossings in Drawings of Bipartite Graphs" Technical Report 108, Department of Computer Science, University od Queensland. (to appear in Algorithmica).

A heuristic for crossing minimization in layered digraphs with 2 layers that guarantees to produce at most three times the minimum number of crossings is given in:

187. P. Eades and N. Wormald, "The Median Heuristic for Drawing 2- Layered Networks", Technical Report 69, Dept. of Computer Science, Univ. of Queensland, 1986.

Other heuristics for crossing minimization in layered digraphs are studied in the following papers:

188. J. Warfield, "Crossing Theory and Hierarchy Mapping," IEEE Trans. on Systems, Man, and Cybernetics, vol. SMC-7, no. 7, pp. 502-523, 1977.

189. P. Eades and D. Kelly, "Heuristics for Reducing Crossings in 2- Layered Networks", Ars Combinatoria, vol. 21.A, pp. 89-98, 1986.

190. E. Mäkinen, "Experiments on Drawing 2-Level Hierarchical Graphs," International Journal of Computer Mathematics 36 (1990), 175-181.

191. E. Mäkinen, "A Note on the Median Heuristic for Drawing Bipartite Graphs", Fundamenta Informaticae XII (1989), 563-570.

192. T. Catarci, "The Assignment Heuristic for Crossing Reduction in Bipartite Graphs", Proc. 26th Annual Allerton Conf., 1988.

193. M. May and K. Szkatula, "On the Bipartite Crossing Number", Control and Cybernetics, vol. 17, no. 1, pp. 85-98, 1988.

194. E. Mäkinen, "Remarks on the Assignment Heuristic for Drawing Bipartite Graphs," Technical Report A-1990-7, Dept. of Computer Science, Univ. of Tampere, Finland, 1990.

195. E. Mäkinen, "On Drawing Regular Bipartite Graphs," International Journal of Computer Mathematics 43 (1992), 39-43.

A heuristic algorithm that simplifies dense hierarchical graphs by replacing complete bipartite subgraphs with a single "concentrator node" is presented in the following paper.

196. F.J. Newbery, "Edge Concentration: A Method for Clustering Directed Graphs," Proc. 2nd Int. Workshop on Software Configuration Management, pp. 76-85, 1989.

The transformation greatly enhances visual simplicity and may reduce the number of crossings; see [175] for a discussion of the complexity issues involved.

The display of symmetries in hierarchical drawings is investigated in [184] and:

197. K. Sugiyama, "Achieving Uniqueness Requirement in Drawing Digraphs: Optimum Code Algorithm and Hierarchic Isomorphism", Research Report No. 58, Int. Inst. for Advanced Study of Social Information Science, FUJITSU, Numazu, Japan, July 1985.

Radial drawings of layered digraphs are investigated in [164] and:

198. M.G. Reggiani and F.E. Marchetti, "A Proposed Method for Representing Hierarchies", IEEE Trans. on Systems, Man, and Cybernetics, vol. 18, no. 1, pp. 2-8, 1988.

6.2 General Digraph Drawing Algorithms

When the representation of *flow* in digraphs with cycles is an important aesthetic, one would like to maximize the number of arcs that are directed upward. This problem is equivalent to reversing a minimum number of arcs to make the digraph acyclic, and is commonly known as the *feedback arc set problem.* The problem is NP-complete in general, but it is polynomially solvable for several classes of graphs including planar digraphs:

199. A. Frank, "How to Make a Digraph Strongly Connected", *Combinatorica*, Vol. 1, No. 2, 1981.

Heuristics for the feedback arc set problem are discussed in [161, 164, 167, 168, 169, 170, 176, 178, 179, 172, 175], and

200. B. Berger and P. Shor, "Approximation Algorithms for the Maximum Acyclic Subgraph Problem", Proceedings of the first annual SIAM-ACM Symposium on Discrete Algorithms (1990), pp. 236-243.

After the transformation into an acyclic digraph, the techniques surveyed in the previous subsection can be applied.

If the representation of flow is not important, algorithms for drawing undirected graphs can be applied by ignoring the directions of the arcs.

6.3 Application-Specific Algorithms

There are several drawing algorithms developed for specific applications, especially circuit schematics and information systems diagrams. In this framework, the semantics of the diagram and the conventions of the application area may put constraints on the drawing. For example, vertices representing interfaces in a Data Flow diagram are conventionally placed on the external boundary.

The problem of dealing with constraints on the drawing imposed by the user is specifically investigated in:

201. R. Tamassia, "New Layout Techniques for Entity-Relationship Diagrams", Proc. 4th Int. Conf. on Entity-Relationship Approach, pp. 304-311, Chicago, 1985.

The automatic generation of schematic diagrams for digital systems is studied in:

202. A. Arya, A. Kumar, V. Swaminathan, and A. Misra, "Automatic Generation of Digital System Schematic Diagrams", Proc. 22nd Design Automation Conf., pp. 388-395, 1985.
203. F. Aoudja, M. Laborie, and A. Saint-Paul, "CASE: Automatic Generation of Electrical Diagrams", Computer-Aided Design, vol. 18, no. 7, pp. 356-360, 1986.
204. M.A. Majewski, F.N. Krull, T.E. Fuhrman, and P.J. Ainslie, "Autodraft: Automatic Synthesis of Circuit Schematics," Proc. IEEE Int. Conf. on Computer-Aided Design, pp. 435-438, 1986.

Following the classical layout approach for integrated circuits, these algorithms perform the placement of modules and the routing of connections in two separate steps.

A drawing algorithm for PERT diagrams is presented in:

205. G. Di Battista, E. Pietrosanti, R. Tamassia, and I.G. Tollis, "Automatic Layout of PERT Diagrams with XPERT," Proc. IEEE Workshop on Visual Languages (VL'89), pp. 171-176, 1989.

An algorithm for drawing flowcharts appears in:

206. D.E. Knuth, "Computer Drawn Flocharts," Communications of the ACM, vol. 6, 1963.

The following papers describe divide-and-conquer algorithms targeted toward Entity Relationship diagrams:

207. D. Reiner, M. Brodie, G. Brown, M. Chilenskas, M. Friedell, D. Kramlich, J. Lehman, and A. Rosenthal, "A Database Design and Evaluation Workbench: Preliminary Report", Proc. Int. Conf. on Systems Development and Requirements Specification, Gothenburg, Sweden, 1984.
208. D. Reiner, G. Brown, M. Friedell, J. Lehman, R. McKee, P. Rheingans, and A. Rosenthal, "A Database Designer's Workbench", in Entity-Relationship Approach (Proc. 5th Int. Conf. on Entity-Relationship Approach, Dijon, France, 1987), ed. S. Spaccapietra, pp. 347-360, North-Holland, 1987.
209. D. Reiner and G. Brown, "Heuristic Layout for DDEW ER+ Diagrams," Manuscript, Computer Corporation of America, 1985.

An algorithm for Entity Relationship diagrams based on visibility representations is in:

210. J. Nummenmaa and J. Tuomi, "Constructing Layouts for ER-Diagrams from Visibility Representations," Proc. 9th Int. Conf. on Entity-Relationship Approach, pp. 303-317, Lausanne, Switzerland, 1990.

Based on the general strategy of [69, 70], drawing algorithms for three diagrammatic representations widely used in information systems design are given in:

211. C. Batini, M. Talamo, and R. Tamassia, "Computer Aided Layout of Entity-Relationship Diagrams", The Journal of Systems and Software, vol. 4, pp. 163-173, 1984.
212. P. Di Felice and R. Tamassia, "Automatic Layout of Flow Diagrams: Preliminary Analysis", Proc. ISMM, pp. 263-267, Madrid, 1985.
213. C. Batini, E. Nardelli, and R. Tamassia, "A Layout Algorithm for Data-Flow Diagrams", IEEE Transactions on Software Engineering, vol. SE-12, no. 4, pp. 538-546, 1986.

Layout methods for class hierarchies used in object-oriented systems are developed in:

214. H. Koike, "An Application of Three Dimensional Visualization to Object-Oriented Programming," *Advanced Visual Interfaces* (Proceedings of AVI'92), World Scientific Series in Computer Science vol. 36, 180-192.

7 Graph Drawing Systems

There are many computer systems available for editing graphs and graph-like diagrams. Some of these contain a naive automatic drawing facility:

215. M. Dao, M. Habib, J. Richard, and D. Tallot, "CABRI, an Interactive System for Graph Manipulation", in Graph- Theoretic Concepts in Computer Science, (Proc. Int. Workshop WG '86, Bernierd, June 1986), ed. G. Tinhofer and G. Schmidt, pp. 58-67, Lecture Notes in Computer Science, vol. 246, Springer-Verlag, 1987.

216. J.M Fourneau, I. Fournier, A. Germa, and D. Sotteau, "Unicorn: A Computer-Aided Scratch Book for Graph Theory", Technical Report 381, L.R.I., UA410 CNRS, Univ. Paris Sud, 1987.

217. F. Aschim and B.M. Mostue, "IFIP WG 8.1 Case Solved Using SYSDOC and SYSTEMATOR", in Information Systems Design Methodologies: a Comparative Review (Proc. of the IFIP WG 8.1 Working Conf. on Comparative Review of Information Systems Design Methodologies), ed. T. Olle et al., pp. 15- 40, North-Holland, Noordwijkerhout, the Netherlands, 1982.

218. M. Nagl and H. Zischler, "A Dialog System for the Graphical Representation of Graphs", Applied Computer Science, vol. 13 (Proc. Workshop WG '78 on Graphtheoretic Concepts in Computer Science), pp. 325-339, 1979.

219. M. Sarkar and Marc H. Brown, "Graphical Fisheye Views of Graphs," Technical Report CS-91-61, Dept. of Computer Science, Brown Univ., 1991.

Other systems use significant layout algorithms. They are described in [258, 127, 128, 137, 161, 162, 164, 167, 169, 170, 43, 45, 52, 70, 203, 204, 205, 207, 208] and:

220. M. Himsolt, "GraphEd: An Interactive Graph Editor," Proc. STACS 89, Lecture Notes in Computer Science, vol. 349, pp. 532- 533, Springer-Verlag, 1989.

221. J. Hynd and P. Eades, "The Typed Graph Editing System - TYGES," Proc. 3rd Australasian Conf. on Computer Graphics (Ausgraph 85), pp. 15-19, Brisbane, Australia, 1985.

222. P. Eades, I. Fogg and D. Kelly, "SPREMB: a System for Developing Graph Algorithms". Congressus Numerantium Vol. 66 (1988), 123-140.

223. C. Batini, E. Nardelli, M. Talamo, and R. Tamassia, "GINCOD: a Graphical Tool for Conceptual Design of Data Base Applications", in Computer Aided Data Base Design, ed. A. Albano, V. De Antonellis, and A. Di Leva, pp. 33-51, North Holland, 1985.

224. G. Di Battista and R. Tamassia, "An Integrated Graphic System for Designing and Accessing Statistical Data Bases", Proc. 7th Symp. on Computational Statistics (COMPSTAT 1986), pp. 231- 236, Physica-Verlag, 1986.

225. C. Batini, P. Brunetti, G. Di Battista, P. Naggar, E. Nardelli, G. Richelli, and R. Tamassia, "An Automatic Layout Facility and its Applications", Proc. Int. Workshop on Software Engineering Environment, pp. 139-157, China Academic Publishers, Beijing, China, 1986. (invited paper)

226. R. Read, "Methods for Computer Display and Manipulation of Graphs, and the Corresponding Algorithms", Research Report CORR 86-12, Faculty of Mathematics, Univ. of Waterloo, July 1986.

227. K. Nakamura, H. Fujimoto, T. Suzuki, Y. Tarui, and Y. Kiyokane, "Visual Programming Environment in Communications Software," Proc. 5th IEEE Global Telecom Conf., pp. 435-439, 1986.

228. W.F. Tichy and F.J. Newbery, "Knowledge-based Editors for Directed Graphs", in ESEC'87 (Proc. 1st European Software Engineering Conf.), ed. H.K. Nichols and D. Simpson, pp. 109-117, Springer-Verlag, 1987.

229. F. Newbery, "An Interface Description Language for Graph Editors", Proc. IEEE Workshop on Visual Languages, 1988.

230. F. Newberry Paulish and W.F. Tichy: "EDGE: An Extendible Graph Editor," Software-Practice and Experience, vol. 20, no. S1, pp. S1/63-S1/88, 1990

231. S.P. Reiss and J.N. Pato, "Displaying Program and Data Structures", Proc. 20th Hawaii Int. Conf. on System Sciences, 1987.

232. S.P. Reiss, "Integration Mechanisms in the FIELD Environment," Technical Report CS-88-18, Dept. of Computer Science, Brown Univ., 1988.

233. S.P. Reiss, S. Meyers, and C. Duby, "Using GELO to Visualize Software Systems", Manuscript, Dept. Computer Science, Brown Univ., 1989.

234. B. Birgisson and G. Shannon, "GraphView: A Workstation-Based Environment for Viewing Graphs and Animating Graph Algorithms", Technical Report 295, Dept. Computer Science, Indiana Univ., 1989.

235. V. Jansen, A. Potthoff, W. Thomas, and U. Wermuth, "A Short Guide to the AMORE System," Technical Report 90-2, Fachgruppe Informatik, RWTH Aachen, 1990.

236. O. Baudon, "Cabri-graphes, un CAhier de BRouillon Interactif pour la theorie des graphes", These de Doctorat de l'Universite Joseph Fourier, Grenoble, France, 1990.

237. J. Bordier and J.M. Laborde, "An Interactive Tool for Graph Theory", Proc. 7th Annual Apple European University Consortium Conference, Paris, France, ed. Apple Computer Europe, Inc, pp. 51-53, 1991.

238. G. Di Battista, G. Liotta, M. Strani and F. Vargiu, "Diagram Server," *Advanced Visual Interfaces* (Proceedings of AVI 92), World Scientific Series in Computer Science, vol. 36, pp. 415-417, 1992.

239. G. Di Battista, A. Giammarco, G. Santucci, and R. Tamassia, "The Architecture of Diagram Server," Proc. IEEE Workshop on Visual Languages (VL'90), pp. 60-65, 1990.

240. M. Beccaria, P. Bertolazzi, G. Di Battista, and G. Liotta, "A Tailorable and Extensible Automatic Layout Facility," Proc. IEEE Workshop on Visual Languages (VL'91), pp. 68-73, 1991.

241. M. Bousset and P. Rosenstiehl, "Twist," Tech. Rep. CAMS P.073, 1991.

242. J.C. Smart and V. Vemuri, "A-Vu: A Visualization Tool for Complex Software Systems" Proc. Symposium on Assessment of Quality Software Development Tools, IEEE Computer Society Press, New Orleans LA, May 27-29, 1992.

243. E.R. Gansner, E. Koutsofios, S.C. North, and K.P. Vo, "Graph Visualization in Software Analysis" Proc. Symposium on Assessment of Quality Software Development Tools, IEEE Computer Society Press, New Orleans LA, May 27-29, 1992.

A graph drawing system for dataflow diagrams based on placement-and-routing techniques is described in:

244. L.B. Protsko, P.G. Sorenson, J.P. Tremblay, and D.A. Schaefer, "Towards the Automatic Generation of Software Diagrams",IEEE Trans. on Software Engineering, vol. SE-17, no. 1, pp. 10-21, 1991.

A tool for displaying large graphs supporting multiple views, nonuniform scaling, and user-defined constraints on the layout is described in:

245. T.R. Henry and S.E. Hudson, "Viewing Large Graphs," Technical Report 90-13, Dept. of Computer Science, Univ. of Arizona, 1990.

A tool that uses clustering techniques is described in:

246. P. Brown, T. Gargiulo, "An Object Oriented Layout for Directed Graphs," Proc. Symposium on Assessment of Quality Software Development Tools, IEEE Computer Society Press, New Orleans LA, May 27-29, 1992.

8 Special Topics

8.1 Parallel Algorithms

A parallel algorithm for planarity testing that runs in $O(\log n)$ time on a CRCW PRAM with $(n \log\log n)/\log n$ processors is presented in:

247. V. Ramachandran and J.H. Reif, "An Optimal Parallel Algorithm for Graph Planarity," Proc. IEEE Symp. on Foundations of Computer Science, pp. 282-293, 1989.

248. V. Ramachandran and J. Reif, "Planarity Testing in Parallel," Technical Report TR-90-15, Dept. of Computer Science, The University of Texas at Austin, 1990.

Previous results on parallel planarity testing are:

249. J. Ja'Ja' and J. Simon, "Parallel Algorithms in Graph Theory: Planarity Testing," SIAM J. Comput., vol. 11, no. 2, pp. 314-328, 1982.

250. G.L. Miller and J.H. Reif, "Parallel Tree Contraction and its Applications," Proc. 26th IEEE Symp. on Foundations of Computer Science, pp. 478-489, 1985.

251. P.N. Klein and J.H. Reif, "An Efficient Parallel Algorithm for Planarity", J. Computer and System Sciences, vol. 37, no. 2, pp. 190-246, 1988.

Parallel graph drawing algorithms for planar graphs are presented in [111] and in the following papers:

252. R. Tamassia and J.S. Vitter, "Optimal Parallel Algorithms for Transitive Closure and Point Location in Planar Structures," Proc. ACM Symp. on Parallel Algorithms and Architectures, pp. 399-408 (1989).

253. R. Tamassia and J.S. Vitter, "Parallel Transitive Closure and Point Location in Planar Structures," SIAM J. Computing, vol. 20, no. 4, pp. 708-725, 1991.

254. M. Furer, B. Raghavachari, X. He, and M.-Y. Kao, "$O(n \log\log n)$-Work Parallel Algorithms for Straight-Line Grid Embeddings of Planar Graphs," Proc. ACM Symp. on Parallel Algorithms and Architectures, 1992.

8.2 Dynamic Algorithms

A reference model for dynamic drawing algorithms is given in:

255. R.F. Cohen, G. Di Battista, R. Tamassia, I.G. Tollis, and P. Bertolazzi, "A Framework for Dynamic Graph Drawing", Proc. ACM Symp. on Computational Geometry, pp. 261-270, 1992.

The paper contains also several results on dynamic problems within the proposed model.

An on-line planarity testing algorithm supporting insertions of vertices and edges with logarithmic query/update time is presented in:

256. G. Di Battista and R. Tamassia, "Incremental Planarity Testing," Proc. 30th IEEE Symp. on Foundations of Computer Science, pp. 436-441, 1989.

The best result on fully dynamic planary testing (where both insertions and deletions are allowed) is an algorithm with $O(\sqrt{n})$ amortized query and update time, given in

257. D. Eppstein, Z. Galil, G.F. Italiano, and T.H. Spencer, "Separator Based Sparsification for Dynamic Planar Graph Algorithms," Proc. ACM Symp. on Theory of Computing, pp. 208-217, 1993.

An algorithm for drawing trees in a dynamic environment is presented in:

258. S. Moen, "Drawing Dynamic Trees," IEEE Software, vol. 7, pp. 21-8, 1990.

The incremental construction of an orthogonal drawing is investigated in

259. K. Miriyala, S.W. Hornick, and R. Tamassia, "An Incremental Approach to Aesthetic Graph Layout," Proc. Int. Workshop on Computer-Aided Software Engineering (CASE '93) (1993).

An important consideration in dynamic graph layout is *preserving the mental map*: when a change is made to a graph by the user, the re-application of a layout algorithm may destroy the user's "mental map". Models and techniques for preserving the mental map are discussed in

260. P. Eades, W. Lai, K. Misue and K. Sugiyama, "Preserving the Mental Map of a Diagram", Proceedings of Compugraphics 91, Portugal 1991, pp. 24-33.

261. K. Lyons, "Cluster Busting in Anchored Graph Drawing", Proceedings of the 1992 CAS Conference, IBM Centre for Advanced Studies, Toronto 1992, pp. 7-16.

262. K. Bohringer and F. Newbery Paulisch, "Using Constraints to Achieve Stability in Automatic Graph Layout Algorithms", Proceedings of ACM CHI 90, (1990) pp 43-51.

8.3 Three Dimensions

Three-dimensional drawings of graphs are investigated in [170, 214] and

263. G.G. Robertson, J.D. Mackinlay, and S.K. Card, "Cone Trees: Animated 3D Visualizations of Hierarchical Information," Proc. CHI, pp. 189-193, 1991.

264. S.P. Reiss, "A Framework for Abstract 3D Visualization," Proc. IEEE Symp. on Visual Languages (VL '93), 1993.

8.4 Hypergraphs

Two notions of planarity for hypergraphs and NP-completeness results are given in:

265. D.S. Johnson and H.O. Pollak, "Hypergraph Planarity and the Complexity of Drawing Venn Diagrams", J. Graph Theory, vol. 10, no. 3, pp. 309-325, 1987.

A new formalism for representing graphs and hypergraphs, called *higraph*, is introduced in:

266. D. Harel, "On Visual Formalisms," Communications of the ACM, vol. 31, no. 5, pp. 514-530, 1988.

An algorithm for drawing hypergraphs is presented in:

267. E. Mäkinen, "How to Draw a Hypergraph," International Journal of Computer Mathematics 34 (1990), 177-185.

8.5 Declarative Methods

Several recent techniques for graph drawing emphasize the expression of the aesthetics rather than the algorithmic complexity of achieving the aesthetics. These techniques, called *declarative techniques*, are perhaps outside the scope of this bibliography, because the algorithms used to achieve the aesthetics are naive and require excessive computational resources. An example is the use of genetic algorithms:

268. C. Kosak and J. Marks, "A Parallel Genetic Algorithm for Network-Diagram Layout", Proc. 4th Int. Conf. on Genetic Algorithms (ICGA91), 1991.

269. C. Kosak, J. Marks, and S. Shieber, "New Approaches to Automating Network-Diagram Layout", IEEE Trans. on Systems, Man, and Cybernetics. To appear. 1993.

Other examples include the simulated annealing methods of [49], and the constraint resolution methods of [30].

The formal specification of constraints in the drawing of a graph is studied in [244] and in

270. J.D. Mackinlay, "Automating the Design of Graphical Presentations of Relational Information", ACM Transactions on Graphics Vol. 5, No. 2, 1986.

271. J. Marks, "A Formal Specification Scheme for Network Diagrams that Facilitates Automated Design," J. Visual Languages and Computing, vol. 2, pp. 395-414, 1991.

272. E. Dengler, M. Friedell, and J. Marks, "Constraint-Driven Diagram Layout," Proc. IEEE Symp. on Visual Languages (VL '93), 1993.

273. S. Deal, "The Specification and Recognition of Optimal Layout Configurations for Graph Structures," Ph.D. dissertation, Dept. Computer Science, Univ. College London, 1989.

8.6 Aesthetics

A discussion of graph drawing aesthetics appears in:

274. C. Esposito, "Graph Graphics: Theory and Practice," Comput. Math. Applic., vol. 15, no. 4, pp. 247-253, 1988.

An experimental study of aesthetics used in Entity Relationship diagrams is reported in:

275. C. Batini, L. Furlani, and E. Nardelli, "What is a Good Diagram? A Pragmatic Approach", Proc. 4th Int. Conf. on the Entity Relationship Approach, Chicago, 1985.

An analogous study in the field of data structure diagrams is in:

276. C. Ding and P. Mateti, "A Framework for the Automated Drawing of Data Structure Diagrams", IEEE Trans. on Software Engineering, vol. SE-16, no. 5, pp. 543-557, 1990.

8.7 Compound Graphs

In *compound digraphs*, edges represent both adjacency and inclusion relations. Compound graphs and similar structures (such as *higraphs* [266]) are powerful modeling tools for relational information.

Layout algorithms are given in [30] and

277. K. Sugiyama and K. Misue, "Visualization of Structural Information: Automatic Drawing of Compound Digraphs," IEEE Transactions on Systems, Man and Cybernetics, vol. 21, no. 4, pp. 876-892, 1991.

8.8 Angles

An interesting aesthetic is to ensure that the angles between the segments that represent edges are not too small. Studies of this aesthetic applied to planar straight line drawings are in:

278. G. Vijayan, "Geometry of Planar Graphs with Angles," Proc. ACM Symp. on Computational Geometry, pp. 116-124, 1986.

279. S. Malitz and A. Papakostas, "On the Angular Resolution of Planar Graphs," Proc. 24th ACM Symp. on the Theory of Computing, pp. 527-538, 1992.

280. G. Di Battista and L. Vismara, "Angles of Planar Triangular Graphs," Proc. ACM Symp. on Theory of Computing, pp. 431-437, 1993.

It is shown in [279] that it is always possible to construct a straight line planar drawing whose smallest angle is $O(\alpha^d)$, where $0 < \alpha < 1$, and d is the maximum degree of a vertex of the graph. Further results are given for outerplanar graphs.

A similar problem, but for nonplanar graphs, is considered in:

281. M. Formann, T. Hagerup, J. Haralambides, M. Kaufmann, F.T. Leighton, A. Simvonis, E. Welzl, and G. Woeginger, "Drawing Graphs in the Plane with High Resolution", Proc. IEEE Symp. on Foundations of Computer Science, pp. 86-95, 1990.

It is shown that it is always possible to construct a drawing whose smallest angle between the edges incident at a vertex is $O(1/d^2)$, where d is the maximum degree of a vertex of the graph. Other results are given for particular classes of graphs.

9 Open Problems

Despite the abundance of literature on automatic graph drawing, many theoretical and practical problems are still open. A few of the most promising directions for further research are listed below.

Performance Bounds for Planarization Although crossing minimization is a fundamental issue, non-trivial performance bounds have not been found for any heuristic. A guaranteed heuristic would be very important both for aesthetic graph drawing and VLSI layout.

Planarity of Acyclic Digraphs There is a combinatorial characterization of the acyclic digraphs that admit a planar upward drawing [149, 150]. However, no polynomial time algorithm for testing "upward planarity" in general acyclic digraphs is known.

Simple Planarity Testing The known planarity algorithms that achieve linear time complexity (Section 5.1) are all difficult to understand and implement. This is a serious limitation for their use in practical systems. A simple and efficient algorithm for testing the planarity of a graph and constructing planar representations would be a significant contribution.

General Strategy for Straight-Line Drawings General strategies have been successfully developed for hierarchical drawings (Section 6.1) and orthogonal grid drawings (Section 4.3). These techniques take several aesthetics into account. The simplicity of straight-line drawings is very appealing, and a general straight-line drawing technique would find immediate applications. The most versatile technique for planar straight-line drawings is the the one by Kant [102].

Dynamic Drawing Algorithms Several graph manipulation systems allow the user to interactively modify a graph by inserting and deleting vertices and edges. Data structures that allow for fast restructuring of the drawing would be very useful. Especially important is the *dynamic planarity testing* problem, where we want a data structure for planar graphs that supports in polylogarithmic time the following operations: (a) testing whether a new edge can be added while preserving planarity; (b) adding vertices and edges which preserve planarity; and (c) removing vertices and edges. When only insertions are allowed, this problem can be efficiently solved in $O(\log n)$ time per test or update, as shown in [256]. However, the best solution for the general problem (insertions and deletions) has $O(\sqrt{n})$ amortized query and update time [257].

10 Acknowledgements

The authors wish to thank the many graph drawers who have pointed out errors and omissions in the first three versions, and those who have helped with updates to create this fourth version.

11 Index of Authors