Algorithms for drawing graphs:
an annotated bibliography

Giuseppe Di Battista\textsuperscript{a}, Peter Eades\textsuperscript{b}, Roberto Tamassia\textsuperscript{c}* ,
Ioannis G. Tollis\textsuperscript{d}

\textsuperscript{a} Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza",
Via Salaria 113, Roma, 00198 Italy

\textsuperscript{b} Department of Computer Science, University of Newcastle, University Drive, Callaghan, NSW 2308, Australia

\textsuperscript{c} Department of Computer Science, Brown University, Providence, RI 02912–1910, USA

\textsuperscript{d} Department of Computer Science, The University of Texas at Dallas, Richardson, TX 75083–0688, USA

Abstract

Several data presentation problems involve drawing graphs so that they are easy to read and understand. Examples include circuit schematics and software engineering diagrams. 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.

1. Introduction

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

\*This document is available via anonymous ftp from wilma.cs.brown.edu (128.148.33.66), files /pub/papers/compgeo/gdbiblio.tex.Z and /pub/papers/compgeo/gdbiblio.ps.Z. Research supported in part by the Army Research Office, by the National Science Foundation, and by the C.N.R. (Italy).

\* Corresponding author. E-mail: rt@cs.brown.edu.

0925-7721/94/$07.00 © 1994 Elsevier Science B.V. All rights reserved
SSDI 0925-7721(94)00014-M
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.

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.

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 al-
Fig. 4. Two drawings of the cube graph.

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. The talks given at the first workshop on graph drawing are listed in Appendix A. An index of authors is provided in Appendix B.

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:


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

Fundamentals of data structures and algorithms are described in:


Algorithms for graph problems and applications are described in:


Algorithms for planar graphs are presented in:


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


Basic concepts of computer graphics and computational geometry are given in:


Two previous versions of this bibliography have appeared as:


Many abstracts of recent papers on graph drawing appear in:


The talks presented at Graph Drawing '93 are listed in Appendix A.
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.

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.


Implementation details of a variation of the algorithm by Reingold and Tilford [20] are discussed by Brueggemann-Klein and Wood; the paper presents a set of \TeX macros to implement the algorithm.


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.


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:


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


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

Free trees do not represent hierarchies and have no specific root. 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 graph-theoretic center of the tree. Folklore on radial and other simple drawings of free trees is summarized in:


Strategies for constructing radial drawings are described in:


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


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

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


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:


Many problems are NP-hard even for restricted classes of graphs, such as trees and planar graphs. Specific results are presented in [24, 27, 33–36, 120] and:


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:

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:


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 [89, 90, 94, 95] 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 [31] and:


48. T. Kamada, Symmetric Graph Drawing by a Spring Algorithm and its Applications to Radial Drawing, Manuscript, Department of Information Science, University of Tokyo, 1989.


A general model for spring algorithms is defined in [188]; 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 a measure of the closeness of vertices. A layout of minimal energy (an thus maximal beauty according to the energy function) is obtained by simulated annealing.
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:


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 [49] and [50] are described in:


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 [49] and [50] are described in:


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


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.


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. Significant examples of this strategy are presented in [73, 107]. 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, as shown in [277] and:


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


65. O. Goldschmidt and A. Takvorian, An Efficient Graph Planarization Two-Phase Heuristic, Technical Report ORP91-01, Department of Mechanical Engineering, University of Texas at Austin, 1991.


Another planarization technique is to find a drawing with the minimum number of crossings. Again, this problem is NP-hard [39]. Heuristics for crossing minimization are given in:


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:


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


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


4.3. Polyline drawings

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


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 [83] and on visibility representations (Section 5.4), is presented in:


An algorithm for constructing polyline grid drawings that allows the user to choose between a hierarchical drawing method and the orthogonal grid drawing
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:


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:


Minor errors of [78] are corrected in:


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


Another approach is presented in:


The aforementioned planarity testing algorithms can be modified to construct planar representations. The following paper extends the algorithm of [82] in this way.


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.


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


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.


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.

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


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.


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.


An elegant algorithm for constructing planar straight-line drawings has been given by Read. The algorithm uses $O(n)$ time but $O(n^2)$ storage.


Manning and Atallah give algorithms for and discuss complexity of displaying symmetries in planar straight-line drawings of planar graphs in [32] and:


101. J. Manning, Geometric Symmetry in Graphs, Ph.D. Thesis, Department of Computer Sciences, Purdue University, West Lafayette, IN, 1990.

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


Chrobak and Payne show that the constructive proof of [102] can be modified to yield an $O(n)$-time drawing algorithm.


The performance of the algorithms in [92, 105, 97, 90] 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.


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.


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:


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:


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


Di Battista, Liotta, and Vargiu give polynomial time algorithms for minimizing bends (considering all the possible embeddings) for series-parallel graphs and graphs with degree at most 3.


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


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.


The structure of orthogonal embeddings of graphs is investigated in:


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:


NP-completeness results related to the minimization of area and total edge length in planar orthogonal grid drawings have been presented in [33, 34, 114, 35, 36] and:


This paper also gives a heuristic for area minimization.

Orthogonal drawing algorithms are briefly surveyed in:


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).

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


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

Algorithms that construct visibility representations in linear time are given in the following papers and in [135].


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


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


Linear time algorithms for constructing visibility representations of trees with optimal area are presented in:


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:


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 [110] and:


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.


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


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:


Background to the architectural motivation can be found in:


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:


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

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:


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


Surveys on drawing techniques for order diagrams appear in:


6.1.1. Upward planarity

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:


Planarization-based 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:


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 [162]. For a survey see:


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:


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


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.

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 arcs are directed upward and 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.
Fig. 14. Hierarchical drawing.

Fig. 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.


Variations and extensions of this approach are presented in:


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


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

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

191. 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. Systems, Man, and Cybernetics,

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

192. 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

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

Report ISI/RS-87-196, Information Sciences Institute, University of Southern California,
Marina del Rey, CA, 1987. (Also in: Proc. Symbolikka '87, Helsinki, Finland, August
1987.)

Orthogonal hierarchical drawings are investigated in:

194. J.E. Savage, Heuristics for Level Graph Embeddings, in: Proc. Workshop on Graph-theo-

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:

Graphs, in: G. Tinhofer and G. Schmidt, eds., Graph-Theoretic Concepts in Computer
Science (Proc. Internat. Workshop WG '86, Bernierd, June 1986), Lecture Notes in


An algorithm which uses a technique adapted from [90] for hierarchical
drawings is presented in:

197. P. Eades, X. Lin and R. Tamassia, An Algorithm for Drawing a Hierarchical Graph,
in: J. Urrutia, ed., Proc. Second Canadian Conference on Computational Geometry,
(Ottawa, Ont.), pp. 142-146, 1990.

Minimizing crossings for layered digraphs is NP-hard even if there are only
two layers [39], and even if there is only one node in each layer:

198. S. Masuda, K. Nakajima, T. Kashiwabara, and T. Fujisawa, Crossing Minimization in
1990.

Further NP-completeness results, as well as analyses of an heuristics (one of
which gives at most three times the minimum number of crossings) are given in:

199. P. Eades, B. McKay, and N. Wormald, On an Edge Crossing Problem, in: Proc. 9th

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

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


203. E. Mäkinen, Experiments on Drawing 2-Level Hierarchical Graphs, Internat. J. Computer

204. E. Mäkinen, A Note on the Median Heuristic for Drawing Bipartite Graphs, Fundamenta

205. T. Catarci, The Assignment Heuristic for Crossing Reduction in Bipartite Graphs, in:

206. M. May and K. Szkatula, On the Bipartite Crossing Number, Control and Cybernetics,

207. E. Mäkinen, Remarks on the Assignment Heuristic for Drawing Bipartite Graphs, Technical
Report A-1990-7, Department of Computer Science, University of Tampere, Finland,
1990.


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


The transformation greatly enhances visual simplicity and may reduce the
number of crossings; see [188] for a discussion of the complexity issues
involved.

The display of symmetries in hierarchical drawings is investigated in [197]
and:

Algorithm and Hierarchic Isomorphism, Research Report no. 58, Internat. Institute for

Radial drawings of layered digraphs are investigated in [177] and:
Dominance drawings

A dominance drawing of an acyclic directed graph $G = (V, E)$ is a function $f : V \rightarrow \mathbb{R}^k$ such that $(f(u), f(v)) \in E$ if and only if $f(u) \neq f(v)$ and each coordinate of $f(v)$ is at least as large as the corresponding coordinate of $f(u)$. A dominance drawing in dimension $k$ can be viewed as an embedding of the graph in a $k$ dimensional partial order. Thus several mathematical results on partial orders can be used to derive algorithms for dominance drawings. Algorithms and complexity of creating such representations are given in [171] and:


Related results appear in [159, 161]. Algorithms for dominance drawings of series parallel graphs are in [172]. A linear time algorithm for finding a dominance drawing of a bipartite graph in two dimensions is given in:


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:


Heuristics for the feedback arc set problem are discussed in [174, 177, 180–183, 189, 191, 192, 185, 188], and:


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 software engineering 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. In this section we list a sample of papers covering such application specific techniques.

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


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


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:


An algorithm for drawing flowcharts appears in:


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

7. Graph drawing systems

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


Other systems use significant layout algorithms. They are described in [279, 139, 140, 149, 174, 175, 177, 180, 182, 183, 44, 46, 53, 73, 220–222, 224, 225] and:


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:


Previous results on parallel planarity testing are:


Parallel graph drawing algorithms for planar graphs are presented in [119] and in the following papers:


8.2. Dynamic algorithms

A reference model for dynamic drawing algorithms is given in:
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:


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


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


The incremental construction of an orthogonal drawing is investigated in:


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:


8.3. Three dimensions

Three-dimensional drawings of graphs are investigated in [183, 231] and


8.4. Hypergraphs

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


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


An algorithm for drawing hypergraphs is presented in:


8.5. Separator-based algorithms

Separator-based algorithms for area-efficient (nonplanar) orthogonal drawings of trees, planar graphs, and other computationally interesting networks (e.g., d-dimensional mesh, cube-connected cycles, and shuffle-exchange) are studied in [111] and:


Goodrich gives an optimal algorithm for the separator decomposition of planar graphs, which improves the time complexity of separator-based algorithm for planar graphs.


8.6. 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, often require very large computational resources, and are perhaps outside the scope of this bibliography. An example is the use of genetic algorithms:


Other examples include the simulated annealing methods of [50], and the constraint resolution methods of [31].

The formal specification of constraints in the drawing of a graph is studied in [263] and in:


An approach to drawing graphs based on graph grammars is presented by Brandenburg:


A visual approach to graph drawing is presented in:


8.7. Aesthetics

A discussion of graph drawing aesthetics appears in:


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


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


8.8. Compound graphs

In compound digraphs, edges represent both adjacency and inclusion relations. Compound graphs and similar structures (such as higraphs [287]) are powerful modeling tools for relational information.

Layout algorithms for compound digraphs are given in [31] and:

8.9. 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:


It is shown in [305] 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:


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 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, nontrivial performance bounds have not been found for any heuristic. A guaranteed heuristic would be very important both for aesthetic graph drawing and VLSI layout.

- **Upward Planarity Testing.** There is a combinatorial characterization of the acyclic digraphs that admit a planar upward drawing [161, 162]. 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 one by Kant [108]. Some further progress in this direction is reported in [257].

• **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 [277]. However, the best solution for the general problem (insertions and deletions) has \(O(\sqrt{n})\) amortized query and update time [278].

• **Complexity of Bend Minimization.** Several issues on the computational complexity of minimizing bends in planar orthogonal drawings are open. No general polynomial-time algorithm for this problem is known. If the embedding is fixed, bend minimization can be done in time \(O(n^2\log n)\) [112]. Particular classes of graphs are investigated in [113]. It would be interesting to improve on the sequential complexity and to develop a fast parallel algorithm for the fixed-embedding problem.

• **Area of Planar Upward Drawings of Trees.** The area requirement of upward planar drawings of trees has been studied in [25, 26], where tight bounds are given for polyline drawings (\(\Theta(n)\)) and orthogonal drawings (\(\Theta(n\log\log n)\)). The area requirement of straight-line drawings is not known instead. The best upper bound is \(O(n\log n)\), while only the trivial \(\Omega(n)\) lower bound is known.

• **Angular Resolution of Planar Straight-Line Drawings.** The angular resolution of a planar straight-line drawing is the minimum angle formed by two edges incident on the same vertex. It has been shown that a planar graph of degree \(d\) has a drawing with angular resolution \(\Omega(1/d)\) [305]. Only the trivial \(O(1/d)\) upper bound is known.

• **Size Bounds for Three-Dimensional Grid Drawings.** Graph drawing systems which exploit for three dimensions already exist but very little theory has
been developed. In particular, practically nothing is known about upper and lower bounds for the sides of the enclosing rectangular prism of a three dimensional grid drawing.

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.

Appendix A. Graph Drawing '93

The following papers have been presented at Graph Drawing '93 [16].

Session 0: Invited Lecture. Chair: Pierre Rosenstiehl
On the Four Colour Problem. C. Berge

Session 1: Geometric Graph Theory. Chair: Roberto Tamassia
New Developments in Geometric Graph Theory. J. Pach

Session 2: Trees. Chair: Giuseppe Di Battista
Characterizing Proximity Trees. P. Bose, W. Lenhart and G. Liotta
A Note on Free Drawings of Binary Trees on a Square. F.J. Brandenburg and P. Eades
Two Algorithms for Drawing Trees in Three Dimensions. B. Regan
Area Requirement of Visibility Representations of Trees. G. Kant, G. Liotta, R. Tamassia and I.G. Tollis

Session 3: Upward Drawings. Chair: Takao Nishizeki
Efficient Computation of Planar Straight-Line Upward Drawings. A. Garg and R. Tamassia
An Approach for Bend-Minimal Upward Drawing. U. Fößmeier and M. Kaufmann

Session 4: Invited Lecture. Chair: Hubert de Fraysseix
Representations of Planar Graphs. C. Thomassen

Session 5: Representations in the Plane I. Chair: Anna Lubiw
On Lattice Structures Induced by Orientations. P.O. de Mendez
Complexity of Intersection Classes of Graphs. J. Kratochvíl and Jiří Matoušek
On Triangle Contact Graphs. H. de Fraysseix, P.O. de Mendez and P. Rosenstiehl

Session 6: Representations in the Plane II. Chair: Ioannis G. Tollis
Characterization and Construction of the Rectangular Dual of a Graph. S. Pimont and M. Terreboën
Two Algorithms for Finding Rectangular Duals of Planar Graphs. G. Kant and X. He
A More Compact Visibility Representation. G. Kant

Session 7: Beyond the Plane I. Chair: János Pach
Circle Packing Representations in Polynomial Time. B. Mohar
Generalizing Kuratowski's Theorem. B. Mohar
Automorphisms and Genus on Generalised Maps. A. Bergey
Upward Drawing on Surfaces. I. Rival

Session 8: Beyond the Plane II. Chair: Ivan Rival
Tessellation and Visibility Representations of Maps on the Torus. B. Mohar and P. Rosenstiehl
A Simple Construction of High Representativity Triangulations. T. M. Przytycka and J.H. Przytycki

On Graph Drawings with Smallest Number of Faces. J. Chen, S.P. Kanchi and J.L. Gross

Session 9: Drawings and Flows. Chair: Michael Kaufmann

A Flow Model of Low Complexity for Twisting a Layout. M. Bousset
Convex and non-Convex Cost Functions of Orthogonal Representations. G. Di Battista, G. Liotta and F. Vargiu

Topology and Geometry of Planar Triangular Graphs. G. Di Battista and L. Vismara

Session 10: Complexity. Chair: Joseph Manning

Algorithms for Embedding Graphs Into a 3-page Book. M.S. Miyauchi
Dominance Drawings of Bipartite Graphs. H. ElGindy, M. Houle, B. Lenhart, M. Miller, D. Rappaport and S. Whitesides
Computing the Overlay of Regular Planar Subdivisions in Linear Time. U. Finke and K. Hinrichs

Generation of Random Planar Maps. A. Denise

Session 11: Symmetry. Chair: Peter Eades

Symmetric Drawings of Graphs. J. Manning
Recognizing Symmetric Graphs. T. Pisanski

Session 12: Declarative Approaches. Chair: Franz J. Brandenburg

Algorithmic and Declarative Approaches to Aesthetic Layout. P. Eades and T. Lin
A Visual Approach to Graph Drawing. I.F. Cruz, R. Tamassia and P. Van Hentenryck
Layout of Trees with Attribute Graph Grammars. G. Zinjmeister

The Display, Browsing and Filtering of Graph-Trees. S.P. Foubister and C. Runciman

Session 13: Graph Drawing Systems I. Chair: David Rappaport

A Layout Algorithm for Undirected Graphs. D. Tunkelang

Drawing Ranked Digraphs with Recursive Clusters. S.C. North

Session 14: Graph Drawing Systems II. Chair: Robert F. Cohen

Graph Drawing Algorithms for the Design and Analysis of Telecommunication Networks. I.G. Tollis and C. Xia

A View to Graph Drawing Algorithms through GraphEd. M. Himsolt
An Automated Graph Drawing System Using Graph Decomposition. C.L. McCreary, C.L. Combs, D.H. Gill and J.V. Warren

Session 15: Embedding and Planarization I. Chair: Bojan Mohar


Heuristics for Planarization by Vertex Splitting. P. Eades and X. Mendonça
Planar Graph Embedding with a Specified Set of Face-Independent Vertices. T. Ozawa

Session 16: Embedding and Planarization II. Chair: Herbert Fleischner

Implementation of the Planarity Testing Algorithm by Demoucron, Malgrange and Pertuiset. S.B. Johansen

A Unified Approach to Testing, Embedding and Drawing Planar Graphs. J.F. Small
A Simple Linear-Time Algorithm for Embedding Maximal Planar Graphs. H. Stamm-Wilbrandt
## Appendix B. Index of Authors

<table>
<thead>
<tr>
<th>Author</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abe, S.</td>
<td>84.</td>
</tr>
<tr>
<td>Ainslie, P.J.</td>
<td>221.</td>
</tr>
<tr>
<td>Andreae, T.</td>
<td>130.</td>
</tr>
<tr>
<td>Aoudja, F.</td>
<td>220.</td>
</tr>
<tr>
<td>Arya, A.</td>
<td>219.</td>
</tr>
<tr>
<td>Aschim, F.</td>
<td>234.</td>
</tr>
<tr>
<td>Batini, C.</td>
<td>72, 73, 228, 230, 240, 242, 301.</td>
</tr>
<tr>
<td>Baudon, O.</td>
<td>254.</td>
</tr>
<tr>
<td>Bailey, D.A.</td>
<td>44.</td>
</tr>
<tr>
<td>Beccaria, M.</td>
<td>258.</td>
</tr>
<tr>
<td>Becker, B.</td>
<td>94, 95.</td>
</tr>
<tr>
<td>Berger, B.</td>
<td>217.</td>
</tr>
<tr>
<td>Bernard, M.A.</td>
<td>30.</td>
</tr>
<tr>
<td>Bhasker, J.</td>
<td>143.</td>
</tr>
<tr>
<td>Bhatt, S.</td>
<td>33.</td>
</tr>
<tr>
<td>Birgisson, B.</td>
<td>252.</td>
</tr>
<tr>
<td>Bodlaender, H.L.</td>
<td>109.</td>
</tr>
<tr>
<td>Bohringer, K.</td>
<td>283.</td>
</tr>
<tr>
<td>Bondy, J.</td>
<td>1.</td>
</tr>
<tr>
<td>Booth, K.</td>
<td>82.</td>
</tr>
<tr>
<td>Bordier, J.</td>
<td>255.</td>
</tr>
<tr>
<td>Bousset, M.</td>
<td>259.</td>
</tr>
<tr>
<td>Brandenburg, F.J.</td>
<td>34, 298.</td>
</tr>
<tr>
<td>Brodie, M.</td>
<td>224.</td>
</tr>
<tr>
<td>Brown, G.</td>
<td>224, 225, 226.</td>
</tr>
<tr>
<td>Brown, M.</td>
<td>236.</td>
</tr>
<tr>
<td>Brown, P.</td>
<td>265.</td>
</tr>
<tr>
<td>Brueggegmann-Klein, A.</td>
<td>23.</td>
</tr>
<tr>
<td>Bruenetti, P.</td>
<td>242.</td>
</tr>
<tr>
<td>Cai, J.</td>
<td>55.</td>
</tr>
<tr>
<td>Card, S.K.</td>
<td>284.</td>
</tr>
<tr>
<td>Carpano, M.J.</td>
<td>177, 178.</td>
</tr>
<tr>
<td>Catarcii, T.</td>
<td>205.</td>
</tr>
<tr>
<td>Cederbaum, I.</td>
<td>80.</td>
</tr>
<tr>
<td>Chiba, N.</td>
<td>10, 57, 84, 92, 93.</td>
</tr>
<tr>
<td>Chilenskas, M.</td>
<td>224.</td>
</tr>
<tr>
<td>Chrobak, M.</td>
<td>105.</td>
</tr>
<tr>
<td>Cohen, R.F.</td>
<td>172, 173, 276.</td>
</tr>
<tr>
<td>Cormen, T.H.</td>
<td>4.</td>
</tr>
<tr>
<td>Cosmadakis, S.</td>
<td>33.</td>
</tr>
<tr>
<td>Crescenzi, P.</td>
<td>25.</td>
</tr>
<tr>
<td>Cruz, I.F.</td>
<td>299.</td>
</tr>
<tr>
<td>Cuny, J.E.</td>
<td>44.</td>
</tr>
<tr>
<td>Czyzowicz, J.</td>
<td>151, 152, 154, 155, 156.</td>
</tr>
<tr>
<td>Dao, M.</td>
<td>232.</td>
</tr>
<tr>
<td>Davidson, R.</td>
<td>50.</td>
</tr>
<tr>
<td>Davis, M.</td>
<td>180.</td>
</tr>
<tr>
<td>Deal, S.</td>
<td>297.</td>
</tr>
<tr>
<td>de Fraysseix, H.</td>
<td>16, 74, 83, 102, 103, 138.</td>
</tr>
<tr>
<td>Dehne, F.</td>
<td>275.</td>
</tr>
<tr>
<td>Delarche, M.</td>
<td>178.</td>
</tr>
<tr>
<td>Delou, G.</td>
<td>106.</td>
</tr>
<tr>
<td>de Mendez, P.O.</td>
<td>138.</td>
</tr>
<tr>
<td>Dengler, E.</td>
<td>296.</td>
</tr>
<tr>
<td>Deo, N.</td>
<td>5, 79.</td>
</tr>
<tr>
<td>Di Felice, P.</td>
<td>229.</td>
</tr>
<tr>
<td>Ding, C.</td>
<td>302.</td>
</tr>
<tr>
<td>Djidjev, H.</td>
<td>275.</td>
</tr>
<tr>
<td>Dolev, D.</td>
<td>120.</td>
</tr>
<tr>
<td>Duby, C.</td>
<td>251.</td>
</tr>
<tr>
<td>Duchet, P.</td>
<td>123.</td>
</tr>
<tr>
<td>Eggleton, R.B.</td>
<td>69.</td>
</tr>
<tr>
<td>Eppstein, D.</td>
<td>278.</td>
</tr>
<tr>
<td>Esposito, C.</td>
<td>300.</td>
</tr>
<tr>
<td>Even, S.</td>
<td>6, 80, 81, 213.</td>
</tr>
<tr>
<td>Fary, I.</td>
<td>86.</td>
</tr>
<tr>
<td>Ferrari, D.</td>
<td>67.</td>
</tr>
<tr>
<td>Fogg, I.</td>
<td>239.</td>
</tr>
<tr>
<td>Foley, J.D.</td>
<td>12.</td>
</tr>
<tr>
<td>Formann, M.</td>
<td>307.</td>
</tr>
<tr>
<td>Fouls, L.</td>
<td>63, 64.</td>
</tr>
<tr>
<td>Fourmeau, J.M.</td>
<td>233.</td>
</tr>
<tr>
<td>Fourrier, I.</td>
<td>233.</td>
</tr>
<tr>
<td>Frank, A.</td>
<td>216.</td>
</tr>
<tr>
<td>Friedell, M.</td>
<td>224, 225, 296.</td>
</tr>
<tr>
<td>Fruchterman, T.</td>
<td>49.</td>
</tr>
<tr>
<td>Fuhrman, T.E.</td>
<td>221.</td>
</tr>
<tr>
<td>Fujimoto, H.</td>
<td>244.</td>
</tr>
<tr>
<td>Fujisawa, T.</td>
<td>198.</td>
</tr>
<tr>
<td>Furter, M.</td>
<td>273, 274.</td>
</tr>
<tr>
<td>Furlani, L.</td>
<td>301.</td>
</tr>
<tr>
<td>Gansner, E.R.</td>
<td>182, 184, 261.</td>
</tr>
<tr>
<td>Garey, M.R.</td>
<td>11, 39.</td>
</tr>
<tr>
<td>Galil, Z.</td>
<td>278.</td>
</tr>
<tr>
<td>Gargiulo, T.</td>
<td>265.</td>
</tr>
<tr>
<td>Germa, A.</td>
<td>233.</td>
</tr>
<tr>
<td>Giammarco, A.</td>
<td>257.</td>
</tr>
<tr>
<td>Gibbons, P.B.</td>
<td>7, 63.</td>
</tr>
</tbody>
</table>
Mennecke, P. 179.
Messinger, E. 180, 181, 191.
Meyer, C. 180.
Meyers, S. 251.
Meyniel, H. 123.
Mezzalira, L. 67.
Miller, G.L. 269.
Miller, M. 215.
Miller, Z. 41.
Miriylia, K. 280.
Misra, A. 219.
Misue, K. 281, 303.
Moen, S. 279.
Moran, A. 106.
Mostue, B.M. 234.
Murtagh, T.P. 192.
Murty, U. 1.
Mutzel, P. 66.
Naggar, P. 242.
Nagl, M. 235.
Nakajima, K. 198.
Nakamura, K. 244.
Nardelli, E. 58, 72, 195, 196, 230, 240, 242, 301.
Nievrygelt, J. 5.
Nishioka, I. 57.
Nishizeki, I. 10, 84, 92, 93.
North, S.C. 42, 182, 184, 261.
Nummenmaa, J. 133, 227.
Onoguchi, K. 93.
Orrin, J.B. 41.
Ostho, H.G. 95.
Otten, R.HJ.M. 131.
Ozawa, T. 59, 84, 141.
Paeh, J. 102, 103.
Papakostas, A. 305.
Pato, J. 249.
Paxton, T.H. 105.
Pelco, A. 151, 152, 153, 156.
Pietrosanti, E. 222.
Piperino, A. 25.
Platt, C. 160.
Puueli, A. 213.
Pollack, R. 102, 103.
Pollak, H.O. 286.
Pothoff, A. 253.
Preparata, F.P. 13.
Protsko, L.B. 263.
Rademacher, H. 88.
Raghavachari, B. 273, 274.
Ramachandran, V. 266, 267.
Rappaport, D. 215.
Read, R. 97, 243.
Reagiani, M.G. 211.
Reif, J.H. 266, 267, 269, 270.
Reinier, D. 224, 225, 226.
Reingold, E. 5, 20, 24, 49.
Reiss, S. 249, 250, 251, 285.
Rheingans, P. 225.
Richard, J. 232.
Richelli, G. 242.
Rival, I. 150, 151, 152, 153, 156, 157, 158, 159, 164.
Robertson, G.G. 284.
Robins, G. 193.
Rosenthal, A. 224, 225.
Rowe, L.A. 180, 191.
Sahni, S. 3, 143.
Saint-Paul, A. 220.
Sandberg, J. 42.
Santucci, G. 257.
Sarkar, M. 236.
Savage, J.E. 194.
Schaefer, D.A. 263.
Schlag, M. 122.
Schneider, W. 104.
Seery, J.B. 51.
Shannon, A. 18.
Shannon, G. 252.
Sherlekar, D. 290.
Shieber, S. 793.
Shiloach, Y. 110.
Shirakawa, I. 57.
Shor, P. 217.
Simon, J. 268.
Simion, T. 307.
Skiena, S. 262.
Smart, J.C. 260.
Sorenson, P.G. 263.
Sotteau, D. 233.
Spencer, T.H. 278.
Spirakis, C. 180.
Stein, J.P. 146.
Steinitz, E. 88.
Storer, J.A. 114.
Strani, M. 256.
Sugiyama, K. 174, 175, 176, 186, 189, 210, 281, 303.
Supowit, K. 24.
Suzuki, T. 244.
<table>
<thead>
<tr>
<th>Author(s)</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swaminathan, V.</td>
<td>219</td>
</tr>
<tr>
<td>Swamy, M.N.S.</td>
<td>60, 61, 148</td>
</tr>
<tr>
<td>Sweet, R.E.</td>
<td>17</td>
</tr>
<tr>
<td>Szkatula, K.</td>
<td>206</td>
</tr>
<tr>
<td>Tagawa, S.</td>
<td>174</td>
</tr>
<tr>
<td>Takahashi, H.</td>
<td>59, 76</td>
</tr>
<tr>
<td>Takvorian, A.</td>
<td>65</td>
</tr>
<tr>
<td>Talamo, M.</td>
<td>58, 72, 228, 240</td>
</tr>
<tr>
<td>Tallot, D.</td>
<td>232</td>
</tr>
<tr>
<td>Tarjan, R.E.</td>
<td>9, 55, 77, 78, 81, 132</td>
</tr>
<tr>
<td>Tarui, Y.</td>
<td>244</td>
</tr>
<tr>
<td>Thomas, W.</td>
<td>253</td>
</tr>
<tr>
<td>Thomassen, C.</td>
<td>91, 124, 163</td>
</tr>
<tr>
<td>Thulasiraman, K.</td>
<td>60, 61, 148</td>
</tr>
<tr>
<td>Tichy, W.F.</td>
<td>245, 248</td>
</tr>
<tr>
<td>Tilford, J.</td>
<td>20, 21</td>
</tr>
<tr>
<td>Toda, M.</td>
<td>174, 175</td>
</tr>
<tr>
<td>Trelecan, J.P.</td>
<td>263</td>
</tr>
<tr>
<td>Trickey, H.</td>
<td>75, 120</td>
</tr>
<tr>
<td>Tuan, A.</td>
<td>180</td>
</tr>
<tr>
<td>Tunkelang, D.</td>
<td>52</td>
</tr>
<tr>
<td>Tuomi, J.</td>
<td>227</td>
</tr>
<tr>
<td>Tutte, W.T.</td>
<td>89, 90</td>
</tr>
<tr>
<td>Urrutia, J.</td>
<td>152</td>
</tr>
<tr>
<td>Valiant, L.</td>
<td>111</td>
</tr>
<tr>
<td>van Dam, A.</td>
<td>12</td>
</tr>
<tr>
<td>Van Hentenryck, P.</td>
<td>299</td>
</tr>
<tr>
<td>van Leeuwen, J.</td>
<td>40</td>
</tr>
<tr>
<td>van Wijk, J.G.</td>
<td>131</td>
</tr>
<tr>
<td>Vargiu, F.</td>
<td>113, 256</td>
</tr>
<tr>
<td>Vaucher, J.</td>
<td>19</td>
</tr>
<tr>
<td>Vemuri, V.</td>
<td>260</td>
</tr>
<tr>
<td>Vijayan, G.</td>
<td>117, 304</td>
</tr>
<tr>
<td>Vismara, L.</td>
<td>306</td>
</tr>
<tr>
<td>Vitter, J.S.</td>
<td>118, 119, 271, 272</td>
</tr>
<tr>
<td>Vo, K.P.</td>
<td>182, 184, 261</td>
</tr>
<tr>
<td>Wagner, K.</td>
<td>85</td>
</tr>
<tr>
<td>Walker II, J.Q.</td>
<td>22</td>
</tr>
<tr>
<td>Ward, N.</td>
<td>106</td>
</tr>
<tr>
<td>Warfield, J.</td>
<td>201</td>
</tr>
<tr>
<td>Watanabe, H.</td>
<td>53</td>
</tr>
<tr>
<td>Welzl, E.</td>
<td>307</td>
</tr>
<tr>
<td>Wermuth, U.</td>
<td>253</td>
</tr>
<tr>
<td>Wetherell, C.</td>
<td>18</td>
</tr>
<tr>
<td>Whitesides, S.</td>
<td>215</td>
</tr>
<tr>
<td>Wigderson, A.</td>
<td>117</td>
</tr>
<tr>
<td>Wille, R.</td>
<td>150</td>
</tr>
<tr>
<td>Wismath, S.</td>
<td>125, 129</td>
</tr>
<tr>
<td>Woeginger, G.</td>
<td>307</td>
</tr>
<tr>
<td>Wong, C.K.</td>
<td>122, 128</td>
</tr>
<tr>
<td>Wood, D.</td>
<td>23</td>
</tr>
<tr>
<td>Woods, D.</td>
<td>140</td>
</tr>
<tr>
<td>Wormald, N.</td>
<td>96, 199, 200</td>
</tr>
<tr>
<td>Yamanouchi, T.</td>
<td>92</td>
</tr>
<tr>
<td>Yannakakis, M.</td>
<td>214</td>
</tr>
<tr>
<td>Zischler, H.</td>
<td>235</td>
</tr>
</tbody>
</table>