Graph Drawing '93

Proceedings of the ALCOM International Workshop on Graph Drawing
Sèvres, Parc of Saint Cloud, Paris
September 25–29, 1993

Edited by
G. Di Battista, P. Eades, H. de Fraysseix,
P. Rosenstiehl, and R. Tamassia

*** DRAFT, do not distribute ***
September 21, 1993

ALCOM International Workshop on Graph Drawing

Sèvres, Parc of Saint Cloud, Paris
September 25–29, 1993

Graph drawing addresses the problem of constructing geometric representations of abstract graphs and networks. It is an emerging area of research that combines flavors of graph theory and computational geometry. The automatic generation of drawings of graphs has important applications in key computer technologies such as software enginering, database design, and visual interfaces. Further challenging applications can be found in architectural design, circuit schematics, and project management. Research on graph drawing has been especially active in the last decade. Recent progress in computational geometry, topological graph theory, and order theory has considerably affected the evolution of this field, and has widened the range of issues being investigated.

This first international workshop on graph drawing covers major trends in the area. Papers describe theorems, algorithms, graph drawing systems, mathematical and experimental analyses, practical experience, and a wide variety of open problems. Authors come from diverse academic cultures: from graph theory, computational geometry, and software engineering.

Support from ALCOM II ESPRIT B-A 7147 and by EHESS is gratefully acknowledged; we would also like to thank Mr. David Montgomery for preparing this document.

Giuseppe Di Battista (Univ. Rome, Italy) dibattista@iasi.rm.cnr.it
Peter Eades (Univ. Newcastle, Australia) eades@cs.newcastle.edu.au
Hubert de Fraysseix (CNRS, France) prosenst@dmi.ens.fr
Pierre Rosenstiehl (EHESS, France) prosenst@dmi.ens.fr
Roberto Tamassia (Brown Univ., USA) rt@cs.brown.edu

Contents

J. Pach. *New Developments in Geometric Graph Theory* 4
Prosenjit Bose, William Lenhart and Giuseppe Liotta. *Characterizing Proximity Trees* 4
Franz J. Brandenburg and Peter Eades. *A Note on a Separator Algorithm for Tree Embeddings and its Application to Free Tree Drawings* 6
Brian Regan. *Two Algorithms for Drawing Trees in Three Dimensions* 8
Goos Kant, Giuseppe Liotta, Roberto Tamassia and Ioannis G. Tollis. *Area Requirement of Visibility Representations of Trees* 11
Ashim Garg and Roberto Tamassia. *Efficient Computation of Planar Straight-Line Upward Drawings* 14
Uli Fößmeier and Michael Kaufmann. *An Approach for Bend-Minimal Upward Drawing* 15
C.Thomassen. *Representations of Planar Graphs* 17
Patrice Ossona de Mendez. *On Lattice Structures Induced by Orientations* 17
Jan Kratochvíl and Jiří Matoušek. *Complexity of Intersection Classes of Graphs* . . . 20
Hubert de Fraysseix, Patrice Ossona de Mendez and Pierre Rosenstiehl. *On Triangle Contact Graphs* 21
Simone Pimont and Michel Terrenoire. *Characterisation and Construction of the Rectangular Dual of a Graph.* 24
Goos Kant and Xin He. *Two Algorithms for Finding Rectangular Duals of Planar Graphs* 26
Goos Kant. *A More Compact Visibility Representation* 28
Anna Lubiw. *Cone Visibility Graphs* 30
Bojan Mohar. *Circle Packing Representations in Polynomial Time* 32
Bojan Mohar, Martin Juvan and Jože Marinček. *Obstructions for Embedding Extension Problems* 32
Antoine Bergey. *Automorphisms and Genus on Generalised Maps* 34
Ivan Rival. *Upward Drawing on Surfaces* 36
Bojan Mohar and Pierre Rosenstiehl. *Tessalation and Visibility Representations of Maps on the Torus* 38
Teresa M. Przytycka and Józef H. Przytycki. *A Simple Construction of High Representativity Triangulations.* 38
Prosenjit Bose, Hazel Everett, Sándor Fekete, Anna Lubiw, Henk Meijer, Kathleen Romanik, Tom Shermer and Sue Whitesides. *On a Visibility Representation for Graphs in Three Dimensions* 40
Jianer Chen, Saroja P. Kanchi and Jonathan L. Gross. *On Graph Drawings with Smallest Number of Faces* 41
Marc Bousset. *A Flow Model of Low Complexity for Twisting a Layout* 43
Giuseppe Di Battista, Giuseppe Liotta, and Francesco Vargiu. *Convex and non-Convex Cost Functions of Orthogonal Representations* 44
Giuseppe Di Battista and Luca Vismara. *Topology and Geometry of Planar Triangular Graphs* 47
Frank Dehne, Hristo Djidjev and Jörg-Rüdiger Sack. *An Optimal PRAM Algorithms for Planar Convex Embedding* 50
Miki Shimabara Miyauchi. *Algorithms for Embedding Graphs Into a 3-page Book* . . . 52
Hossam ElGindy, Michael Houle, Bill Lenhart, Mirka Miller, David Rappaport and Sue Whitesides. *Dominance Drawings of Bipartite Graphs* 54
Ulrich Finke and Klaus Hinrichs. *Computing the Overlay of Regular Planar Subdivisions in Linear Time* 56
Alain Denise. *Generation of Random Planar Maps* 57
Joseph Manning. *Symmetric Drawings of Graphs* 58

Tomaž Pisanski. *Recognizing Symmetric Graphs* 60
Peter Eades and Tao Lin. *Algorithmic and Declarative Approaches to Aesthetic Layout* 61
Isabel F. Cruz, Roberto Tamassia and Pascal Van Hentenryck. *A Visual Approach to Graph Drawing* 64
Gaby Zinßmeister. *Layout of Trees with Attribute Graph Grammars* 66
Sandra P. Foubister and Colin Runciman. *The Display, Browsing and Filtering of Graph-trees* 70
Daniel Tunkelang. *A Layout Algorithm for Undirected Graphs.* 73
Stephen C. North. *Drawing Ranked Digraphs with Recursive Clusters* 75
Ioannis G. Tollis and Chunliang Xia. *Graph Drawing Algorithms for the Design and Analysis of Telecommunication Networks* 75
Michael Himsolt. *A View to Graph Drawing Algorithms through GraphEd* 78
C. L. McCreary, C. L. Combs, D. H. Gill and J. V. Warren. *An Automated Graph Drawing System Using Graph Decomposition* 80
Michael Jünger and Petra Mutzel. *Maximum Planar Subgraphs and Nice Embeddings: Practical Layout Tools* 82
Peter Eades and Xavier Mendonça. *Heuristics for Planarization by Vertex Splitting* . . 83
Takao Ozawa. *Planar Graph Embedding with a Specified Set of Face-Independent Vertices* 85
Bjorn Sigurd Benestad Johansen. *Implementation of the Planarity Testing Algorithm by Demoucron, Malgrange and Pertuiset* 87
Joel Small. *A Unified Approach to Testing, Embedding and Drawing Planar Graphs* . 89
Hermann Stamm-Wilbrandt. *A Simple Linear-Time Algorithm for Embedding Maximal Planar Graphs* 92
H. de Fraysseix and P. Rosenstiehl. *The Left-Right Algorithm for Planarity Testing and Embedding* 94
P. O. de Mendez. *Connexity of Bipolar Orientations* 95

New Developments in Geometric Graph Theory

J. Pach.

Abstract not Available.

Characterizing Proximity Trees

Prosenjit Bose,* William Lenhart,† and Giuseppe Liotta‡

Much attention has been given over the past several years to developing algorithms for embedding abstract graphs in the plane such that the resulting drawing has certain geometric properties. For example, those graphs which admit planar drawings have been completely characterized and efficient algorithms for producing planar drawings of these graphs have been designed ([4], [9]). For an overview of graph drawing problems and algorithms, the reader is referred to the excellent bibliography of Di Battista, Eades, Tamassia and Tollis [2]. Moreover, many problems in pattern recognition and classification, geographic variation analysis, geographic information systems, computational geometry, computational morphology, and computer vision use the underlying structure present in a set of data points revealed by means of a *proximity graph*. A proximity graph attempts to exhibit the relation between points in a point set. Two points are joined by an edge if they are deemed *close* by some proximity measure. It is the measure that determines the type of graph that results. Many different measures of proximity have been defined. The relatively closest graph [6], the relative neighborhood graph [10], the gabriel graph [3], the modified gabriel graph [1] and the delaunay triangulation are but a few of the graphs that arise through different proximity measures.

An extensive survey on the current research in proximity graphs can be found in Jaromczyk and Toussaint [5]. As the survey suggests, interest in proximity graphs has been increasing rapidly in the last few years, but most of the interest has been algorithmic and little attention has been given to the combinatorial characteristics of these graphs. Monma and Suri [8] show that any tree with maximum vertex degree five can be drawn as a minimum spanning tree.

We study the problem of drawing trees as certain types of proximity graphs. We say that a tree T can be drawn as a proximity graph when there exists a set of points in the plane such that the proximity graph of that set of points is isomorphic to T. Some sufficient conditions for such drawings to exist have been given by Cimikowski [1] for relatively closest graphs, by Matula and Sokal [7] for gabriel graphs and by Urquhart [11] for relative neighborhood graphs. However, there exists no complete combinatorial characterization for any of the these types of proximity graphs. To this end, we give a complete characterization of the trees that can be realized as either the relative neighborhood graph, relatively closest graph, gabriel graph or modified gabriel graph of a set of points in the plane.

*Research supported in part by NSERC and FCAR. School of Computer Science, McGill University, 3480 University, Montréal, Québec, H3A 2A7. jit@muff.cs.mcgill.ca

†Department of Computer Science, Williams College, Williams- town, MA 01267. This work has been done when this author was visiting the School of Computer Science of McGill University. lenhart@cs.williams.edu

‡Dipartimento di Informatica e Sistemistica, Università di Roma 'La Sapienza', via Salaria 113, I-00198 Roma, Italia. This work has been done when this author was visiting the School of Computer Science of McGill University. liotta@infokit.ing.uniroma1.it

The sufficiency of the conditions of our theorems is established by providing drawing algorithms. Given an abstract tree that admits a particular type of drawing, say as a gabriel graph, the drawing algorithm will compute a set of points such that the gabriel graph of the set of points is a tree that is isomorphic to the given tree. As for the necesssity, for each type of proximity graph considered, we exhibit a set of trees, called forbidden trees, which we show cannot be drawn as that type of proximity graph. We then prove that no tree which can be drawn as such a proximity graph can contain any of the forbidden trees.

Theorem 1 *A tree T can be drawn as the relatively closest graph or relative neighborhood graph if and only if the maximum vertex degree of T is at most 5. The drawing can be obtained in $O(n)$ time, where n is the number of vertices in T, in the real RAM model.*

Theorem 2 *A tree T can be drawn as the modified gabriel graph if and only if the maximum vertex degree of T is at most three. This drawing can be obtained in $O(n)$ time where n is the number of nodes in T, in the real RAM model.*

The characterization of trees which can be drawn as gabriel graphs depends on a set of subgraphs called *wide trees*. A *rooted tree* (T, u) consists of a tree T and a distinguished vertex u of T, called the *root* of T. Given a tree T, a vertex v of T and a neighboring vertex x of v, $T_x(v)$ is defined to be the component of $T - v$ containing x.

Definition 1: A rooted tree (T, u) is *wide* if

1. $deg(u) = 2$ or
2. $deg(u) = 1$, u has neighbor v of degree four and for each neighbor x of v, $x \neq u$, $(T_x(v), x)$ is wide.

Definition 2: A tree T is *forbidden* if it contains any of the following

1. A vertex of degree at least five
2. Two adjacent vertices of degree four
3. A vertex v of degree four such that for each neighbor x of v, the subtree $(T_x(v), x)$ is wide.

Theorem 3 *A tree T can be drawn as the gabriel graph if and only if T is not forbidden. This drawing can be obtained in $O(n)$ time where n is the number of nodes in T, in the real RAM model.*

References

[1] R. Cimikowski. Properties of some Euclidean proximity graphs. *Pattern Recognition Letters,* **13**, pp. 417-423, 1992.

[2] G. Di Battista, P. Eades, R. Tamassia and I. G. Tollis. Algorithms for Automatic Graph Drawing: An Annotated Bibliography. Dept. of Computer Science, Brown Univ., Technical Report, 1993.

[3] K. R. Gabriel and R. R. Sokal. A New Statistical Approach to Geographical Analysis. *Systematic Zoology,* **18**, 1969, pp. 54-64.

[4] J. Hopcroft and R.E. Tarjan. Efficient Planarity Testing. *J. ACM,* **21, 4**, pp. 549-568, 1974.

[5] J.W. Jaromczyk and G.T. Toussaint. Relative Neighborhood Graphs and Their Relatives. *Proceedings of the IEEE,* **80, 9**, pp. 1502-1517, 1992.

[6] P. M. Lankford. Regionalization: Theory and Alternative Algorithms. *Geographical Analysis,* **1**, 1969, pp. 196-212.

[7] D.W. Matula and R.R. Sokal. Properties of Gabriel Graphs Relevant to Geographic Variation Research and the Clustering of Points in the Plane. *Geographical Analysis,* **12, 3**, pp. 205-222, 1992.

[8] C. Monma and S. Suri. Transitions in Geometric Minimum Spanning Trees. Bell Communications Research Technical Report, 1991.

[9] T. Nishizeki and N. Chiba. Planar Graphs: Theory and Algorithms. *Annals of Discrete Mathematics,* North Holland, 1988.

[10] G.T. Toussaint. The relative neighborhood graph of a finite planar set. *Pattern Recognition,* **12**, pp. 261-268, 1980.

[11] R.B. Urquhart. Some properties of the planar Euclidean relative neighbourhood graph. *Pattern Recognition Letters,* **1, 5**, pp. 317-322, 1983.

A Note on a Separator Algorithm for Tree Embeddings and its Application to Free Tree Drawings

Franz J. Brandenburg* **and Peter Eades†**

Divide and Conquer is one of the major principles for algorithm design. It leads to fast algorithms with good outputs, particularly if the division partitions a problem instance into two parts of almost equal size. For a problem instance of size n the partition tree of solvable subproblems is then a complete binary tree of size $2n-1$. This means only a linear growth in size or an $O(1)$ expansion. However, if there is no strict partition into equal sized subproblems, then the partition tree may be any binary tree. Binary trees are 1-separable, i.e., they admit a partition into two subtrees in the range 1/3..2/3. In fact, for every $k \leq 3\ n$, where n is the size of the tree, there is a subtree of size m and $k/3 \leq m \leq 2k/3$. Moreover, there is a fast search algorithm for a proper subtree in the 1/3..2/3 range, searching in the bigger subtree until its size falls under $2k/3$. There may be a subtree t' which approximates $k/2$ much better than the guaranteed range of $k/3..2k/3$, and this t' may lie somewhere in a smaller branch. These well-known strategies are used in our embedding algorithm.

Algorithm DFS-EMBED

DFS-EMBED maps a binary tree t into a complete binary tree t' such that each node and each edge of t' are visited twice according to a depth-first traversal. Moreover, the root of t is pinned to the root of t'. t' should be big enough. It is easily reduced to minimal size.

Let $S = v_0, v_1, \ldots, v_p$ be the sequence of nodes of t' visited in a dfs traversal by the recursion: root, left, right, root. Let $P = x_0, x_1, \ldots, x_{q+1}$ be the path in t from the root to some designated

*Universität Passau Lehrstuhl für Informatik 94030 Passau, Germany. brandenb@informatik.uni-passau.de

†University of Newcastle Department of Computer Science Callahan, NSW 2308, Australia. eades@cs.newcastle.edu.au

node x_{q+1} selected by a good strategy. For $i = 0, \ldots, q$ let y_i be the other child of x_i and let t_i denote the subtree of t with root y_i. Let t_{q+1} be the subtree with root x_{q+1}. Rearrange P to the sequence $Q = x_1, \ldots, x_q, x_{q+1}, x_0$ and consider the sequence of subtrees $t_1, \ldots, t_q, t_{q+1}, t_0$. Rename x_0 into x_{q+2} and t_0 into t_{q+2}, such that $Q = x_1, \ldots, x_{q+2}$.

For $i = 1, \ldots, q+2$, if t_i is nonempty then map its root y_i to the next vertex $v_{j(i)}$ in the sequence S such that the vertex $v_{j(i)}$ was unmarked and - by recursion - t_i fits into the complete subtree of t' with root $v_{j(i)}$. All vertices of this subtree will be doubly marked. Map the ancestor x_i of y_i to the ancestor w of $v_{j(i)}$, increase the mark of w and update all ancestors of w by a single mark.The edge from x_i to y_i is directly mapped into the tree edge of t'. If t_i is empty, then map x_i to the next not yet doubly marked vertex in the sequence S and increase the mark by one. It remains to route the edges on the path $x_0, x_1, \ldots, x_{q+1}$ and from x_0 to the second child x_{q+2}. For $i = 1, \ldots, q+1$, the edge from x_{i-1} to x_i is mapped to the path according to the *dfs* traversal of t', with the edge from x_0 to x_1 along the first traversal and the others along the second. The edge from x_0 to x_{q+2} is routed along the second traversal from x_{q+2} to x_0.

For a illustration of DFS-EMBED draw $x_1, \ldots, x_{q+1}$ as the rightmost path in the left subtree with each t_i below x_i and let t_0 be the right subtree of the root x_0.

Algorithm DFS-EMBED has the following properties:

Lemma: DFS-EMBED runs in linear time.

Lemma: Let t be a binary tree of size n. If DFS-EMBED chooses the path $P = x_0, x_1, \ldots, x_{q+1}$ such that

(i) for $i = 1, \ldots, q+1$ the subtree with root x_i is bigger than the other subtree t_i or

(ii) $1/4\, size(t) \leq size(t_0) + size(t_{q+1}) \leq 3/4\, size(t)$ or

(iii) (i) and (ii) hold

then DFS-EMBED(t) is a complete binary tree with height at most $2 \cdot \log n$.

Proof: (Sketch) DFS-EMBED maps a sequence of chained trees $T = (x_1, t_1, \ldots, x_q, t_q)$, where x_i is the ancestor of t_i and x_{i+1} into a complete tree whose height exceeds the height of an almost optimal binary search tree at most by one. The access distribution for the search tree is $1/W$ for each x_i and $2^{log n_i + 1} - 1/W$ for each t_i of size n_i with $W = 2 \sum 2^{\log n_i}$. □

We wood like to have an additive increase of the height only, but for DFS-EMBED the factor $c = 2$ is best possible.

Lemma: There is an infinite sequence of binary trees T_i such that DFS-EMBED (T_i) produces complete binary trees of height $2 \cdot \log(n_i)$ with $n_i = size(T_i)$.

Proof: (Sketch) Let the left subtree of T_i consist of a new vertex with two copies of T_{i-1} as its subtrees and let the right subtree of T_i be trivial, consisting e.g. of two vertices. Let T_0 be the complete tree with three vertices. Then T_i has $14 \cdot 2^i - 4$ vertices. By induction, one of the two T_{i-1} needs height at least $2 \cdot \log(size(T_{i-1}))$, and DFS-EMBED sets the root of this T_{i-1} at depth 2 . □

Corollary: Every binary tree t of size n can be embedded into a complete binary tree t' of height $c \cdot \log n$ with $c \leq 2$, congestion 2 and dilation $O(\log n)$. Thus, the expansion is $O(n)$, each edge of t' is used at most twice and each edge of t is mapped into a path of length at most $c \cdot \log n$.

Remark: The linear expansion obtained from the factor $c = 2$ for the increased height is better than the one obtained by Hong et al [HMR] and Ruzzo and Snyder [RS]. They demand bounded dilation and then achieve $c = 5$ (Ruzzo, personal communication) which yields an expansion of $O(n^4)$.

For free tree drawings or equivalently planar embeddings of binary trees one may use DFS-EMBED as an intermediate step, but this is a detour.

Corollary: For every binary tree t of size n there is a free tree drawing $d(t)$ derived from tree drawings of complete binary trees, such that $d(t)$ uses $O(n^2)$ area and has edges with $O(\log n)$ bends.

This result is non-competitive. Valiant [Va] has shown that there are planar tree embeddings with $O(n)$ area and $O(\log n)$ bends and Crescenci et al. [CBP] have upwards drawings with $O(n \cdot \log n)$ area and no bends

References

[CBP] P. Crescenci, G. Di Battista, and A. Piperno *"A note on optimal area algorithms for upward drawings of binary trees."* Computational Geometry: Theory and Applications, 2, pp. 187-20 (1992)

[BET] G. Di Battista, P. Eades and R. Tamassia *"Algorithms for Drawing Graphs: An Annotated Bibliography"* (1993)

[HMR] J.W. Hong, K. Mehlhorn, A. L. Rosenberg *"Cost Trade-offs in Graph Embeddings, with Applications"* Journal of the Assoc. for Comput. Machinery, Vol. 30, No. 4, pp. 709-728(1983)

[Me] K. Mehlhorn *"Data Structures and Algorithms"* EATCS Monograph Computer Science (1984)

[RS] W.L. Ruzzo and L. Snyder *"Minimum Edge Length Planar Embeddings of Trees"* In H.T. Kung, B. Sproull, and G. Steele, ed., VLSI Systems and Computations, pp.119-123 (1981)

[Va] L.G. Valiant *"Universality Considerations in VLSI Circuits"* IEEE Transactions on Computers, C-30, pp. 135-140 (1981)

Two Algorithms for Drawing Trees in Three Dimensions

Brian Regan*

This paper explores the presentation of tree structures in three dimensions. Some visualization systems have begun to use three dimensional tree drawings [8]; however, the wealth of fundamental results on two dimensional tree layout (for example, [12, 6, 10, 10, 4, 11, 1, 14, 13, 10, 2, 5]) is not matched in three dimensions. We are specifically interested in drawing *free* trees, that is, trees with no pre-specified root.

Two extensions to existing tree drawing algorithms were developed. The first takes the H-tree drawing algorithm for binary trees and extends it into three dimensions. This additional dimension is achieved by alternating the direction of the next node between three rather than two axes. The H-tree algorithm produces a grid drawing. The second algorithm takes the principle of a radial drawing algorithm [3] and extends it into three dimensions. The successive annuli of the radial algorithms are replaced by successive spherical shells sharing a common

*Department of Management, University of Newcastle, New South Wales, AUSTRALIA.

centre. Working from a root node at the centre, for each node an area for its descendants is projected onto the next shell. This area is compared with the area formed by the intersection of the tangent plane at the given node with the next outer shell. This second area is the equivalent of the annulus wedges formed in the two dimensional case. The smaller of these two areas is subdivided according to the number of children of the original node. The subdivision is accomplished by using a Lambert equal area projection of the sphere onto a plane [9], and allocating rectangles for each of the child nodes, from which spherical coordinates of the nodes are then determined. Improvement to the final layout will be gained from further refinement to the allocation algorithm for the projected surface area.

One of the main aims of this study is to develop measures for the effectiveness of three dimensional tree drawings. In particular, the conventional measure of the *area* of a grid drawing has an equivalent in terms of *volume* when working in three dimensions. However, this measure loses its relevance when the graph is drawn in a plane: the volume drops to zero. In addition, when considering the display of the graph on a screen, it becomes obvious that the crucial measure of the graph's image is the maximum size of its sides. With such a measure we can determine whether the full graph will remain visible after the application of any rotation. Thus we define the *size* of a grid drawing of a graph to be the maximum length of a side of its smallest enclosing isothetic rectangular prism. We believe that this measure is especially relevant to orthogonal drawings, that is, drawings in which edges are parallel to one of the coordinate axes. The size of an H-tree presentation for any binary tree within 3 dimensions with n nodes is $O(n)$.

Theorem 1 *The size of an H-Tree drawing in 3 dimensions of a complete binary tree with n nodes is* $O\left(n^{1/3}\right)$.

The radial algorithm above does not give grid drawings and so the size measure loses its significance.

The *diameter* of a graph drawing in three dimensions is the maximum distance between any two vertices. We propose two further measures for the quality of a graph drawing in three dimensions:

- The ratio λ/l, where l is the length of the shortest edge and λ is the length of the longest edge. Long edges are difficult to follow and we believe a large value for this ratio characterises a poor layout.
- The ratio η/d, where d is the diameter of a drawing and η is the shortest distance between a (adjacent or nonadjacent) pair of nodes. In this case, a large value is an indication of a good layout.

Theorem 2 *If l is the length of the shortest edge, and λ is the length of the longest edge in a drawing of a tree with n nodes obtained from the radial algorithm, then λ/l is* $O\left(\sqrt{n}\right)$.

Theorem 3 *If d is the diameter and η is the shortest distance between a pair of nodes in a drawing of a tree with n nodes obtained from the radial algorithm, then η/d is* $\Omega\left(\frac{1}{n}\right)$.

Both our algorithms have been implemented in Windows and Unix environments. Samples of binary trees have been processed against the H-Tree algorithm and a varied collection of trees were processed with the radial algorithm. These samples give some support for the proposed measures; further, we believe that they form an interesting collection which may become useful for benchmarking future three dimensional tree drawing algorithms.

References

[1] F. J. Brandenburg. Nice drawing of graphs and trees are computationally hard. Technical Report MIP-8820, Fakultat fur Mathematik und Informatik, University of Passau, 1988.

[2] 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*, 2:187–200, 1992.

[3] P. Eades. Drawing free trees. Technical Report IIAS-RR-91-17E, International Institute for Advanced Study of Social Information Science, 1991.

[4] P. Eades, X. Lin, and T. Lin. Two tree drawing conventions. (to appear in *Computational Geometry and Applications*), 1993.

[5] A. Garg, M.T. Goodrich and R. Tamassia. Area-Efficient Upward Tree Drawings. Proc. ACM Symp. on Computational Geometry, 1993.

[6] J. Manning and M.J. Atallah. Fast detection and display of symmetry in trees. *Congressus Numerantium*, 1989.

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

[8] G. G.Robertson, J. D. Mackinley and S. K.Card. Cone Trees: Animated 3D visualizations of hierarchical information. *Proc. CHI*, pp. 189-193, 1991.

[9] J. Snyder. Map Projections - A Working Manual. *US Geological Survey Professional Paper 1395*, US Government Printing Office, Washington 1987.

[10] K. Supowit and E. Reingold. The complexity of drawing trees nicely. *Acta Informatica*, 18:377 – 392, 1983.

[11] J. S. Tilford. Tree drawing algorithms. Master's thesis, Department of Computer Science, University of Illinois at Urbana Champaign, 1981.

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

[13] J. Vaucher. Pretty printing of trees. *Software Practice and Experience*, 10(7):553 – 561, 1980.

[14] C. Wetherall and A. Shannon. Tidy drawings of trees. *IEEE Transactions on Software Engineering*, SE-5(5):514 – 520, 1979.

Area Requirement of Visibility Representations of Trees *

Goos Kant, † Giuseppe Liotta, ‡ Roberto Tamassia, § and Ioannis G. Tollis ¶

The problem of drawing a graph in the plane has received increasing attention recently due to the large number of applications [1]. Examples include VLSI layout, algorithm animation, visual languages and CASE tools. Vertices are usually represented by points and edges by simple open curves. Another interesting representation is to map vertices into horizontal segments and edges into vertical segments [7, 11]. Such a representation is called a *visibility representation*. In this paper we study the area requirement of various types of visibility representations of trees, and we present linear time algorithms for drawing such representations with optimal area.

The concept of *visibility* between objects plays an important role in various problems of computational geometry, where we say that two objects of a given set are *visible* if they can be joined by a segment which does not intersect any other object. Two objects of the set are *ϵ-visible* if they can b e joined by a non-zero thickness band which doesn't intersect any other object. The objects are non overlapping. A visibility representation of a graph maps vertices into objects and edges into segments between visible vertex-objects. Various visibility representations have been considered in the literature, and received increasing attention recently (see [9] for an up to date overview).

Tamassia & Tollis [11] studied three types of visibility representations (weak, ϵ, and strong) of graphs. A *weak visibility representation* maps vertices to horizontal segments and edges to vertical segments having only points in common with the pair of horizontal segments corresponding to the vertices they connect. Algorithms for constructing weak visibility representations were presented in [11] and independently in [7]. This type of representation has become a core item in the field of graph drawing. Recently, Kant [8] showed that such a visibility representation can be constructed in linear time on a grid of size at most $(\lfloor \frac{3}{2}n \rfloor - 3) \times (n-1)$. A *strong visibility representation* maps vertices to horizont al segments such that two segments are visible if and only if the corresponding vertices are adjacent [11]. Tamassia & Tollis showed that every triangular planar graph and 4-connected planar graph has a strong visibility representation [11]. However, deciding whether a general planar graph has a strong visibility representation is NP-complete [1].

Several years after the publication of the first papers, researchers started the study of the 2-dimensional variant of this problem: vertices are represented by isothetic rectangles, and edges are represented by horizontal or vertical segments, having only points in common with the pair of rectangles corresponding to the vertices they connect. We only consider rectangles with non-zero area and sides parallel to the x-axis and y-axis. This representation is called *2-weak visibility representation*. In [12] Wismath proves that every planar graph admits a *2-ϵ visibility representation*, that is a 2-weak visibility representation representation with the ad ditional property that two rectangles are ϵ-visible if and only if the corresponding vertices in the graph

*Research supported in part by the National Science Foundation under grant CCR-9007851, by the U.S. Army Research Office under grant DAAL03-91-G-0035, by the Office of Naval Research and the Advanced Research Projects Agency under contract N00014-91-J-4052, ARPA order 8225, and by ESPRIT Basic Research Action No. 7141 (Project ALCOM II). This work was performed in part at the Bellairs research Institute of McGill University.

†Department of Computer Science, Utrecht.University P.O. box 80.089, 3508 TB Utrecht, NL. goos@ cs.ruu.nl.

‡Dipartimento di Informatica e Sistemistica Universita' di Roma La Sapienza, 00185 Roma, Italy. This work has been done while this author was visiting the School of Computer Science of McGill University, Montreal. liotta@infokit.ing.uniroma1.it.

§Department of Computer Science, Brown University, Providence, RI 02912-1910. rt@cs.brown.edu.

¶Department of Computer Science, University of Texas at Dallas, Richardson, TX 75083-0688. tollis@utdallas.edu.

are adjacent. A *2-strong visibility representation* maps each vertex to a rectangle such that two rectangles are visible if and only if the corresponding vertices in the graph are adjacent.

Recently, the *area* of the representation has gained a lot of attention, especially in the field of graph drawing [11, 4, 5, 2]. Eades et al. in [6, 7] study different possible representations of trees under the constraint of minimizing the size of the drawing. For a complete survey on graph drawing see [1]. The *area* of a drawing is the area of the smallest rectangle with sides parallel to the axes covering the drawing. The *width* and the *height* of the drawing are the width and the height of the covering rectangle. We assume the existence of a *resolution rule*, which implies that the width and the height of a drawing cannot be arbitrarily scaled down. A typical resolution rule for 1- and 2- visibility representations is requiring for the endpoints of the vertex segments or vertex rectangles to be placed at the points of an integer grid. The existence of such a resolution rule naturally raises the problem of computing 1- and 2-visibility representations of a graph with minimum area.

We investigate the strong visibility problem for trees. The contribution is twofold. First we show lower bounds on the area occupied by any 1- and 2-strong visibility representation of trees. Next we present linear time drawing algorithms that obtain such representations achieving these bounds. Since in this paper we only study 1- and 2-*strong* visibility representations, we call them 1- and 2-visibility representations, for brevity.

In the rest of this abstract we give a list of the main results.

Theorem 1 *Let T be a rooted tree with n vertices, l leaves and height h. The area required by a 1-visibility representation $\Gamma(T)$ of T is $\Theta(h \cdot l)$. Also $\Gamma(T)$ can be computed in $O(n)$ time.*

Let T be a free (i.e. unrooted) tree. Let v be a vertex of T and let $T_v^1, \ldots, T_v^k$ be the subtrees obtained by removing v and the edges incident on v. We root each subtree T_v^i at the unique vertex of T_v^i, adjacent to v in T. We assume that always $h(T_v^1) \geq h(T_v^2) \geq \ldots \geq h(T_v^k)$ in this paper. T_v^k is called the k-th highest subtree of v.

We denote with T_v the tree obtained by deleting from T the first and the second subtree of v and the incident edge of v to T_v^1 and to T_v^2. Root T_v at v. We call the *third vertex* of T the vertex v^* such that the height of the third highest subtree of v^* is maximum, i.e., for which $h(T_{v^*})$ is maximum. Let k^* be the corresponding height, i.e., $k^* = h(T_{v^*})$.

Theorem 2 *Let T be a free tree with n vertices and l leaves; let k^* be the height of the third subtree of the third vertex of T. The area required by a 1-visibility representation $\Gamma(T)$ of T is $\Theta(k^* \cdot l + n)$. Also $\Gamma(T)$ can be computed in $O(n)$ time.*

Theorem 3 *Let T be a free tree with n vertices, l leaves and height h. The area required by a 2-visibility representation $\Gamma(T)$ of T is $\Theta(l \cdot n)$. Also $\Gamma(T)$ can be computed in $O(n)$ time.*

Acknowledgments

This research is a consequence of the authors' participation to the Workshop on Visibility Representations of Graphs organized by Sue Whitesides and Joan Hutchinson at the Bellairs Research Institute of McGill University, Feb. 12-19, 1993. We thank the other participants of the Workshop for useful discussions.

References

[1] T. Andreae, Some Results on Visibility Graphs, 1989, preprint.

[2] P. Crescenzi, G. Di Battista, and A. Piperno, A Note on Optimal Area Algorithms for Upward Drawings of Binary Trees, to appear in *Comp. Geometry: Theory and Applications.*

[3] G. Di Battista, P. Eades, R. Tamassia, and I.G. Tollis Algorithms for Automatic Graph Drawing: An Annotated Bibliography, Dept. of Comp. Science, Brown Univ., Technica l Report, 1993 . Available via anonymous ftp from wilma.cs.brown.edu (128.148.33.66), files /pub/gdbiblio.tex.Z and /pub/gdbiblio.ps.Z.

[4] G. Di Battista, R. Tamassia and I.G. Tollis, Constrained Visibility Representations of Graphs, *Inform. Process. Letters* 41 (1992), pp. 1–7.

[5] G. Di Battista, R. Tamassia and I.G. Tollis, Area Requirement and Symmetry Display of Planar Upward Drawings, *Discr. and Comp. Geometry* (1992), pp. 381–400.

[6] P. Eades, T. Lin, and X. Lin, Two tree Drawing Conventions, Key Centre for Software Technolog y, Dept. of Comp. Science, The Univ. of Queensland, Techinal Report No. 174, 1990. (to appear in *Computatio nal Geometry and Applications*).

[7] P. Eades, T. Lin, and X. Lin, Minimum Size h-v Drawings, *Advanced Visual interfaces* (Proc eedings of AVI 92), World Scientific Series in computer science Volume 36, 1992, pp. 386-394.

[8] G. Kant, A More Compact Visibility Representation, in: J. van Leeuwen (Ed.), *Proc. 19th Intern. Workshop on Graph-Theoretic Concepts in Comp. Science (WG'93)*, Lecture Notes in Comp. Science, Springer-Verlag, 1993, to appear.

[9] J. O'Rourke, Computational geometry column 18, *Int. Journal of Comp. Geometry & Appl.* 3 (1993), pp. 107–113.

[10] P. Rosenstiehl, and R. E. Tarjan, Rectilinear Planar Layouts andBipolar Orientations of Planar Graphs, *Discr. and Comp. Geometry* 1 (1986), pp. 343–353.

[11] R. Tamassia, and I. G. Tollis, A Unified Approach to Visibility Representations of Planar Graphs, *Discr. and Comp. Geometry* 1 (1986), pp. 321–341.

[12] S.K. Wismath, Bar-Representable Visibility Graphs and Related Flow Problems, Dept. of Comp. Science, Univ. of British Columbia, Technical Report, 1989.

Efficient Computation of Planar Straight-Line Upward Drawings *

Ashim Garg† **and Roberto Tamassia**†

An *upward* drawing of a digraph is such that the edges are curves monotonically increasing in the y-direction. Clearly, a digraph admits an upward drawing if and only if it is acyclic. Upward drawings effectively visualize hierarchical relationships, such as partial orders, PERT-diagrams, and is-a diagrams.

In this paper we investigate planar straight-line upward drawings of digraphs. We shall denote with n the number of vertices of the digraph currently being considered. Previous results are [1, 2]:

- A digraph admits a planar straight-line upward drawing if and only if it is a subgraph of a *planar st-digraph*, i.e., a planar acyclic digraph with one source and one sink, joined by an edge.
- A planar straight-line upward drawing of a planar st-digraph can be constructed in $O(n \log n)$ time. This algorithm assigns real coordinates to the vertices, and no bound on the area of the drawing is provided.
- There exists a family of planar acyclic digraphs that require exponential area in any planar straight-line upward drawing with vertices placed at grid points. Namely, for any positive integer n there exists an n-vertex planar acyclic digraph G_n such that any planar straight-line upward drawing of G_n with integer vertex coordinates has area $\Omega(\sqrt{2}^n)$.

Our new results are summarized as follows:

- We give an optimal $O(n)$-time algorithm for constructing a planar straight-line upward drawing of a planar st-digraph.
- We present the first NC parallel algorithm for constructing planar straight-line upward drawings. Our algorithm runs in time $O(\log^2 n)$ on a CRCW PRAM with n processors.
- We show that the exponential area lower bound for planar st-digraphs is tight and can be efficiently attained. Namely, we argue that the drawings produced by the aforementioned sequential and parallel algorithms have area $O(c^n)$, for some constant c.
- Both the parallel and sequential algorithms use integer arithmetic where the size of the operands is $O(A^{2/3})$, where A is the area of the drawing produced by the algorithm.
- We give a partial characterization of the area requirement of planar straight-line upward drawings of maximal planar st-digraphs, based on the "nesting" of separating triangles.

Our algorithms are based on a technique that first contracts a large ($\Omega(n)$-size) subset of edges, then recursively computes a drawing of the resulting subgraph, and finally restores the contracted edges to yield a drawing of the original digraph.

*Research supported in part by the National Science Foundation under grant CCR-9007851, by the U.S. Army Research Office under grant DAAL03-91-G-0035, and by the Office of Naval Research and the Defense Advanced Research Projects Agency under contract N00014-91-J-4052, ARPA order 8225.

†Department of Computer Science, Brown University, Providence, RI 02912–1910. {ag,rt}@cs.brown.edu

References

[1] G. Di Battista and R. Tamassia, "Algorithms for Planar Representation of Acyclic Digraphs", Theoretical Computer Science, vol. 61, pp. 175-198, 1988.

[2] G. Di Battista, R. Tamassia, and I.G. Tollis, "Area Requirement and Symmetry Display of Planar Upward Drawings", Discrete & Computational Geometry, vol. 7, no. 4, pp. 381-401, 1992.

An Approach for Bend-Minimal Upward Drawing

Uli Fößmeier * **and Michael Kaufmann** *

Besides minimizing the area of the embedding and edge lengths the minimization of the number of bends is the measure of the quality of a planar embedding. The only provably optimal algorithm for this criterion of Tamassia [6] reduces the problem to a min-cost-flow problem and has therefore a quite high runtime of $O(n^2 \cdot \log n)$. Several heuristics ([8, 11]) are must faster but do not guarantee optimality.

Our approach has the goal to achieve optimality in a simple way for a certain class of embeddings, namely upward-drawings. We show how to get optimal bounds on the number of bends in linear time. Upward drawings are a popular way to display acyclic digraphs such that all edges flow in the same direction (from bottom to top). Most algorithms for upward drawings for planar graphs concern the minimization of the area, display of symmetries and isomorphisms. The problem of rectilinear upward drawing does not arise in the literature so often (a variation appears in [1]).

We define a rectilinear drawing to be *upward* if all segments of the edges are either horizontally or upward, and for each node v with incoming/outgoing edges one incoming/outgoing edge is incident to v by a vertical segment. As in [6], we first determine a topological embedding of the graph, namely the direction each edge starts in and the sequence of bends on it. After this, a linear standard compaction algorithm computes the final embedding of the graph in the grid.

Here we describe only the first step. We assume that the graph is a subgraph of a 4-planar s-t-graph. Rectilinear upward drawings exists only for those graphs.

Basic Ideas

Before describing the algorithm we start with some observations. Let $G = (V, E)$ be an acyclic planar s-t graph with maximum degree 4 embedded in the plane. Every edge has two directions: a start direction (the way how the edge leaves a node) and an end direction (the way how the edge joins a node). Valid directions are only left, upward and right. If the start direction and the finish direction of an edge are different, the edge gets one or two bends.

Our algorithm assigns appropriate directions for the start and end segments to all edges and puts the fragments together by producing bends if necessary.

At any node there are at most two different ways of assigning the edges (i.e. the connections to the neighbours) to the pins (left, upward, right and downward). If a node has e.g. two exits, we can assign them either to the upper and right pin or to the left and upper pin. If we make the wrong decision there may be difficulties to connect the remaining edges (in the example,

*Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Sand 13, 72076 Tübingen, Germany. mk@informatik.uni-tuebingen.de

the entries). Therefore it is important to determine an order how to do the assignment. Clearly there is no problem with nodes with one or three entries (or exits respectively): In the former case it has to be the lower (upper) pin, in the latter case the left, lower and right (right, upper and left) pin in this order. Nodes of degree 3 or with two entries and two exits are called *critical*. For critical nodes we always have two choices.

It is well known [4] that the incoming edges of each node v of a planar s-t-graph appear consecutively around v, and so do the outgoing. This makes sense to say that an exit (entry) is at the left or at the right of another.

Lemma 1. *Let v be a node of degree 3 with two exit edges e_i and e_j, s.t. e_i is left of e_j. An arbitrary assignment of e_i to a pin increases the number of bends on e_j by at most one. There are no global effects on the embedding, i.e. the neighbours of v 'do not realize the difference'. An analogous fact holds for critical nodes with two entries.*

Proof: Let c be the number of bends in the case that the left and upper pins are the exit pins of v. We can simulate the other configuration (upper/right pins are exits) by turning the edge out of the upper pin to the left and the edge out of the right pin upward using two new bends. $\Diamond$

Lemma 2. *Let v be a critical node of degree 4. For every pair (e_i, e_j) of adjacent edges their bendcosts increase by at most two if e_i and e_j are assigned arbitrarily to pins. The situation of the neighbours is not changed thereby (no global effects).*

As already seen computing start and end directions of an edge connecting two non-critical nodes is easy. For an edge between a critical node v of degree 3 and a non-critical node Lemma 1 ensures that it is always good to choose the cheaper way to connect the non-critical node, because the other entry (exit) pin of v requires at most one bend more than its neighbour pin to connect any edge. Thus the chosen solution is not worse than the alternative. After this action v becomes non-critical (its other entry (exit) pin has no more choice).

The case of an edge from a non-critical node u to a critical node v of degree 4 is more complicated since three pins are influenced by fixing one; thus it is not good to apply the greedy strategy here.

Definition. *A critical node v is called maiden if less than two of its neighbours have already fixed their connection to v. If the two alternatives of the connections to the fixed neighbours have different costs, v is called decided and tie otherwise.*

Note that a tie node always has degree 4. Lemma 2 tells us that if a node is decided, the difference between the two alternatives is exactly 2 (two bends). With the argument above we can choose the cheaper one in such cases, because the two other pins require at most two bends more than in their optimal layout.

If there are no more decided nodes, i.e. all nodes still unfixed are tie, we compute a *component* $M \subseteq V$ of nodes such that every $v \in M$ is tie, M is connected in the underlying undirected graph and $|M|$ is maximal. We take a node on the border of M (i.e. connected with two nodes $\notin M$) and assign its pins in an arbitrary way. Thereby its neighbours in M become decided and so on. Step by step the whole component can be fixed. Note that the state of the critical nodes may change after each step.

The algorithm follows the description of the different cases above. The nodes are partitioned into different classes (critical, non-critical,...) and the edge directions are assigned by local inspections. The missing components at the end are handled as described above.

The correctness of the algorithm follows from the fact that it only determines a connection, if either the alternative cannot be better (non-critical nodes, decided nodes) or the alternatives have been proved to be equal (components of tie nodes).

The Algorithm

At each step of the algorithm we have to determine the new state of at most four critical nodes. This can be done in constant time. The first two phases (non-critical nodes and decided nodes) are trivially linear. The only difficulty is to compute the components M in phase 3. But this is easy, too: starting with an actual subset M' of V (at the beginning one tie node) we have to test the neighbours of nodes in M' if they are tie. Therefore we make at most $4|M|$ tests. All components are disjoint (follows from the definition of a component), therefore we have to make $O(n)$ tests.

Theorem. *The algorithm works in linear time.*

Remarks

The class of rectilinear upward drawings considered here is a proper subclass of the class commonly known as rectilinear upward drawings (each edge is represented by a monotonically increasing curve). If we consider the differences between the two models, we find examples with much more bends for our model. But it seems that these unnecessary bends can be removed very easily. With small modifications our algorithm is a good heuristic for the problem of general rectilinear upward drawing.

References

[1] Di Battista, G., R. Tamassia and I.G. Tollis, Area requirement and symmetry display in drawing graphs, *Discrete and Comp. Geometry* 7 (1992), pp. 381–401.

[2] Storer, J.A., On minimal node-cost planar embeddings, *Networks* 14 (1984), pp. 181–212.

[3] Tamassia, R., On embedding a graph in the grid with the minimum number of bends, *SIAM J. Comput.* 16 (1987), pp. 421-444.

[4] Tamassia, R., and I.G. Tollis, A Unified Approach to Visibility Representations of Planar Graphs, *Discrete and Comp. Geometry* 1 (1986), pp. 321–341.

[5] Tamassia, R., and I.G. Tollis, Efficient embedding of planar graphs in linear time, in: *Proc. IEEE Int. Symp. on Circuits and Systems*, Philadelphia, pp. 495–498, 1987.

Representations of Planar Graphs

C. Thomassen

Abstract not Available.

On Lattice Structures Induced by Orientations

Patrice Ossona de Mendez *

First we remark that an acyclic orientation of a rooted graph defines a distributive lattice on its algebraic cocircuits (i.e. oriented cuts). Let v_0 be the root of an acyclic oriented graph G, recall that an algebraic cocircuit of G may be expressed as a sum of vertex cocircuits with integer coefficients. If we constrain the v_0 cocyle's coefficient to be null, the decomposition is unique and defines an injection from the cocircuits into a $\mathbb{Z}$-free module. The total order of $\mathbb{Z}$ defines a partial order on the algebraic cocircuits of G. This partial order is a distributive lattice : the infimum and supremum operators are expressed in terms of min and max on the coefficients. The Hasse diagram of a lattice is the oriented graph whose vertices are the elements of the lattice and whose edges correspond to the immediate-sucessor relation. In the Hasse diagram of the cocircuits' lattice, two cocircuits are adjacent if and only if their distance is 1 in the $\mathbb{Z}$-module. By duality, the algebraic circuits of an oriented face-rooted planar graph have also a distributive lattice structure. This circuits' lattice seems to be more powerfull.

We shall first give an example where this circuits' lattice structure, defined on the e-bipolar orientations, leads to relevant geometric interpretations. Thatfor, we express a bijection between e-bipolar orientations and algebraic circuits of a planar graph [11] [10].

Given a biconnected graph G and an oriented edge $e = (s, t)$ of G, an e-bipolar orientation of G is an acyclic orientation of the edges of G having s as a unique source and t as a unique sink.

Bipolar orientations are closely related to connexity and planarity [5] [3] [10]. They have been extensively used to perform graph drawing of planar graphs [8] [14] [4] [6] [8], upward drawings [7] and testing graph planarity [9].

In the planar case, the angle graphe $\mathcal{A}(G)$ of G (also called radial graph) is defined as the vertex/face adjacency graph related to an embedding of G. The e-bipolar orientations of G are in bijection with the edge 2-colorations of $\mathcal{A}(G)$ satisfying local conditions. Given one such 2-coloration, the e-bipolar orientations are in bijection with the alternating cycles [12] [1] [5].

An edge 2-coloration of a bipartite graph naturally defines an orientation of the edges mapping alternating cycles into oriented circuits. The e-bipolar orientations of G are thus in bijection with the circuits of an induced orientation of $\mathcal{A}(G)$. If the graph is 3-connected, each edge of $\mathcal{A}(G)$ (except those incident to the vertices corresponding to vertices and faces of G incident to e) belongs to a circuit [5]. Hence, the graph $\mathcal{A}(G)$ is totaly cyclic and, by duality, we may apply the lattice construction defined previously : the set of the e-bipolar orientations of a 3-connected plane graph G is a distributive lattice and its Hasse diagram is the adjacency graph of the e-bipolar orientations of G (two orientations being adjacent if they differ by the orientation of a unique edge). The connexity of the lattice implies the connexity of the adjacency graph of the e-bipolar orientations of a 3-connected. This last result holds also for non planar 3-connected graphs, as proved in [5] [10]. The minimum and maximum e-bipolar orientations of the lattice have a very simple geometric interpretation exhibited by the left and the right path-packing algorithms [4].

The e-bipolar orientations of a 3-connected graph G are in bijection with its st-upward drawings. The Hasse diagram describes allowable local deformations of an upward drawing. The connexity implies that any two upward drawings can be derived from one another through successive local deformations.

*Ecole des Hautes Etudes en Sciences Sociales, 54 Boulevard Raspail – PARIS

As an other application, we mention the lattice structure of the 3-trees decompositions introduced by W. Schnyder and C. Thomassen. In this later case, the successor relation is related to a circular permutation of the tree assignments among a triangle. This local transformation generates the lattice and hence all the 3-trees decompositions [2].

References

[1] M. Bousset. *Orientation d'un schéma par passage d'un flot dans les angles.* PhD thesis, Ecole des Hautes Etudes en Sciences Sociales, Paris, 1993.

[2] H. de Fraysseix and P. O. de Mendez. On tree decompositions and angle marking of planar graphs. in preparation.

[3] H. de Fraysseix and P. O. de Mendez. Planarity and edge poset dimension. submitted to the European Journal of Combinatorics.

[4] H. de Fraysseix, P. O. de Mendez, and J. Pach. Representation of planar graphs by segments. to appear in Intuitive Geometry, 1993.

[5] H. de Fraysseix, P. O. de Mendez, and P. Rosenstiehl. Bipolar orientations revisited. In *Fifth Franco-Japanese Days on Combinatorics and Optimization*, 1992.

[6] H. de Fraysseix, P. O. de Mendez, and P. Rosenstiehl. On triangle contact graphs. In *Cambridge combinatorial Conference in honor of Paul Erdös*, 1993.

[7] G. Di Battista and R. Tamassia. Algorithms for plane representations of acyclic digraphs. *Theoret. Comput. Science*, 61:436–441, 1988.

[8] G. Kant. Hexagonal grid drawings. Internal report of the Utrecht University, 1992.

[9] A. Lempel, S. Even, and I. Cederbaum. An algorithm for planarity testing of graphs. In Gordon and Breach, editors, *Theory of Graphs*, pages 215–232, 1967.

[10] P. O. de Mendez. *Orientations bipolaires.* PhD thesis, Ecole des Hautes Etudes en Sciences Sociales, Paris, 1993. To be defended.

[11] P. O. de Mendez. The plane bipolar orientations' lattice. In *Sixth Franco-Japanese Days on Combinatorics and Optimization*, 1993.

[12] P. Rosenstiehl. Embedding in the plane with orientation constraints : the angle graph. *Ann. N.Y. Acad. Sci.*, pages 340–346, 1983.

[13] P. Rosentiehl and R.E. Tarjan. Rectilinear planar layout and bipolar orientation of planar graphs. *Discrete and Computational Geometry*, 1:343–353, 1986.

[14] R. Tamassia and I.G. Tollis. A unified approach to visibility representations of planar graphs. *Discrete Comput. Geom.*, 1:321–341, 1986.

Complexity of Intersection Classes of Graphs

Jan Kratochvíl * **and Jiří Matoušek** *

Many of the well known and studied classes of intersection graphs of geometrical objects in the plane possess nice properties, with respect to algorithmic complexity. Often such problems like stable set, clique or chromatic number are polynomially solvable, and many of such classes are recognizable in polynomial time (cf. interval graphs, circular arc graphs, circle graphs, function graphs, etc.). On the other hand, already slightly more general classes are NP-hard to recognize (e.g., intersection graphs of straight segments). Here we give a brief survey of the complexity of the recognition problem.

If $\mathcal{C}$ is a class of sets (geometric objects in our case), then the class of *intersection graphs of* $\mathcal{C}$, denoted by $IG(\mathcal{C})$, will be the class of all simple undirected graphs, isomorphic to graphs of the form $G = (V, E)$, where $V \subseteq \mathcal{C}$ and $e = uv \in E$ iff $u \cap v \neq \emptyset$. We call this V a *representation* of (the isomorphism class of) G.

We set (all the objects in consideration are planar)

$$\begin{aligned} STRING &= IG(\{\text{all simple curves}\}) \\ CONV &= IG(\{\text{all convex sets}\}) \\ SEG &= IG(\{\text{all straight line segments}\}) \\ k-DIR(d_1,\ldots,d_k) &= IG(\{\text{all segments with slopes among } d_1,\ldots,d_k\}), \\ &\qquad d_1,\ldots,d_k \text{ real numbers} \\ k-DIR &= \bigcup\{k-DIR(d_1,\ldots,d_k);\ d_1,\ldots,d_k \text{ real numbers}\}. \end{aligned}$$

(Note that in our setting, interval graphs = IG({segments of a line}), circular arc graphs = IG({segments of a circle}), circle graphs = IG({chords of a circle}) and all of these classes can be recognized in polynomial time).

Theorem 1 i) *Recognition of STRING graphs is NP-hard* [2], *no upper bound is known.* ii) *Recognition of CONV and SEG graphs is NP-hard* [1, 2] *and both these problems are in PSPACE* [5].
iii) *For every fixed* $k \geq 2$, *recognition of k-DIR graphs is NP-complete* [4, 5].
iv) *Recognition of* $k-DIR(d_1,\ldots,d_k)$ *graphs is NP-complete.* [5] *(note that here k and* $d_1,\ldots,d_k$ *are part of the input).*
v) *Intersection graphs of isothetic rectangles (i.e., graphs of boxicity 2) are NP-complete to recognize* [4].
vi) *Recognition of SEG graphs is polynomially equivalent to deciding solvability of a system of strict polynomial inequalities in real numbers* [5].

Unlike most decesion problems of similar nature, the recognition problems treated in Theorem 1 are not known to be in NP (cv. parts i),ii)), or their membership in NP is nontrivial (parts iii),iv)). The following results show why the usual 'guess and check' scheme of proving NP–membership fails for STRING, SEG and k-DIR graphs.

*Prague, Czech Republic.

Theorem 2 i) *For every n, there is a graph $G_n \in STRING$ on $O(n^4)$ vertices, such that in each of its STRING-representations there are two curves which share at least 2^n crossing points* [3].

ii) *For every n, there is a graph $G_n \in SEG$ on $O(n^2)$ vertices, such that in each of its SEG-representations whose segments have integer endpoints, there is segment with an endpoint coordinate of size at least 2^{2^n}* [5].

iii) *For every n, there is a graph $G_n \in 3-DIR$ on $O(n^2)$ vertices, such that in each of its 3-DIR-representations whose segments have integer endpoints, there is a segment with and endpoint coordinate at least 2^n* [5].

References

[1] J.Kratochvíl, J. Matoušek: *NP-hardness results for intersection graphs* Comment. Math. Univ. Carolin. 30 (1989), 761-773 (MR 91c:05156)

[2] J.Kratochvíl: *String graphs II. Recognizing string graphs is NP-hard*, J. Combin. Theory Ser. B 52 (1991), 67-78 (MR 92g:05157)

[3] J.Kratochvíl, J. Matoušek: *String graphs requiring exponential representations*, J. Combin. Theory Ser. B 53 (1991), 1-4 (MR 92g:05080)

[4] J.Kratochvíl: *A special planar satisfiability problem and some consequences of its NP-completeness*, Discrete Appl. Math. (to appear)

[5] J.Kratochvíl, J. Matoušek: *Intersection graphs of segments* , J. Combin. Th. Ser. B (to appear)

On Triangle Contact Graphs*

Hubert de Fraysseix, † Patrice Ossona de Mendez † and Pierre Rosenstiehl †

Various representations of the same planar map are described below. Vertices, edges and faces are alternatively represented by points, arcs or disks of the plane. By representation of a planar map, it is understood that the circular order of the edges around each vertex is preserved.

An old problem of geometry consists of representing a planar map M by a collection of disks matched with the vertices of M. These disks are disjoint except at contact points for some pairs of them, the contacts representing the edges of M. The case of unconstrained disks is merely solved by drawing at each vertex point v a closed curve surrounding v and half of the edges arcs incident to v, in a tubular way. The difficulty starts when the disks have to respect a prescribed shape. The famous case of circular disks, known as the Andrew-Thurston circle packing Theorem [1], hits difficulties of numerical analysis; it has been improved to polynomial complexity and generalized recently [7].

We are concerned here with triangular disks, such that each contact point is a vertex of one triangle and belongs to the side of another one. This asymetry induces an orientation of the contact, that is an orientation of the corresponding edge. Such an arrangement is called a

*This work was partially supported by the ESPRIT Basic Research Action Nr. 7141 (ALCOM II).

†CNRS, EHESS, 54 Boulevard Raspail, 75006, Paris, France.

triangle contact system. So, it is obvious that to any triangle contact system S is associated an oriented planar map $M(S)$. We assume below that the orientation of the contact edge is towards the triangle which gives a vertex in the contact.

A first result is that any planar map may be represented by a triangle contact system. The result still holds if we impose a coloration to the triangles and the contacts in the following way : the edges taken in clockwise order are colored respectively a, b and c and each vertex is colored as its opposite side; now a contact has to be between a vertex and a side of the same color. An instance of this orientation constraint consists of forcing the triangles to have a common oriented angle in the affine plane.

In the case of maximal planar maps M the coloration of the triangles is intrinsicaly imposed. We introduce a coloration that partitions the edges not incident to the infinite face in three oriented trees rooted on the frame, which are nothing else but the Schnyder trees [9] of a maximal planar map. A 3-trees decomposition of Schnyder, or a Schnyder orientation, is defined as an orientation of the edges besides the infinite face and a coloration of them in three colors a, b and c such that, at each vertex besides those of the infinite face, there is exactly three incoming edges, colored respectively a, b and c in clockwise order and such that the outgoing edges of one color are grouped in the angle formed by the two incoming edges of the other colors. By definition, each color defines a tree rooted on the infinite face.

The construction of a colored triangle contact system representing a planar map is achieved in linear time and space (considering that arithmetic is performed in constant time over the rational field).

A variation of our main result is that any maximal planar map may be represented by an isosceles triangle contact system, each having an horizontal basis and the opposite vertex placed below.

Let a T-shape in the (Ox, Oy) plane be a pair of an horizontal segment and a vertical one placed below with a contact point. A **T-contact system** is a collection of T's, all disjoints except for contact points, consisting of an extreme point from one exactly and a side point from the other. We show how a colored triangle contact system is transformed into a T-contact system which provides very compact representations of planar graphs.

A **rectilinear representation** of a planar map represents each vertex by an horizontal segment and each edge by a vertical segment incident to two horizontal vertex segment [8] [10]. In [8], each face of M is represented by a disk with a lowest vertex and a highest vertex segment, a (Oy)-monotone left boundary and a (Oy)-monotone right boundary, this one being a straight segment. Therefore, by extending each vertex segment on its right, up to the first met vertical segment, the representation of M becomes a partition of a rectangle into rectangles : each vertex is represented by an horizontal segment, each face by a vertical segment, and each edge by a rectangular disk. Such a representation is called a tessalation representation of M. Any 2-connected planar map has a tessalation representation, as also explained by Tamassia and Tollis [10].

Actually, all the tessalation representations of a 2-connected graph are obtained by using the bipartite planar graph representation of the radial graph by contact segments as defined in [3] [4].

We get here a tessalation representation of a maximal planar map from an interesting feature of the triangle contact representation of a maximal planar map M, that is the representation of the faces of M by triangles. In case each vertex triangle is isosceles with its basis parallel to (Ox) and the opposite vertex placed below, each face is a triangle with each basis parallel to (Ox) and the opposite vertex placed above. By drawing a vertical height segment of a proper length at each triangular face, and extending the horizontal segments of the vertex bases we obtain a tessalation representation of M.

References

[1] E.M. Andreev. On convex polyhedra in lobacevskii spaces. *Mat. Sb.*, 81:445–478, 1970.

[2] H. de Fraysseix and P. O. de Mendez. On tree decompositions and angle marking of planar graphs. to appear.

[3] H. de Fraysseix, P. O. de Mendez, and J. Pach. A streamlined depth-first search algorithm revisited. to appear.

[4] H. de Fraysseix, P. O. de Mendez, and J. Pach. Representation of planar graphs by segments. to appear in Intuitive Geometry, 1993.

[5] H. de Fraysseix, J. Pach, and R. Pollack. Small sets supporting Fary embeddings of planar graphs. *Combinatorica*, 10:41–51, 1990. Stoc Conference 1988.

[6] P. Eades and R. Tamassia. Algorithms for drawing planar graphs: an annotated bibliography. Tech. Rep. No. CS-89-09, Brown University, 1989.

[7] B. Mohar. Circle packings of maps in polynomial time. to appear, 1992.

[8] P. Rosentiehl and R.E. Tarjan. Rectilinear planar layout and bipolar orientation of planar graphs. *Discrete and Computational Geometry*, 1:343–353, 1986.

[9] W. Schnyder. Planar graphs and poset dimension. *Order*, 5:323–343, 1989.

[10] R. Tamassia and I.G. Tollis. Tessalation representation of planar graphs. In *Proc. Twenty-Seventh Annual Allerton Conference on Communication, Control, and Computing*, pages 48–57, 1989.

Characterisation and Construction of the Rectangular Dual of a Graph

Simone Pimont * **and Michel Terrenoire** †

Introduction

Circuit placement consists in positionning rectangular blocks, defined by their area and interconnected by nets, on a rectangular surface to be minimized. An available strategy deals with the rectangular dualization of a connectivity graph. We introduce the notion of polarity index, then the concept of polarization, in order to characterize graphs with rectangular duals, and to build an associated rectangular representation.

The three steps of placement

The minimization objective is approached by the construction of an initial placement (step 1), such as the sum of the lengths of the connections is minimum. This initial placement is obtained by means of a data analysis method (a factorial analysis on a distance matrix). It defines in R^2 a set X of points corresponding to the blocks. From this placement, we define a connectivity graph G(X) that resumes the proximity relations between the corresponding points. Then, by dualization, a rectangular representation associated to G(X) is researched (step 2); that is to say a dissection of a rectangle into rectangles for which G(X) describes the adjacency relations among the rectangles. In step 3, an iterative construction of rectangular placements is processed, whose objective is area minimization [FOU 89]. The related work focuses on step 2, particularly on the characterization and on the construction of a rectangular graph.

A dualization process to construct a rectangular graph

We know that a necessary condition for a planar graph to admit a rectangular dual is that it is triangular (every face, except the outer one, is a triangle) [KOZ 85, BHA 88]. So, we consider the DELAUNAY graphe D(X) associated to the set X. However the above condition is not sufficient. A triangular graph admits a rectangular dual if an only if it does not contain complex triangle [KOZ 85]. Moreover, from a triangular graph, one can built more than one rectangular dual : an edge between two nodes in the graph can correspond to a vertical or horizontal boundary in the rectangular dual. To take into account this phenomenon, we introduce **indices of polarity** for the edges [PIM 93]. The indices can take two values : horizontal or vertical. We define an heuristic that assigns a polarity index to each edge of D(X) in order to construct a rectangular dual. The elimination of the complex triangles is attempted at the same time. A program was realized in PASCAL on PC, and in C on APOLLO. But the above polarity does not account by itself of the rectangularization. So we introduce another concept that deepens this notion.

*Laboratoire d'Ingenierie des Systemes d'Information. pimont@lisisun.univ-lyon1.fr

†Laboratoire des methodes et analyses des systemes et des structures, Universite Claude Bernard LYON 1, 43 Boulevard du 11 Novembre 1918, F-69622 Villeurbanne Cedex (FRANCE).

Characterization of a dualizable graph : Polarizable graph

We consider a triangular graph H=(X,W). It admits a 4-completion graph Hc=(Xc,Wc), introducing the four external vertices : North, East, South, West (see [KOZ 85] for the definition of 4-completion).

- **Notations :**
  - Z is the set Wc without the four edges [N,O] [O,S] [S,E] [E,N]
  - Π is a mapping of Z into the set {v,h} ("v" for vertical and "h" for horizontal)
  - δ is a mapping (named orientation) of Z into Xc : δ([x,y]) is the origin of [x,y]
  - Hc^{δ} is the oriented graph deduced from the graph (Xc,Z) by δ
  - a path p in Hc^{δ} is a v-path (resp. h-path) if Π(w) = v (resp. h) for every edge w corresponding to an arc of p
  - N_x, O_x, S_x, E_x are the set of the v-paths from N to x, the set of the h-paths from O to x, the set of the v-paths from x to S, the set of the h-paths from x to E

- We suppose that for every x in X, (Π,δ) satisfies :
  - $\forall$ [N,x] $\in$ Z, δ([N,x]) = N and Π([N,x]) = v
  - $\forall$ [O,x] $\in$ Z, δ([O,x]) = O and Π([O,x]) = h
  - $\forall$ [S,x] $\in$ Z, δ([S,x]) = x and Π([S,x]) = v
  - $\forall$ [E,x] $\in$ Z, δ([E,x]) = x and Π([E,x]) = h

- (Π,δ) is a **polarization for Hc** if for every x in X
  - N_x, O_x, S_x, E_x are not empty
  - $\forall$ n $\in N_x$, $\forall$ o $\in O_x$, $\forall$ s $\in S_x$, $\forall$ e $\in E_x$, the four sets : n $\cap$ o, n $\cap$ s, n $\cap$ e, o $\cap$ s, o $\cap$ e, s $\cap$ e are reduced to x

Then we can define :

- A graph H is **polarizable** if it is triangular and if it admits a 4-completion graph Hc with an associated polarization.
- For each node x, we define a **local polarization for x**, based on *local properties* about the existence and the clockwize order of the oriented edges admitting x as endpoints.

We state two theorems [PIM 93] :

- **Theorem 1 :**
  A triangular graph admits a rectangular dual if an only if it is polarizable.
- **Theorem 2 :**
  For a triangular graph H with an associated triangular 4-completion Hc, (Π,δ) is a polarization if and only if (Π,δ) is a local polarization for each vertex in Hc.

Conclusion

The concept of polarity index allows us to realize a program rather efficient over our examples; but this approach is heuristic. The concept of polarizable graph, in view of the theorems 1 and 2, will enable the development of more efficient algorithms for circuit placement. For example, we propose to choose a couple candidate for polarization, and then to modify it using the local properties associated to the node admissible polarization. Our theoretical and practical results allow us to argue that the polarization and the associated properties give a pertinent framework to elaborate efficient dualization algorithms.

References

[BHA 88] J. Bhasker and S. Sahni, "A Linear Algorithm to Find a Rectangular Dual of a Planar Triangulated Graph", Algorithmica, vol.3(2), 1988, pp. 247-278.

[FOU 89] D.Fourre and S. Pimont, "Heuristique de rectangularisation de graphe pour le placement de circuits", rapport de recherche, Laboratoire Informatique de l'Universite de Lyon 1, 1989.

[KOZ 85] K. Kozminshi and E. Kinnen, "Rectangular duals of planar graph", Networks, vol.15(2), 1985, pp. 145-157.

[PIM 93] S. Pimont and M. Terrenoire, "Dual rectangulaire d'un graphe pour le placement de circuit", TSI, vol.12(2), 1993, pp. 193-216.

Two Algorithms for Finding Rectangular Duals of Planar Graphs

Goos Kant* **Xin He†**

The problem of drawing a graph on the plane has received increasing attention due to a large number of applications [3]. Examples include VLSI layout, algorithm animation, visual languages and CASE tools. Vertices are usually represented by points and edges by curves. In the design of floor planning of electronic chips and in architectural design, it is also common to represent a graph G by a *rectangular dual*, defined as follows. A *rectangular subdivision system* of a rectangle R is a partition of R into a set $\Gamma = \{R_1, R_2, \ldots, R_n\}$ of non-overlapping rectangles such that no four rectangles in Γ meet at the same point. A *rectangular dual* of a planar graph $G = (V, E)$ is a rectangular subdivision system Γ and a one-to-one correspondence $f : V \rightarrow \Gamma$ such that two vertices u and v are adjacent in G if and only if their corresponding rectangles $f(u)$ and $f(v)$ share a common boundary. In the application of this representation, the vertices of G represent circuit modules and the edges represent module adjacencies. A rectangular dual provides a placement of the circuit modules that preserves the required adjacencies.

This problem was studied in [1, 2, KK89]. Bhasker and Sahni gave a linear time algorithm to construct rectangular duals [2]. The algorithm is fairly complicated and requires many intriguing procedures. The coordinates of the rectangular dual constructed by it are real numbers and bear no meaningful relationship with the structure of the graph. This algorithm consists of two major steps: (1) constructing a so-called *regular edge labeling* (REL) of G; and (2) constructing the rectangular dual using this labeling. A simplification of step (2) is given in [5]. The coordinates of the rectangular dual constructed by the algorithm in [5] are integers and carry clear combinatorial meaning. However, the step (1) still relies on the complicated algorithm in [2]. (A parallel implementation of this algorithm, working in $O(\log n \log^* n)$ time with $O(n)$ processors, is given in [6].)

*Department of Computer Science Utrecht University Padualaan 14, 3584 CH Utrecht the Netherlands. goos@cs.ruu.nl. Research supported by the ESPRIT Basic Research Actions program of the EC under contract No. 7141 (project ALCOM II).

†Department of Computer Science State University of New York at Buffalo Buffalo, NY 14260 United States of America. xinhe@cs.buffalo.edu. Research supported by National Science Foundation, grant number CCR-9011214.

In this paper we present two linear time algorithms for finding a regular edge labeling. The two algorithms use totally different approaches and both are of independent interests. The first algorithm is based on the *edge contraction* technique, which was also used for drawing triangular planar graphs on a grid [10]. The second algorithm is based on the *canonical ordering* for 4-connected planar triangular graphs. This technique extends the canonical ordering, which was defined for triangular planar graphs [4] and triconnected planar graphs [5], to this class of graphs. Another interesting representation of planar graphs is the *visibility representation*, which maps vertices into horizotnal segments and edges into vertical segments [7, 11]. It turns out that the canonical ordering also gives a reduction of a factor 2 in the width of the visbility representation of 4-connected planar graphs.

References

[1] Bhasker, J., and S. Sahni, A linear algorithm to check for the existence of a rectangular dual of a planar triangulated graph, Networks 7 (1987), pp. 307-317.

[2] Bhasker, J., and S. Sahni, A linear algorithm to find a rectangular dual of a planar triangulated graph, *Algorithmica* 3 (1988), pp. 247–178.

[3] Di Battista, G., P. Eades, R. Tamassia and I.G. Tollis, *Algorithms for Automatic Graph Drawing: An Annotated Bibliography*, Dept. of Comp. Science, Brown Univ., Technical Report, 1993.

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

[5] He, X., On finding the rectangular duals of planar triangulated graphs, *SIAM J. Comput.*, to appear.

[6] He, X., *Efficient Parallel Algorithms for two Graph Layout Problems*, Technical Report 91-05, Dept. of Comp. Science, State Univ. of New York at Buffalo, 1991.

[7] Kant, G., Drawing planar graphs using the *lmc*-ordering, *Proc. 33th Ann. IEEE Symp. on Found. of Comp. Science*, Pittsburgh, 1992, pp. 101-110.

[8] Koźmiński, K., and E. Kinnen, Rectangular dual of planar graphs, *Network* 5 (1985), pp. 145–157.

[9] Rosenstiehl, P., and R. E. Tarjan, Rectilinear planar layouts and bipolar orientations of planar graphs, *Discr. and Comp. Geometry* 1 (1986), pp. 343–353.

[10] Schnyder, W., Embedding planar graphs on the grid, in: *Proc. 1st Annual ACM-SIAM Symp. on Discr. Alg.*, San Fransisco, 1990, pp. 138–147.

[11] Tamassia, R., and I. G. Tollis, A unified approach to visibility representations of planar graphs, *Discr. and Comp. Geometry* 1 (1986), pp. 321–341.

A More Compact Visibility Representation*

Goos Kant†

Introduction

The problem of "nicely" drawing a graph in the plane has received increasing attention due to the large number of applications [1]. Examples include VLSI layout, algorithm animation, visual languages and CASE tools. Several criteria to obtain a high aesthetic quality have been established. Typically, vertices are represented by distinct points in a line or plane, and are sometimes restricted to be grid points. (Alternatively, vertices are sometimes represented by line segments.) Edges are often constrained to be drawn as straight lines or as a contiguous set of line segments (e.g., when bends are allowed). The objective is to find a layout for a graph that optimizes some cost function, such as area, minimum angle, number of bends, or satisfies some other constraint (see [1] for an up to date overview).

One of the most beautiful ways for drawing G is by using a *visibility representation*. In a visibility representation every vertex is mapped to a horizontal segment, and every edge is mapped to a vertical line, only touching the two vertex segments of its endpoints. It is clear that this leads to a nice and readable picture, and it therefore gains a lot of interest. It has been applied in several industrial applications, for representing electrical diagrams and schemas (Rosenstiehl, personal communication). Otten & Van Wijk [9] showed that every planar graph admits such a representation, and a linear time algorithm for constructing it is given by Rosenstiehl & Tarjan [7] (independently, Tamassia & Tollis [11] came up with the same algorithm). The size of the required grid is $(2n-5)\times(n-1)$, with n the number of vertices. The algorithm is based on a so called *st-numbering*: a numbering $v_1,\ldots,v_n$ of the vertices such that $(v_1,v_n)\in G$ and every vertex v_i $(1<i<n)$ has neighbors v_j and v_k with $j<i<k$. The *height* of the drawing is the longest path from v_1 to v_n, which has length at most $n-1$. The *width* of the drawing is the longest path in the dual graph, which is $f-1$, where f is the number of faces in G (by Euler's formula: $m\leq 3n-6$ and $f=m-n+2$).

The algorithm is used in several drawing algorithms. We mention here the algorithm of Tamassia & Tollis [12] for constructing an orthogonal drawing, and the work of Di Battista, Tamassia & Tollis [3] for computing *constrained* visibility representations. Rosenstiehl & Tarjan also discuss the open problems concerning the grid size of visibility representations [7]. The requirement of using a small area seems to become a core area in the research field of graph drawing, due to the important applications in VLSI-design and chip layout (e.g., see Kant [5]).

In this paper we show that every planar graph can be represented by a visibility representation on a grid of size at most $(\lfloor\frac{3}{2}n\rfloor-3)\times(n-1)$. This improves all previous bounds considerably. An outline of the algorithm to achieve this is as follows. Assume the input graph G is triangulated (otherwise a simple linear time algorithm can be applied to make it so [7]). Then we split G into its 4-connected components, and construct the 4-block tree of G. We show that we can do this in linear time for triangulated planar graphs, thereby improving the $O(n\cdot\alpha(m,n)+m)$ time algorithm of Kanevsky et al. [4] for this special case. To each 4-connected component the algorithm of Kant & He is applied, who showed that if the planar graph is 4-connected, then a

*This work was supported by ESPRIT Basic Research Action No. 7141 (project ALCOM II: *Algorithms and Complexity*). Part of this work was done while visiting the Graph Theory workshop at the Bellairs Research Institute of McGill University (Montreal), Feb. 12-19, 1993.

†Dept. of Computer Science, Utrecht University, Padualaan 14, 3584 CH Utrecht, the Netherlands. goos@cs.ruu.nl

visibility representation of it can be constructed with grid size at most $(n-1) \times (n-1)$ [8]. The representations of the 4-connected components are combined into one entire drawing, leading to the desired width.

References

[1] Chiba, N., and T. Nishizeki, Arboricity and subgraph listing algorithms, *SIAM J. Comput.* 14 (1985), pp. 210–223.

[2] Di Battista, G., Eades, P., and R. Tamassia, *Algorithms for Automatic Graph Drawing: An Annotated Bibliography,* Dept. of Comp. Science, Brown Univ., Technical Report, 1993.

[3] Di Battista, G., R. Tamassia and I.G. Tollis, Constrained visibility representations of graphs, *Inform. Process. Letters* 41 (1992), pp. 1–7.

[4] Kanevsky, A., R. Tamassia, G. Di Battista and J. Chen, On-line maintenance of the four-connected components of a graph, in: *Proc. 32th Annual IEEE Symp. on Found. of Comp. Science*, Puerto Rico, 1991, pp. 793–801.

[5] Kant, G., Drawing planar graphs using the *lmc*-ordering, *Proc. 33th Ann. IEEE Symp. on Found. of Comp. Science*, Pittsburgh, 1992, pp. 101–110.

Revised and extended version in:

[6] Kant, G., *Algorithms for Drawing Planar Graphs*, PhD thesis, Dept. of Computer Science, Utrecht University, 1993.

[7] Kant, G., On triangulating planar graphs, submitted to *Information and Computation*, 1993.

[8] Kant, G., and X. He, Two Algorithms for Finding Rectangular Duals of Planar Graphs, Tech. Report RUU-CS-92-41, Dept. of Computer Science, Utrecht University, 1992.

[9] Otten, R.H.J.M., and J.G. van Wijk, Graph representation in interactive layout design, in: *Proc. IEEE Int. Symp. on Circuits and Systems*, 1978, pp. 914–918.

[10] Rosenstiehl, P., and R.E. Tarjan, Rectilinear planar layouts and bipolar orientations of planar graphs, *Discr. and Comp. Geometry* 1 (1986), pp. 343–353.

[11] Tamassia, R., and I.G. Tollis, A unified approach to visibility representations of planar graphs, *Discr. and Comp. Geometry* 1 (1986), pp. 321–341.

[12] Tamassia, R., and I.G. Tollis, Planar grid embedding in linear time, *IEEE Trans. Circuits and Systems* 36 (1989), pp. 1230–1234.

Cone Visibility Graphs

Anna Lubiw*

Introduction

Two main ways of representing graphs are as intersection graphs and as visibility graphs using the symmetric relations of intersection and visibility respectively. See [G] for a survey on intersection graphs, and [O'R] for a survey on visibility graphs. A main difference between the two is that visibility usually carries with it the notion of "blocking"; consequently the classes of graphs are usually not closed under induced subgraphs, and are more difficult to characterize.

This work explores a class of directed graphs defined by means of an asymmetric relationship which has some of the features of intersection and some of the features of visibility. Let P be a set of points in the plane. With each point p let there be an ordered pair of rays $r_1(p), r_2(p)$ emanating from the point, determining the closed clockwise cone $C(p)$ going from $r_1(p)$ to $r_2(p)$. The *cone visibility digraph* of this configuration is a directed graph on vertex set P, with an edge from vertex p_1 to vertex p_2 iff p_2 is contained in the cone $C(p_1)$. A digraph is a *cone visibility digraph* if it has such a representation (for some choice of points and cones).

These are "visibility" graphs in the sense that the cone at a point determines which other points it can "see". There are two differences between this notion and previous definitions of visibility graphs: one is that the relationship is asymmetric so the graphs are directed; another is that there is no notion of "blocking"—whether one point sees a second point is independent of the other points. There are two significant consequences of this: One is that cone visibility graphs are closed under induced subgraphs—thus there might be be a forbidden subgraph characterization of the class. Another consequence is that given a representation of a cone visibility graph, there is an $O(1)$ test for whether two points are joined by an edge—thus a cone visibility representation of a graph is a very efficient representation.

Relationship to visibility graphs of polygons

*Un*directed cone visibility graphs can be obtained by symmetrizing the relationship: put an undirected edge between p_1 and p_2 iff p_1 is in the cone $C(p_2)$ and p_2 is in the cone $C(p_1)$. These graphs have enough relation to visibility graphs of polygons that an understanding of them may help us understand visibility graphs of polygons. Given a simple polygon in the plane, the vertices and edges determine a cone system in the obvious way. The undirected cone visibility graph contains all the edges of the visibility graph of the polygon, plus edges joining vertices whose visibility in the polygon is blocked by an edge not incident with either vertex.

Relationship to dimension of posets

Whenever the rays $r_1(p)$ are all parallel (for all points p), and the rays $r_2(p)$ are all parallel, and the angle between them is no more than 180°, then the resulting cone visibility digraph is a poset. See [T] for information on posets. In particular, when all the rays $r_1(p)$ are parallel to the positive y axis, and all the rays $r_2(p)$ are parallel to the positive x axis, the realizable cone visibility digraphs are exactly the posets of dimension 2. A poset has *dimension* n if it

*Research supported by NSERC. Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, N2L 3G1. alubiw@uwaterloo.ca

can be realized in n-dimensional space as the visibility digraph of cones that are translates of the positive quadrant, but cannot be so realized in a lower-dimensional space. The bipartite graph S_n with vertices $v_1, \ldots, v_n$ and $u_1, \ldots, u_n$ and an edge from v_i to u_j iff $i \neq j$ is a standard example of a poset of dimension n. Though S_n, for $n > 2$, does not have dimension 2, it is a cone visibility digraph (in the plane).

Positive results

Among the undirected graphs that are cone visibility graphs are trees, cycles, complete graphs, complete bipartite graphs, and bipartite permutation graphs. These results are not difficult and are proved by giving constructions. Turning to digraphs, it is not difficult to show that trees and cycles with arbitrary directions on the edges are cone visibility digraphs.

Negative results

This section is joint work with Jonathan F. Buss. One example of a graph that is not a cone visibility digraph is the bipartite graph with vertex set $V_1 \cup V_2$ where $|V_1| = 11$ and V_2 has a vertex corresponding to each of the $\binom{11}{3}$ triples of V_1 with edges to those three vertices of V_1. To prove that this graph is not a cone visibility digraph it suffices to prove that it is not possible to arrange 11 points in the plane so that every triple can be separated from the remaining points by a cone. We cannot have 6 points forming a convex 6-gon, otherwise the three odd points around the 6-gon cannot be separated. We also cannot have three or more convex hulls in the onion peeling of the point set, otherwise three points of the middle convex hull cannot be separated. But with 11 or more points, we cannot avoid both these situations.

Open questions

Characterize cone visibility digraphs and/or find a good recognition algorithm.

References

[G] M.C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, 1980.

[O'R] J. O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, 1987.

[T] W.T. Trotter. *Combinatorics and Partially Ordered Sets: Dimension Theory*. The Johns Hopkins University Press, 1992.

Circle Packing Representations in Polynomial Time

Bojan Mohar*

The Andreev-Koebe-Thurston [1, 2, 3, 5] circle packing theorem is generalized and improved in two ways. Simultaneous circle packing representations of the map and its dual map are obtained such that any two edges dual to each other cross at the right angle. The necessary and sufficient condition for a map to have such a primal-dual circle packing representation is that its universal cover graph is 3-connected. A polynomial time algorithm is obtained that given such a map M and a rational number $\varepsilon > 0$ finds an ε-approximation for the primal-dual circle packing representation of M. In particular, there is a polynomial time algorithm that produces simultaneous geodesic line convex drawings of a given map and its dual in a surface with constant curvature, so that only edges dual to each other cross.

In combination with graph embedding algorithms [4], these results give rise to efficient algorithms for drawing graphs on general surfaces.

References

[1] E. M. Andreev, On convex polyhedra in Lobačevskiĭ spaces, Mat. Sb. (N. S.) 81 (1970) 445–478; Engl. transl. in Math. USSR Sb. 10 (1970) 413–440.

[2] E. M. Andreev, On convex polyhedra of finite volume in Lobačevskiĭ space, Mat. Sb. (N. S.) 83 (1970) 256–260; Engl. transl. in Math. USSR Sb. 12 (1970) 255–259.

[3] P. Koebe, Kontaktprobleme auf der konformen Abbildung, Ber. Verh. Saechs. Akad. Wiss. Leipzig, Math.-Phys. Kl. 88 (1936) 141–164.

[4] B. Mohar, Generalizing Kuratowski Theorem, this workshop.

[5] W. P. Thurston, The geometry and topology of 3-manifolds, Princeton Univ. Lect. Notes, Princeton, NJ.

Obstructions for Embedding Extension Problems

Bojan Mohar,* Martin Juvan* and Jože Marinček *

Kuratowski's theorem gives rise to linear time algorithms which for a given graph determine whether the graph is planar (Hopcroft and Tarjan [7], Booth and Lueker [2], Fraysseix and Rosenstiehl [FR91]). The extensions of original algorithms also produce an embedding [4], or find a Kuratowski subgraph [24].

The result of Kuratowski has been generalized to non-orientable surfaces by Archdeacon and Huneke [2] and in a much more general setting to arbitrary surfaces by Robertson and Seymour [18]. They proved that for every fixed surface there is a finite set of obstructions for embeddability of graphs in this surface.

*University of Ljubljana, Slovenia

*University of Ljubljana

Although the genus problem is NP-hard [6], for every fixed surface there is a polynomial time algorithm which checks if a given graph can be embedded in the surface (Filotti et al. [5], Robertson and Seymour [18, 22], $O(n^3)$ algorithm using graph minors, $O(n^2 \log n)$ improvement by B. Reed [19, 20, 21]). A constructive version in [1] has running time $O(n^{10})$.

Let $K \subseteq G$, and suppose that we are given a 2–cell embedding of K into a (closed) surface Σ. The *embedding extension problem* asks whether it is possible to extend the given embedding of K to an embedding of G. In our talk we will briefly report on our results concerning the embedding extension problem. We are able to determine in linear time if the given 2-cell embedding of K can be extended to G. In case of a positive answer, such an extension is exhibited. Otherwise, an obstruction is given. Obstructions are not always small but they can be completely characterized. This enables us to get:

(a) a relatively short proof (in total approx. 100 pages) of the Kuratowski's Theorem for general surfaces, and

(b) a linear time algorithm that for every fixed surface S solves the embeddability problem in S. In case of the positive answer, the algorithm also exhibits an embedding, in case of the negative answer, we get a small obstruction — a subgraph which can not be embedded in S and whose number of branches is small.

In [13], a linear time algorithm for embedding graphs in the projective plane is given.

[14] presents the solution to the embedding extension problems in a disk or a cylinder. In [15], Möbius band and the projective plane obstructions are characterized and linear time algorithms for their discovery are presented.

The essential work is [16] Let B be a bridge of K in G. It is shown that B contains a nice (small up to a small number of almost disjoint millipedes) subgraph $\tilde{B}$ such that if K is 2-cell embedded in some surface and F is a face of K, then $\tilde{B}$ admits exactly the same types of embeddings in F as B. Moreover, such a universal obstruction $\tilde{B}$ can be constructed in linear time.

In [12] a special case of the embedding extension problem is solved when we are restricted for every bridge of K in G to have at most two essentially different embeddings.

The paper [10] contains a linear time algorithm for embedding graphs in the torus. It is supported by two other works [8, 9]. This is the other essential step of our work.

In [17], the generalization of the Kuratowski Theorem is proved.

Finally, [11] presents a linear time algorithm for embedding graphs in a general (fixed) surface.

References

[1] D. Archdeacon, The complexity of the graph embedding problem, in "Topics in Combinatorics and Graph Theory," R. Bodendiek and R. Henn (ed.), Physica-Verlag, Heidelberg, 1990, pp. 59–64.

[2] D. Archdeacon, P. Huneke, A Kuratowski theorem for non-orientable surfaces, J. Combin. Theory, Ser. B 46 (1989) 173–231.

[3] K. S. Booth, G. S. Lueker, Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-trees, J. Comput. System Sci. 13 (1976) 335–379.

[4] N. Chiba, T. Nishizeki, S. Abe, T. Ozawa, A linear algorithm for embedding planar graphs using PQ-trees, J. Comput. System Sci. 30 (1985) 54–76.

[5] I. S. Filotti, G. L. Miller, J. Reif, On determining the genus of a graph in $O(v^{O(g)})$ steps, in "Proc. 11th Ann. ACM STOC," Atlanta, Georgia (1979) pp. 27–37.

[6] H. de Fraysseix, P. Rosenstiehl, A depth-first search characterization of planarity, Ann. Discr. Math. 13 (1982) 75–80.

[7] J. E. Hopcroft, R. E. Tarjan, Efficient planarity testing, J. ACM 21 (1974) 549–568.

[8] M. Juvan, J. Marinček, B. Mohar, Elimination of local bridges, submitted.

[9] M. Juvan, J. Marinček, B. Mohar, Corner obstructions, submitted.

[10] M. Juvan, J. Marinček, B. Mohar, Embedding graphs in the torus, in preparation.

[11] M. Juvan, J. Marinček, B. Mohar, Efficient algorithm for embedding graphs in arbitrary surfaces, in preparation.

[12] M. Juvan, B. Mohar, A linear time algorithm for the 2-restricted embedding extension problem, in preparation.

[13] B. Mohar, Projective planarity in linear time, J. Algor., in press.

[14] B. Mohar, Obstructions for the disk and the cylinder embedding extension problem, submitted.

[15] B. Mohar, Obstructions for the Möbius band embedding extension problem, submitted.

[16] B. Mohar, Universal obstructions for embedding extension problems, submitted.

[17] B. Mohar, Generalizing Kuratowski's Theorem, in preparation.

[18] N. Robertson, P. D. Seymour, Graph minors. VIII. A Kuratowski theorem for general surfaces, J. Combin. Theory, Ser. B 48 (1990) 255–288.

[19] N. Robertson, P. D. Seymour, Graph minors. XIII. The disjoint paths problem, preprint, last revision of February 1992.

[20] N. Robertson, P. D. Seymour, Graph minors. XXI. Graphs with unique linkages, preprint, 1992.

[21] N. Robertson, P. D. Seymour, Graph minors. XXII. Irrelevant vertices in linkage problems, preprint, 1992.

[22] N. Robertson, P. D. Seymour, An outline of a disjoint paths algorithm, in: "Paths, Flows, and VLSI-Layout", Springer, 1990, pp. 267–292.

[23] C. Thomassen, The graph genus problem is NP-complete, J. Algorithms 10 (1989) 568–576.

[24] S. G. Williamson, Depth-first search and Kuratowski subgraphs, J. ACM 31 (1984) 681–693.

Automorphisms and Genus on Generalised Maps

Antoine Bergey *

Various combinatorial data structures were suggested to encode topological maps (2-cell embedding of a connected graph in a compact, connected, orientable surface without boundary). One of them is *combinatorial map* (or simply *2-maps*), wich are triples $C = (B, \alpha, \sigma)$ such that α and σ are two permutations on the set B [3, 1]. The cycles of α are called the *edges* of the 2-map : the cycles of σ, the *vertices* ; and those of $\alpha\sigma$, the *faces* of the 2-map C. Thus we can

*LaBRI, Université BORDEAUX I, TALENCE 33405. bergey@labri.u-bordeaux.fr

define, in a purely combinatorial way, the *genus* of a 2-map. We will say that a 2-map is *planar* if its genus is equal to 0.

An *automorphism* on a 2-map $C = (B, \alpha, \sigma)$ consists of a permutation ϕ such $\phi\alpha = \alpha\phi$ and $\phi\sigma = \sigma\phi$. Let C be a combinatorial map encoding a topological map $\mathcal{C}$. We can associate with each orientation-preserving automorphism of $\mathcal{C}$, an automorphism ϕ on C. Many analogues of well-known results in Riemann surfaces theory were proven concerning automorphisms on combinatorial maps [1]. In particular, it can be shown that an automorphism on a planar map has two (and only two) fixed points [2].

Lienhard suggested n-maps and n-G-maps (a set B together with n or $n+1$ permutations on B) as extensions of combinatorial maps in order to describe subdivisions of orientable or non-orientable spaces of dimension n, possibly with boundary [4, 5].
$G = (B, \alpha_0, \alpha_1 \ldots, \alpha_n)$ is a *n-G-map* if : permutations α_i $(i < n)$ and products $\alpha_i\alpha_j$ $(|i-j| \leq 2)$ are involutions without fixed points ; α_n is an involution ; G is connected.
A connected component of $G_i = (B, \alpha_0 \ldots, \alpha_{i-1}, \alpha_{i+1} \ldots, \alpha_n)$ is called an *i-cell*.

Let us define an automorphism on a n-G-map or n-map G in the following way : a permutation ϕ wich commutes with every permutation of G is called an automorphism of G. We consider automorphisms on n-G-maps without boundary.

We show that if an n-G-map G is orientable, then the automorphisms of G fall into two classes (Φ and $\overline{\Phi}$) of automorphisms : orientation-preserving automorphisms (Φ) and orientation-reversing automorphisms ($\overline{\Phi}$). Obviously, Φ is a subgroup of the group of the automorphisms of G.

Let $\overline{G}$ be the n-map associated with an orientable n-G-map G. As n-maps describe only orientable subdivisions, we can associate in a one-to-one manner an orientation-preserving automorphism of G to each automorphism of $\overline{G}$.

Therefore, in 2 dimensions, more information about the symmetry of a topological map is given using 2-G-maps instead of 2-maps encoding.

- We can express any orientation-reversing symmetry of a topological map C as an automorphism of the associated 2-G-map G. For example : when C is a planar map, automorphisms of G are associated with symmetries around the center of the sphere or equatorial planes.
- As we can use a 2-G-map G to encode graph drawings on non-orientable surfaces, we can also treat symmetries of such embeddings as an automorphism of G.

Let ϕ be an automorphism of a 2-G-map G. We show that there are two classes of cells that are fixed by ϕ, let us call them *even* and *odd*-cells respectively. Let us consider relations between genus, orientability of G and the number of cells left fixed by ϕ.
When G is orientable, ϕ has only odd-cells if it is an orientation-reversing automorphism, even-cells if it preserves the orientation.
If the genus of G is equal to 0, then ϕ has two even-cells if G is orientable, one if G is not orientable.
We show that every odd-cell is adjacent exactly to two odd-cells. Thus, we can define *ovals* of odd-cells. These adjacency relations implie that :

- each oval contains an even number of odd-cells ;
- we have $n_v \leq n_f + n_e$, $n_e \leq n_f + n_v$, $n_f \leq n_v + n_e$, where n_e, n_v, and n_f are the numbers of odd-edges, odd-faces, and odd-vertices respectively.

If there are at least one odd-cell, then the automorphism ϕ is an involution. When G is orientable, these ovals can be considered as the intersection of the symmetry plane (associated with ϕ) with the surface on which the graph is drawn. Moreover, when G is orientable, we find an analogue to Harnack's theorem. More precisely, we have

$$g \geq n_o - 1$$

where g is the genus of G and n_o the number of ovals. This is shown by exhibiting G', a 2-G-submap of G, wich is of genus $n_o - 1$.

Let $\mathcal{C}$ be a planar topological map $\mathcal{C}$ encoded by a 2-G-map G and let $\mathcal{O}$ be an oval of an automorphism on G. Splitting the topological map along $\mathcal{O}$ let us get symmetrical drawings in the following way : we divide $\mathcal{C}$ along the cells of $\mathcal{O}$ into two parts $C^{\top}$ and $C^{\perp}$; we draw $C^{\top}$ on the top-side of the sphere (each cell of $\mathcal{O}$ lying on the equatorial plane) ; then the bottom-side of the sphere is symmetrical in relation to the equatorial plane.
In other algorithms, these splittings may be usefull in order to save space or time.

References

[1] R. Cori, A. Machí. Maps and Hypermaps : a survey I, II, III. *Expositiones Mathematicae*, **10** (1992) 403–467.

[2] R. Cori, A. Machí, J.G. Penaud and B. Vauquelin, On the automorphism group of a planar hypermap, *Europ. J. Combinatorics* **2** (1981) 331-334.

[3] J. Edmonds. A combinatorial representation for polyhedral surfaces, *Notices Amer. Math. Soc.*, **7**, 1970.

[4] P. Lienhardt. Topologic models for boundary representation : a comparison with n-dimensional generalized maps, *computer-aided design.* **23 1** (1991) 59–82.

[5] P. Lienhardt. Subdivisions de surfaces et cartes généralisées en dimension 2, *RAIRO Theoretical Informatics and Appications*, **25 2** (1991) 171–202.

Upward Drawing on Surfaces

Ivan Rival *

Extended Abstract

The modern theoretical computer science literature is preoccupied with efficient data structures to code and store ordered sets. Among these data structures, graphical ones play a decisive rôle, especially in decision-making problems. Choices must be made from among alternatives ranked hierarchically according to precedence or preference. And, loosely speaking, graphical data structures must be *drawn* in order that they may be easily *read.*

Besides the well known metaphors inspired by *layout design*, *project management*, and *database design*, several unexpected application areas are driving our recent investigations:

*Department of Computer Science, University of Ottawa, Ottawa K1N 6N5 Canada. rival@csi.uottawa.ca

(*i*) *Ice flows* consisting of vast areas of ice, largely of recent vintage (a few years old), interspersed with old ice (many years old) pose a profound danger to boats and oil-rigs, indeed, for any man-made vessel at all. Ocean currents, wind, and temperature affect the icebergs' direction of flow, changing position and velocity substantially — even within hours.

(*ii*) The increasing use of personal workstations has led to program visualization techniques in order to grasp complex computer programs. *Fisheye* techniques provide one such tool to elucidate the structure and behaviour of computer programs. That, in turn, can simplify, and hence advance the effectiveness of programming.

(*iii*) Inspired by the problem to unify the known forces, *quantum topology* combines *space* and *time* to produce a 4-dimensional picture of the world. In this rarified air, current research in physics meets up with classical mathematics.

From the viewpoint of graph drawing there is a common thread to these themes. Each involves an *ordered set* (whether it consists of moving icebergs, hierarchies of subroutines, or light cones) and each views the ordered set *monotonically* on a *two-dimensional surface*. We are led ineluctably to study *upward drawings* of ordered sets with vertices drawn on an oriented surface (usually in **3**–space) whose edges are monotonice paths with respect to the *z-axis*. The tools of *topological graph theory* and the traditional machinery of *differential topology*, may be brought to bear.

Here are some of the highlights of our work.

1. *Every triangle-free graph has a planar upward drawing.*[**Kisielewicz/Rival 1993**]

2. *Every ordered set has an upward drawing on a vertical multiple-holed torus.* [**Nowakowski/Rival 1989**]

3. *The order genus of an ordered set is an invariant among all of the orientations of its covering graph.*[**Ewacha/Li/Rival 1991**]

4. *Every bounded ordered set of genus g has an upward drawing on a surface with precisely g saddlepoints.* [**Musin/Rival/Tarasov 1993**]

These results, in turn, create new algorithmic considerations. For instance, as the *covering graph* of an ordered is *triangle-free*, we must replace the commonplace reliance on *triangulation* by an analogous, but different, subdivision: *pentangulation*. They inspire intriguing (and disarmingly simple) questions, too: *does every ordered set have a cellular upward drawing on a surface?*

References

K. Ewacha, W. Li, and I. Rival (1991) Order, genus and diagram invariance, *ORDER*, **8**, 107–113.

A. Kisielewicz and I. Rival (1993) Every triangle-free graph has an upward drawing, *ORDER* **10**, 1–16.

O. R. Musin, I. Rival and S. Tarasov (1993) Upward drawings on surfaces and the index of singularity, manuscript.

R. Nowakowski and I. Rival (1989) Bending and stretching orders into three channels, *Technical report, University of Ottawa.*

I. Rival (1993) Reading, drawing and order, *Algebras and Order* (eds. I. Rosenberg and G. Sabidussi), *NATO ASI*, Kluwer, pp. 359–404.

Tessalation and Visibility Representations of Maps on the Torus

Bojan Mohar* **and Pierre Rosenstiehl** *

It is shown that maps on the torus whose universal covering graph is 2-connected behave very much like 2-connected plane graphs. In particular, the results on visibility, tessalation representations and upward drawings of plane graphs are generalized to maps on the torus.

A Simple Construction of High Representativity Triangulations

Teresa M. Przytycka * **and Józef H. Przytycki** †

One natural way of drawing a topological surface is to start with a drawing of its triangulation. Thus one can ask which triangulation of a given surface leads to a nice drawing of the surface. The best candidate for such a triangulation is a triangulation in which all vertices are "evenly distributed" over the surface. Such a triangulation can be achieved by maximizing a parameter of surface triangulation called *representativity*.

The concept of the representativity of a graph embedding was introduced by Robertson and Seymour [5]. Robertson and Vitray [6] consider as a major effect of high representativity the fact that it makes the embedding "highly locally planar" and that "the locally Euclidean property of the surface is mirrored by the locally planar property of the embedded graph".

Formally, the representativity of a graph G embedded in a surface Σ is equal to the length of the shortest noncontractible facial walk (a walk of type $v_1, f_1, v_2, f_2, \ldots, v_k, f_k, v_1$, where for any i v_i is a vertex, f_i is a face of the graph and v_i, v_{i+1} are vertices of f_i). In particular, the representativity of a surface triangulation is equal to the length of the shortest noncontractible cycle of the triangulation. (Recall that a cycle, C, on a surface Σ is called *noncontractible* if none of the components of $\Sigma - C$ is homeomorphic to an open disc.)

It is not known what is the highest possible representativity that can be achieved when triangulating a genus g surface, Σ_g, with an n-vertex graph. Joan Hutchinson [2] showed that the representativity of such a triangulation is at most $c''\sqrt{n/g}\log g$, where c'' is a constant. Hutchinson gave a simple construction that allows to triangulate Σ_g with representativity $\Theta(\sqrt{n/g})$. This result was improved to $\Omega(\sqrt{n/g}\log^* g)$([3]) and further $\Omega(\sqrt{n/g}\sqrt{\log\log g})$ ([4]) using a covering spaces technique.

*University of Ljubljana, Slovenia and EHESS, Paris.

*Department of Mathematics and Computer Science, Odense University, DK-5230 Odense M, Denmark. przytyck@imada.ou.dk & jozef@imada.ou.dk

In this paper, we improve substantially the previously known lower bound for the representativity of such a triangulation and give an efficient algorithm for its construction. More precisely, we present an $O(\max(g^2, n))$-time algorithm that, for given g and n such that $g > 1$ and $n > c'g \log g$, constructs an n-vertex triangulation of a genus g surface, Σ_g, such that the representativity of the triangulation is at least $c\sqrt{n/g}\sqrt{\log g}$ (where c, c' are constants). We extend our result to nonorientable surfaces and surfaces with boundary. In the later case we replace the genus, g, of the surface with parameter $g' = g + d/2$, where d is the number of boundary components.

In our construction we use a relation between cubic graphs and orientable surfaces without boundary. Namely, given a $2N$-vertex cubic graph, G_0, one can construct an orientable surface Σ_g, where $g = N+1$. In fact, Σ_g can be taken as a tubular neighborhood of G_0 embedded in R^3. Then we can move G_0 so that it is embedded in Σ_g and thus all cycles of G_0 are noncontractible in Σ_0.

Let ℓ be the *girth* (the length of the shortest cycle) of G_0. Given the embedding of a cubic graph G_0 as described above, we construct a triangulation of Σ_g by adding new vertices in such a way that no noncontractible cycle shorter than ℓ is introduced. The total number of vertices added is $N_g = O(N\ell)$. The construction takes $O(N_g)$ time. Thus to maximize the representativity of the triangulation obtained using this construction we should start with G_0 being a $(3, \ell)-$cage, where a $(3, \ell)$-*cage* is a cubic graph with girth ℓ and the smallest possible number of vertices. Unfortunately, there is no efficient algorithm that, for a given value ℓ, constructs a $(3, \ell)$-*cage*. However, by a theorem of Erdös and Sachs [1], for any integer ℓ, there exists a cubic graph of girth ℓ and $O(2^\ell)$ vertices.

Based on the proof of the theorem of Erdös and Sachs we give an $O(N^2)$-time algorithm to construct a cubic graph with $2N$ vertices and girth $\Theta(\log N)$. This gives us, for any $N \geq 1$, an N_g-vertex triangulation, T_g, of Σ_g such that representativity of T_g is $\Theta(\log N)$ and $N_g = O(N \log N)$. Thus the lower bound claimed is achieved for $n = N_g$. The next step is to extend the lower bound to all values of n. We achieve this by appropriate subdivision of the triangulation T_g. Finally, we show an extension of the construction to nonorientable surfaces and surfaces with boundary. We achieve this by cutting some handles of the surface $\Sigma_{\lceil g' \rceil}$ along edges of the $T_{\lceil g' \rceil}$. To obtain triangulations of nonorientable surfaces we cap off at most two of the holes created in the above construction with (triangulated) Möbius bands.

References

[1] P.Erdös, H.Sachs. Reguläre Graphen gegebener Taillenweite mit minimaler Knotenzahl, Wiss. Z. Univ. Halle-Wittenberg Math.-Nat. 12 (1963), 251-257

[2] J.Hutchinson, On short noncontractible cycles in embedded graphs, *SIAM J. Disc. Math,* (1988) 185-192.

[3] T.M.Przytycka, J.H.Przytycki, On lower bound for short noncontractible cycles in embedded graphs, *SIAM J. Disc. Math,* (1990) 281-293.

[4] T.M.Przytycka, J.H.Przytycki, Surface Triangulations Without Short Noncontractible Cycles, *Contemporary Mathematics, Graph Structure Theory* 147, (1993) 303–349.

[5] N.Robertson, P.D.Seymour, Graph minors. VII. Disjoint paths on a surface, *J.Comb. Theory,* Ser. B, 48 (1990), 212-254.

[6] N.Robertson, R.Vitray, Representativity of surface embeddings. *Algorithms and Combinatorics, Volume 9, Paths, Flows and VLSI-Layout.* Springer-Verlag, Berlin, Heidelberg, Volume Editors: B. Korte, L. Lovász, H. J. Promel, and A. Schrijver, (1990), 293 - 328.

On a Visibility Representation for Graphs in Three Dimensions

Prosenjit Bose,[*] Hazel Everett,[†] Sándor Fekete,[‡] Anna Lubiw,[§] Henk Meijer,[¶] Kathleen Romanik,[‖] Tom Shermer[**] and Sue Whitesides[††]

Visibility representations of graphs map vertices to sets in Euclidean space and express edges as visibility relations between these sets. Application areas such as VLSI wire routing and circuit board layout have stimulated research on visibility representations where the sets belong to R^2. Here, motivated by the emerging research area of graph drawing, we study a 3-dimensional visibility representation.

Consider an arrangement of closed, disjoint rectangles in R^3 such that the planes determined by the rectangles are perpendicular to the z (vertical) direction, and the sides of the rectangles are parallel to the x or y (horizontal) directions. A vertical *thick line of sight* between two rectangles R_i and R_j is a closed cylinder C of non-zero length and radius such that the ends of C are contained in R_i and R_j, the axis of C is parallel to the z direction, and the intersection of C with any other rectangle in the arrangement is empty. We call such an arrangement a *B-representation* of an abstract graph $G = (V, E)$ if, and only if, the following hold:

- there exists a 1-1 onto correspondence between the rectangles and the vertices, and
- vertices v_i and v_j are adjacent if, and only if, their corresponding rectangles R_i and R_j have a vertical thick line of sight between them.

Our main results are as follows. All planar graphs are B-representable, as are many non-planar graphs. In particular, $K_{m,n}$ is B-representable for all m and n, and K_n is B-representable for values of $n \leq 20$. However, K_n is not B-representable for $n \geq 103$. We have also considered variants of B-representations.

B-representability of planar graphs: The proof that all planar graphs are B-representable has two main ingredients. The first is the result due independently to Wismath[W] and to Tamassia and Tollis[TT] that any 2-connected planar graph has what [TT] calls an *ϵ-visibility* representation. (Vertices correspond to closed, disjoint, horizontal line segments in the plane, and two vertices are adjacent in the graph if, and only if, their corresponding segments can be joined by a vertical band of non-zero width (and length) with ends lying in the segments.) The second ingredient is the use of the 3rd dimension to deal with cut vertices. This is similar to an idea of [W] for obtaining a visibility representation for all planar graphs by rectangles in R^2 that can look in both x and y directions.

[*]McGill University, Canada. jit@muff.cs.mcgill.ca
[†]Université du Québec à Montréal, Canada. hazel@opus.cs.mcgill.ca
[‡]SUNY Stony Brook, USA. sandor@ams.sunysb.edu
[§]University of Waterloo, Canada. alubiw@maytag.uwaterloo.ca
[¶]Queen's University, Canada. henk@qucis.queensu.ca
[‖]McGill University, Canada. romanik@opus.cs.mcgill.ca
[**]Simon Fraser University, Canada. shermer@cs.sfu.edu
[††]McGill University, Canada. sue@cs.mcgill.ca

B-representability of $K_{m,n}$ and K_n: $K_{m,n}$ has a simple, general B-representation for all m, n, but this is not the case for K_n. While we have constructed an explicit B-representation for K_{20}, and hence for K_n, $n \leq 20$, we have also shown that K_n has no B-representation for $n \geq 103$. We conjecture that $n = 20$ is close to the correct bound.

Non-B-representability of K_n for $n \geq 103$: In any B-representation of a complete graph, no two rectangles can lie on the same z=constant plane. Also, if K_n has a B-representation, then it has one in which no two rectangles have sides at the same x or y=constant values. In other words, K_n can be represented by rectangles that can be linearly ordered by z coordinate and also, by x coordinate of left side, x coordinate of right side, y coordinate of top side and y coordinate of bottom side. To obtain the result, we use such orderings together with repeated application of the following result of Erdös and Szekeres[ES]:
For any positive integers j and k, any sequence of more than jk distinct integers has a (not necessarily contiguous) increasing subsequence of length $j+1$ or decreasing subsequence of length $k+1$.

Variations: Our representation of K_{20} can be carried out with squares, provided the squares need not have the same area. Representations by discs, unit squares, and squares for complete graphs and for complete bipartite graphs have been considered.

Acknowledgment: Our study of B-representations began at Bellairs Research Institute of McGill University during the Workshop on Visibility Representations organized by S. Whitesides and J. Hutchinson, February 12-19, 1993. We are grateful to the other conference participants Joan Hutchinson, Goos Kant, Marc van Kreveld, Beppe Liotta, Steve Skiena, Roberto Tamassia, Yanni Tollis, and Godfried Toussaint.

References

[ES] Erdös, P. and Szekeres, A., "A combinatorial problem in geometry, " *Compositio Mathematica* v. 2 (1935), 463-470.

[TT] Tamassia, R. and Tollis, I. G., "A unified approach to visibility representations of planar graphs," *Discrete Comput. Geom.* v. 1 (1986), 321-341.

[W] Wismath, Stephen Kenneth. Bar-Representable Visibility Graphs and a Related Network Flow Problem. University of British Columbia, Dept. of Computer Science Technical Report 89-24, August 1989.

On Graph Drawings with Smallest Number of Faces

Jianer Chen, * **Saroja P. Kanchi,** * **and Jonathan L. Gross**†

We report here our recent progress in the study of graph drawings with the smallest number of faces, or equivalently, graph embeddings with the largest genus. Formally, the *maximum*

*Department of Computer Science, Texas A&M University, College Station, TX 77843-3112. chen@cs.tamu.edu & skanchi@cs.tamu.edu.

†Department of Computer Science, Columbia University, New York, NY 10027.

genus $\gamma_M(G)$ of a connected graph G is defined to be the largest integer k such that there exists a cellular embedding of G into the orientable surface of genus k.

Since the introductory investigation of maximum genus by Nordhaus, Stewart, and White [5], there has been considerable interest in maximum genus embeddings of graphs. Two outstanding results in this research are Xuong's characterization of maximum genus embedding in terms of components of the complements of spanning trees [9] and a first polynomial-time algorithm for computing maximum genus developed by Furst, Gross, and McGeoch [3].

Recent investigations have focused on deriving a lower bound on the maximum genus of graphs. Skoviera [8] showed that the maximum genus of a 2-connected graph of diameter 2 is at least $\lceil \beta(G)/2 \rceil - 2$. More recently, Chen and Gross [1] proved that the maximum genus of a 2-connected simplicial graph or of a 3-connected graph is at least $\Omega(\log \beta(G))$. The last result was further improved by Chen, Gross, and Rieper [2] who proved that the maximum genus of a 2-connected simplicial graph G is at least $\beta(G)/8$. A related topic, the upper-embeddability of graphs, has also been studied extensively in literature [5, 4, 6, 7, 11, 8, 10].

In this paper, we prove that $\gamma_M(G) > \beta(G)/4$ for a simplicial graph G, and we show that our bound is tight. Our proof selectively sharpens Xuong's characterization of the maximum genus embedding. We first show that every 3-regular simplicial graph G has a Xuong tree T such that every odd component in the Xuong co-tree $G - T$ has only one edge. This enables us to compare the number of odd components to the number of even components in the Xuong co-tree and thereby arrive at an upper bound for the number of odd components. This upper bound is used to obtain the desired lower bound for the maximum genus of a 3-regular simplicial graph. Finally, the restriction of 3-regularity is removed by using a theorem of Chen and Gross concerning edge-contractions.

Our result on the lower bound of maximum genus has several interesting consequences to the average genus of graphs. Using techniques developed by Chen and Gross, we show that the average genus of a simplicial graph is at least 1/8 of its cycle rank. This improves a result by Chen, Gross and Rieper [2] that the average genus of a 2-connected simplicial graph is at least 1/16 of its cycle rank. Moreover, our result implies that the average genus of a simplicial graph is at least 1/4 of its maximum genus, thereby complementing another result of Chen, Gross and Rieper [2] that for a 3-regular graph, the average genus is at least half its maximum genus.

References

[1] J. Chen and J. L. Gross, Limit points for average genus I. 3-connected and 2-connected simplicial graphs, *J. Combinatorial Theory B* **55**, (1992), 83-103.

[2] J. Chen, J. L. Gross, and R. G. Rieper, Lower bounds for the average genus, Submitted for publication, (1991).

[3] M. L. Furst, J. L. Gross, and L. A. McGeoch, Finding a maximum-genus graph imbedding, *J. of ACM* **35-3**, (1988), 523-534.

[4] L. Nebeský, Every connected, locally connected graph is upper imbeddable, *J. Graph Theory* **5**, (1981), 205-207.

[5] E. Nordhaus, B. Stewart, and A. White, On the maximum genus of a graph, *J. Combinatorial Theory B* **11**, (1971), 258-267.

[6] R. Ringeisen, "The maximum genus of a graph," Ph.D. thesis, Department of Mathematics, Michigan State University, East Lansing, MI, 1970.

[7] R. RINGEISEN, Determining all compact orientable 2-manifolds upon which $K_{m,n}$ has 2-cell imbeddings, *J. Combinatorial Theory B* **12**, (1972), 101-104.

[8] M. SKOVIERA, The maximum genus of graphs of diameter two, *Discrete Mathematics* **87**, (1991), 175-180.

[9] N. H. XUONG, How to determine the maximum genus of a graph, *J. Combinatorial Theory B* **26**, (1979), 217-225.

[10] N. H. XUONG, Upper-imbeddable graphs and related topics, *J. Combinatorial Theory B* **26**, (1979), 226-232.

[11] J. ZAKS, The maximum genus of cartesian products of graphs, *Canad. J. Math.* **26-5**, (1974), 1025-1035.

A Flow Model of Low Complexity for Twisting a Layout

Marc Bousset *

We deal with (s,t)-bipolar orientations of a 2-connected plane graph G, taking local orientation constraints into account. A laterality constraint links the orientations of two edges adjacent in the circular order around a vertex in the following way :

- an **extremal** constraint on an angle implies that the two edges adjacent to that angle are either both outgoing or both incoming,
- a **lateral** constraint on an angle implies that among the two edges adjacent to that angle, one is incoming and one is outgoing.

Our approach is based on the theory of the (s,t)-bipolar marking of the angle graph $\widehat{G}$ (a special 2-colour marking of the edges of $\widehat{G}$) introduced by P. Rosenstiehl. There is a one-to-one correspondance between (s,t)-bipolar markings of $\widehat{G}$ and (s,t)-bipolar orientations of G.

The local invariants on the colours around each vertex and around each face allow us to translate the problem into mathematical programming, namely into a flow model. The network is built upon the angle graph $\widehat{G}$, to which we add two nodes S and T. We also add edges from S to each vertex of G (V-edges) and from each face of G to T (F-edges). The capacities of the angles are 1 (when there are no constraints). The capacities of the V-edges are 2 except for (S,s) and (S,t) for which they are zero. The capacities of each F-edge (f,T) is equal to the degree of the face f minus 2. A maximum flow in this network saturates both the V-edges and the F-edges, and the flow through each angle gives an (s,t)-bipolar marking.

The integer capacities around S and T and the unit capacities elsewhere allow us to extend a result of R. E. Tarjan, that is : a drastic reduction of the complexity of the maximum flow algorithm down to $\mathcal{O}(m\sqrt{m})$ for Dinic's algorithm.

Laterality constraints are enforced by modifying the capacities on the angles. When no (s,t)-bipolar marking consistant with the constraints exists, the maximum flow does not

*C.A.M.S. - E.H.E.S.S, Paris. bousset@dassault-avion.fr

saturate all the V-edges and the F-edges. It is then possible to identify a set of angles $\mathcal{I}$ so that relaxing the constraints on all angles of $\mathcal{I}$ guarantees the existence of an (s,t)-bipolar marking compatible with the remaining constraints.

The method developed here has been implemented in an industrial context in a CAD system for generating layouts of electrical networks, and also in the TWIST software, created for the ALCOM project.

References

[1] M. Bousset and P. Rosenstiehl. *TWIST (Version 1)*, 1990. ALCOM report 90-73.

[2] M. Bousset and P. Rosenstiehl. *TWIST (Version 2)*, 1990. ALCOM report 91-103.

[3] R. Cori. *Un code pour les graphes planaires et ses applications*, volume 27 of *Astérisque*. Société Mathématique de France, 1975.

[4] H. de Fraysseix, P. Ossona de Mendez, and P. Rosenstiehl. Bipolar orientations revisited. *Discrete and Applied Mathematics*, (to appear).

[5] E. A. Dinic. Algorithm for solution of a problem of maximum flow in a network with power estimation. *Soviet Math. Dokl.*, 11:1277–1280, 1970.

[6] S. Even and R. E. Tarjan. Network flow and testing graph connectivity. *SIAM J. Comput.*, 4(4):507–518, 1975.

[7] P. Rosenstiehl. Embedding in the plane with orientation constraints : The angle graph. In *Proceedings of the Third International Conference (New York, 1985)*, pages 340–346. Annals of the New York Academy of Sciences, 1989. 555.

[8] P. Rosenstiehl and R. E. Tarjan. Rectilinear planar layouts and bipolar orientations of planar graphs. *Discrete and Computational Geometry*, 1:343–353, 1986.

[9] R. E. Tarjan. *Data Structures and Network Algorithms*. Society for Industrial and Applied Mathematics, 1983.

[10] M. Veldhorst. A bibliography on network flow problems. *Algorithms Review*, 1(2):97–117, 1990. ALCOM.

Convex and non-Convex Cost Functions* of Orthogonal Representations

Giuseppe Di Battista † **Giuseppe Liotta** ‡ **and Francesco Vargiu** §

*Work partially supported by Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo of the Italian National Research Council (CNR) and by Esprit BRA of the EC Under Contract 7141 Alcom II

†Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", via Salaria 113, I-00198 Roma, Italia. dibattista@iasi.rm.cnr.it

‡Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", via Salaria 113, I-00198 Roma, Italia. Part of this work has been done when this author was visiting the Department of Computer Science of McGill University. liotta@infokit.ing.uniroma1.it

§Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", via Salaria 113, I-00198 Roma, Italia and Database Informatica Spa. vargiu@infokit.ing.uniroma1.it

An orthogonal drawing of a graph is a planar drawing such that all the edges are polygonal chains of horizontal and vertical segments. Finding the planar embedding of a planar graph such that its orthogonal drawing has the minimum number of bends is a fundamental open problem in graph drawing. We provide the first partial solution to the problem. First, we give a new combinatorial characterization of orthogonal representations of 2-connected graphs based on the concept of spirality and we relate the number of bends of representations to the spirality by means of the concept of cost function. Second we exploit the behaviour of cost functions associated to orthogonal representations of components of 2-connected graphs. Third we use the characterization to find in polynomial time the planar embedding of a series-parallel graph and of a 2-connected 3-planar graph such that its orthogonal drawing has the minimum number of bends.

In this talk we give a new combinatorial characterization of orthogonal representations of 2-connected graphs and use the results to provide a first partial solution to the problem of finding the planar embedding of a planar graph such that its of orthogonal drawing has the minimum number of bends.

First, we introduce the new concept of *spirality*, that is a measure of how an orthogonal drawing is "rolled up", giving a combinatorial characterization that relates the number of bends of an orthogonal drawing and its spirality by means of the concept of cost function.

Second we study the behaviour of cost functions of 2-connected graphs, with the following results:

- Cost functions of components of 3-planar graphs are non decreasing convex piecewice linear functions.
- Cost functions of components of 4-planar graphs are piecewice linear functions, but possibly concave ("w"-shape).

Third, from the above results, we solve in polynomial time the problem of finding the planar embedding that leads to the orthogonal drawing with the minimum number of bends for 3-planar graphs and series-parallel graphs.

Minimizing the number of bends of orthogonal representations is a classical problem of graph drawing.

Tamassia [9] has proposed a very elegant representation algorithm that solves the problem in polynomial time for graphs with a fixed embedding. The algorithm is based on a combinatorial characterization that allows to map the problem into a min-cost-flow one. Linear time heuristics for the same problem have been proposed by Tamassia and Tollis in [12, 13]. Such heuristics guarantee at most $2n + 4$ bends for a biconnected graph with n vertices. Recently, Kant [7] has proposed efficient heuristics with better bounds for triconnected 4-planar graphs and general 3-planar graphs (a graph is k-planar if it is planar and each vertex has degree at most k). Tamassia, Tollis and Vitter [14] have given lower bounds for the problem and the first parallel algorithm. A brief survey on orthogonal drawings is in [10].

However, all the above papers work within a fixed embedding, where it can be seen that the choice of the embedding can deeply affect the results obtained by the algorithms. The problem of finding the planar embedding that leads to the minimum number of bends is not known to be NP-hard or not and has been explicitely mentioned as open by several

authors. Although the problem is quite natural there are only a few contributions on this topic, because of the exponential number of embeddings a planar graph (in general) has.

Our technique exploits both the properties of the spirality and a variation of the $SPQR$ trees [2, 3]: a data structure that implicitely represents all the planar embeddings of a planar graph. Moreover we adopt a slight modification of the algorithm of Tamassia [9] for computing orthogonal representation of triconnected components.

Observe that series-parallel graphs arise in a variety of problems such as scheduling, electrical networks, data-flow analysis, database logic programs, and circuit layout. Also, they play a very special role in planarity problems [2, 3].

References

[1] P. Bertolazzi, R.F. Cohen, G. Di Battista, R. Tamassia, and I.G. Tollis, "How to Draw a Series-Parallel Digraph," Proc. 3rd Scandinavian Workshop on Algorithm Theory, 1992.

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

[3] G. Di Battista and R. Tamassia "On Line Planarity Testing," Technical Report CS-89-31, Dept. of Computer Science, Brown Univ. 1989.

[4] P. Eades and R. Tamassia, "Algorithms for Automatic Graph Drawing: An Annotated Bibliography," Technical Report CS-89-09, Dept. of Computer Science, Brown Univ. 1989.

[5] S. Even "Graph Algoritms," Computer Science Press, Potomac, MD, 1979.

[6] D. R. Fulkerson "An Out-of-Kilter Method for Minimal Cost Flow Problems," SIAM Journal Appl. Math. no. 9, pp. 18-27, 1961.

[7] G. Kant "A New Method for Planar Graph Drawings on a Grid," Proc. IEEE Symp. on Foundations of Computer Science, 1992.

[8] E. L. Lawler "Combinatorial Optimization: Networks and Matroids," Holt, Rinehart and Winston, New York, Chapt. 4, 1976.

[9] 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.

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

[11] 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.

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

[13] 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.

[14] R. Tamassia, I. G. Tollis, and J. S. Vitter "Lower Bounds and Parallel Algorithms for Planar Orthogonal Grid Drawings," Proc. IEEE Symp. on Parallel and Distributed Processing, 1991.

[15] K. Takamizawa, T. Nishizeki, and N. Saito "Linear-Time Computability of Combinatorial Problems on Series-Parallel Graphs," Journal of the ACM, vol. 29, no. 3, pp. 623-641, 1982.

[16] J. Valdes, R.E. Tarjan, and E.L. Lawler "The Recognition of Series Parallel Digraphs," SIAM J. on Comp., No.11, 1982.

[17] L. Valiant "Universality Considerations in VLSI Circuits," IEEE Trans. on Computers, vol. c-30, no. 2, pp. 135-140, 1981.

Topology and Geometry of Planar Triangular Graphs *

Giuseppe Di Battista † **and Luca Vismara** ‡

The contribution of this talk is twofold. On one side, we give several topological results: (1) a new operation to construct plane triangular graphs from a triangle graph; (2) a basic lemma about the planarity of all the drawings of plane triangular graphs; (3) a new ordering for the vertices of plane triangular graphs. On the other side, we give a characterization of all the planar drawings of a triangular graph by means of a system of equations and inequalities relating its angles, solving a problem that is explicitely mentioned as open by several authors; we also discuss minimality properties of the characterization. The characterization can be used: (1) to decide in linear time whether a given distribution of angles between the edges of a planar triangular graph can result in a planar drawing; (2) to tackle the problem of maximizing the minimum angle of the drawing of a planar triangular graph by studying the solution-space of a non-linear optimization problem; (3) to give a characterization of the planar drawings of a triconnected graph through a system of equations and inequalities relating its angles; (4) to give a characterization of Delaunay triangulations through a system of equations and inequalities relating its angles; (5) to give a characterization of all the planar drawings of a triangular graph through a system of equations and inequalities relating the length of its edges; in turn, this result allows to give a new characterization of the disc packing representations of planar triangular graphs.

Introduction

Planar straight-line drawings of planar graphs are a classical topic of the graph drawing field (surveys on graph drawing can be found in [18, 7]).

A classical result established by Wagner, Fary, Stein, and Steinitz shows that every planar graph has a planar straight-line drawing [17, 22, 8, 16].

A grid drawing is a drawing in which the vertices have integer coordinates. Independently, de Fraysseix, Pach, and Pollak [2, 3], and Schnyder [15] have shown that every n-vertex planar graph has a planar straight-line grid drawing with $O(n^2)$ area.

Straight-line drawings have also been studied with the constraint for all the faces to be represented by convex polygons (convex drawings) [19, 20]. Tutte shows that, for a triconnected graph, convex drawings can be constructed by solving a system of linear equations. Recently, Kant has shown an algorithm to construct grid convex drawings with quadratic area [11].

In the research on planar straight-line drawings a very special role is played by angles between the segments that compose the drawing. In particular, Vijayan [21] studied angle graphs. An angle graph is a planar embedded graph in which the angles between successive edges incident at vertices are given. The problem of the existence of a planar straight-line

*Research supported in part by ESPRIT Basic Research Action No. 7141 (ALCOM II) by Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo of the Italian National Research Council (CNR).

†Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Via Salaria, 113 - 00198 Roma, Italy. dibattista@iasi.rm.cnr.it

‡Research performed in part while this author was at IASI-CNR. Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Via Salaria, 113 - 00198 Roma, Italy. vismara@iasi.rm.cnr.it

drawing of an angle graph that preserves the angles is tackled and partial characterization results are shown.

In [9] the problem of constructing straight-line drawings of graphs with large angles is studied. 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 vertex degree of the graph. Other results are given for particular classes of graphs. For planar graphs the bound is improved to $O(1/d)$; however, in general, the obtained drawing is non-planar.

Malitz and Papakostas [12] have shown that it is always possible to construct a planar straight-line drawing of a planar graph whose smallest angle is $O(\alpha^d)$, where $0 < \alpha < 1$. The bound that is presented is only existential; in fact they exploits a disc packing representation of the graph. In a disc packing representation (1) each vertex is a disc, (2) two vertices are adjacent in the graph if and only if their discs are tangent, and (3) the interiors of the discs are pairwise disjoint. No polynomial algorithm is known to construct a disc packing representation. Recently, Mohar [13] has shown for this problem an approximation algorithm.

A fundamental tool for several algorithms and characterizations described above are planar triangular graphs. For instance the algorithm by de Fraysseix et al. and the algorithm by Schnyder have an intermediate step in which the given planar graph is triangulated. Also, planar triangular graphs play a very special role in a number of problems arising in Computational Geometry. However, as far as we know, characterizing angles of planar triangular graphs has been an elusive goal for a long time. The contribution of this paper can be summarized as follows:

- We define a new operation, named *close-wheel*, such that any plane triangular graph G with n vertices can be constructed starting from the triangle graph by a sequence of $O(n)$ close-wheel operations.
- We prove a lemma about the planarity of all the drawings of plane triangular graphs.
- We define a new ordering method for the vertices of plane triangular graphs.
- We give a characterization of all the planar drawings of a triangular graph through a system of equations and inequalities relating its angles. The problem is explicitely mentioned as open by several authors (see e.g. [21, 1, 12]) We also discuss minimality properties of the characterization.

The characterization above has several applications.

- It can be used to decide in linear time whether a given distribution of angles between the edges of a planar triangular graph can result in a planar drawing.
- It allows to tackle the problem of maximizing the minimum angle of the drawing of a planar triangular graph by studying the solution-space of a non-linear optimization problem.
- It gives a characterization of the planar drawings of a triconnected graph through a system of equations and inequalities relating its angles.
- It gives a characterization of Delaunay triangulations through a system of equations and inequalities relating its angles, solving a problem stated in [4]. Recently, the problem of characterizing angles of Delaunay triangulations has been tackled by Dillencourt and Rivin who have shown that a system of equations and inequalities relating the angles of a plane triangular graph G can be used to decide whether G can be drawn as a Delaunay triangulation [6].

- It can be exploited to give a characterization of all the planar drawings of a triangular graph through a system of equations and inequalities relating the length of its edges; in turn, this result allows to give a new characterization of the disc packing representations of planar triangular graphs.

References

[1] F.J. Brandenburg, P. Kleinschmidt, and U. Schnieders, "Drawing Planar Graphs with Wide Angles," Manuscript, 1991.

[2] 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.

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

[4] M.B. Dillencourt, "Graph-Theoretical Properties of Algorithms Involving Delaunay Triangulations," Tech. Rep. CS-TR-2059, University of Maryland, 1988.

[5] M.B. Dillencourt, "Toughness and Delaunay Triangulations," Discrete & Computational Geometry, vol. 5, no. 6, pp. 575-601, 1990.

[6] M.B. Dillencourt and I. Rivin, Personal Communication, 1992.

[7] P. Eades and R. Tamassia, "Algorithms for Automatic Graph Drawing: An Annotated Bibliography," Technical Report CS-89-09, Dept. of Computer Science, Brown Univ., 1989.

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

[9] 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.

[10] C.D. Hodgson, I. Rivin, and W.D. Smith, "A Characterization of Convex Hyperbolic Polyhedra and od Convex Polyhedra Inscribed in the Sphere," Bull. of the American Mathematical Society, vol. 27, no. 2, pp. 246-251, 1992.

[11] G. Kant, "A New Method for Planar Graph Drawings on a Grid," Proc. IEEE Symp. on Foundations of Computer Science, 1992.

[12] 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.

[13] B. Mohar, "Circle packings of maps in polynomial time," Manuscript, submitted to the Bull. of the American Mathematical Society.

[14] T. Nishizeki and N. Chiba, Planar Graphs: Theory and Algorithms, Annals of Discrete Mathematics, North Holland, 1988.

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

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

[17] E. Steinitz and H. Rademacher, Vorlesung uber die

[18] 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.

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

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

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

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

An Optimal PRAM Algorithms for Planar Convex Embedding

Frank Dehne,* † **Hristo Djidjev,**‡ **and Jörg-Rüdiger Sack**§

Introduction

The task of representing a diagram in understandable and readable form arises in a variety of areas including, e.g., circuit design and information system analysis/design. The reader is referred to Eades and Tamassia [6] for an annotated bibliography of graph drawing algorithms.

The existence of a planar representation of a graph can be tested in linear time as was first proved by Hopcroft and Tarjan [1]. Chiba et al. [4] provided a linear time algorithm to find a planar representation of a planar graph. Their result is based on an algorithm to determine the existence of a planar representation due to Booth and Lueker [2].

A subclass of planar graphs whose representation is particularly aestetically pleasing are those planar graphs whose bounded faces can all be drawn convexly, i.e. the embedding of each bounded face is a convex polygon. Thomassen [19] describes a sequential method for embedding a planar graph, once an extendible outer cycle F is given. An extendible outer facial cycle F of G can be determined in linear time as described in [4, 14]. See also for other related work [K92, 20, 21].

Results

We give an optimal parallel algorithm for determining whether a graph can be convexly embedded and if so, for constructing such a convex embedding for the graph. The algorithm developed runs in $O(\log n)$ time with $O(n)$ space and the same processor bound as graph connectivity.

*This research was partially supported by Natural Sciences and Engineering Council of Canada.

†School of Computer Science, Carleton University, Ottawa, Ontario K1S 5B6, Canada. dehne@scs.carleton.ca

‡Department of Computer Science, Rice University, Houston, Texas 77251, USA.

§School of Computer Science, Carleton University, Ottawa, Ontario K1S 5B6, Canada, sack@scs.carleton.ca

Our solution to this problem is based on the following. Using [s,t]-numbers we present a decomposition scheme and develop a parallel algorithm to determine such a decomposition. To embed the paths so that each face of the original is convex we generalize the notion of convexity of polygons to pseudo-convexity. We give an algorithm to embed a path inside a pseudo-convex polygon so that the resulting subpolygons are also pseudo-convex. When applying this embedding to the path decomposition, a convex embedding of the entire graph is obtained.

We use the following parallel algorithms: planarity testing [15], st- numbering [12], list-ranking [5], lowest common ancestor in a tree [16], biconnectivity [18] (this algorithm is not optimal as described, but with list-ranking algorithm it becomes optimal), triconnectivity [7], parallel transitive closure and point location in planar structures [17].

References

[1] O. Berkman, B. Schieber and U. Vishkin, *Some doubly logarithmic optimal parallel algorithms based on finding all nearest smaller values*, Technical Report UMIACS-TR-88-79, University of Maryland, 1988.

[2] K. Booth, G. Lueker, *Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithm*, J. Comp. Syst. Sci. 13, 1976, pp. 335-379.

[3] N. Chiba, K. Onoguchi, T. Nishizeki, *Drawing planar graphs nicely*, Acta Informatica 22, 1985, pp. 187-201.

[4] N. Chiba, T. Yamanouchi, T. Nishizeki, *Linear algorithms for convex drawings of planar graphs*, in Progress in Graph Theory, J.A. Bondy and U.S.R. Murth (eds.), Academic Press, 1984, pp. 153-173.

[5] R. Cole, U. Vishkin, *Optimal parallel algorithms for expression tree evaluation and list ranking*, Proc. AWOC'88, Lecture Notes in Computer Science 319, Springer-Verlag, 1988, pp. 91-100.

[6] G. Di Battista, P. Eades, R. Tamassia, and I.G. Tollis *Algorithms for drawing graphs: an annotated bibliography*, Technical Report, Brown University, 1988; updated 1993.

[7] D. Fussell, V. Ramachandran, R. Thurimella, *Finding triconnected components by local replacements*, Proc. ICALP 89, LNCS 372, Springer-Verlag, 1989, pp. 379-393.

[8] H. Gazit, *Optimal EREW parallel algorithms for connectivity, ear decomposition and st-numbering of planar graphs*, Proc. 5th Int. Parallel Processing Symp., 1991.

[9] J. Hopcroft and R.E. Tarjan, *Efficient planarity testing*, J.ACM, 21:4, 1974, pp. 549-568.

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

[11] A. Lempel, S. Even, I. Cederbaum, *An algorithm for planarity testing of a graph*, Theory of Graphs: International Symposium, Gordon and Breach, New York, 1967, pp. 215-232.

[12] Y. Maon, B. Schieber, U. Vishkin, *Parallel ear decomposition search (EDS) and st-numbering in graphs*, TCS 47, 1986, pp. 277-296.

[13] G. Miller, V. Ramachandran, *A new graph triconnectivity algorithm and its parallelization*, ACM Symp. on Theory of Computing, 1987, pp. 335-344.

[14] T. Nishizeki, N. Chiba, *Planar graphs: theory and algorithms*, North Holland, 1988.

[15] V. Ramachandran, J. Reif, *An optimal parallel algorithm for graph planarity*, Proc. 30th Ann. IEEE Symp. on Found. of Comp. Sci., 1989, pp. 282-287.

[16] B. Schieber, U. Vishkin, *On finding lowest common ancestors: simplification and parallelization*, Proc. AWOC'88, LNCS 319, Springer-Verlag, 1988, pp. 111-123.

[17] R. Tamassia and J. S. Vitter, Parallel transitive closure and point location in planar structures, SIAM J. Comput. 20:4, pp. 708-725, 1991.

[18] R.E. Tarjan, U.Vishkin, *An efficient parallel biconnectivity algorithm*, SIAM J. Computing 14, 1984, pp. 862-874.

[19] C. Thomassen, *Planarity and duality of finite and infinite planar graphs*, J. Combinatorial Theory, Series B 29, 1980, pp. 244-271.

[20] W.T. Tutte, *Convex Representations of Graphs*, Proc. London Math Soc., vol. 10, 1960, pp. 304-320.

[21] W.T. Tutte, *How to Draw a Graph*, Proc. London Math Soc., vol. 3, no. 13, 1963, pp. 743-768.

Algorithms for Embedding Graphs Into a 3-page Book

Miki Shimabara Miyauchi *

A *book* is a two-part object that consists of a spine, which is a line, and pages, each of which is a half-plane bounded by the spine. A *book embedding* of a graph orders the nodes of the graph linearly along the the spine of a book and arranges each edge on pages so that the edges do not intersect. Book embeddings have applications in several areas of theoretical computer science, including VLSI design [1] and complexity theory [4]. In the Diogenes method for designing fault-tolerant VLSI processor arrays [4], for example, a book embedding is used to implement the live processors of a multilayer chip (the processors extended through all the layers). The *single-row routing problem* [6] is also considered to be book embedding problem having two pages, and questions such as how to minimize the number of lines crossing the spine of the book and how to minimize the number of *tracks* of a graph have been investigated.

The book-embedding problem has been studied for several kinds of graphs. When book embeddings are restricted so that each edge is embedded on one page, for example, M. Yannakakis [8] showed that any planar graph can be embedded in a book of four pages and Chung, Leighton, and Rosenberg [1] showed that the complete graph K_n is embeddable in $n/2$ pages. When each edge can be embedded in more than one page, Atneosen [1], Babai [7], and Bernhart [2] have each shown that every graph is embeddable in a 3-page book.

Atneosen's proof, however, is nonconstructive. By using Leighton's lower bound on the crossing number of a complete graph [5], we show the following:

Theorem 1 *Babai and Bernhart's algorithm that embeds K_n into a 3-page book takes $\Omega(n^4)$ time.*

*NTT Basic Research Laboratories, 3-9-11, Midori-Cho, Musashino-Shi, Tokyo, 180 Japan. miki@ntt-20.ntt.jp

We also present a new embedding algorithm for 3-page book embeddings of graphs.

Theorem 2 *There is an algorithm that embeds any graph into 3-page book and that runs in time $O(mn)$ for a graph of size m and order n.*

Sketch of the proof. Let D be a half disk, with center c_0, on the xy-plane and let C be the boundary of D. Arrange the n nodes $V = \{v_i\}$ on C counterclockwise.

for each edge $e = (v_s, v_i)$ $(n \geq s > i > 0) \in E$ **do**
  if $s = i + 1$ or $i = 1$ **then** draw (v_s, v_i) as a straight line segment on D
  else let p_s be the middle point of the straight line segment (v_{s-1}, v_s)
    and p_s^i be the point at which the straight line segment (p_s, v_1)
    crosses the auxiliary line $c_0 v_i$. Join the pair of points $\{v_s, p_s^i\}$ by a straight
    line segment on D, and join the pair of points $\{p_s^i, v_i\}$ by a half circle
    in a plane perpendicular to D.

Let c_i be the point at which the auxiliary line $c_0 v_i$ crosses the boundary of a small neighborhood of c_0. The path v_1, c_1, c_2, v_2, v_3, c_3, ..., v_n is considered as the spine L of the book, D is considered two pages, and the third page is taken as the sharply bent surfaces perpendicular to D. The for statement repeats $m(= |E|)$ times, and each inner for loop repeats at most $n(= |V|)$ times. ■

Theorem 3 *Any algorithm embedding K_n into a 3-page book takes $\Omega(n^2)$ time.*

Theorem 4 *Let K_n be a complete graph with $n = 4s+1$ $(s \in I,\ s \geq 1)$ nodes. Then there is an embedding of K_n, into a 3-page book, with $(n-3)(5n^2 - 18n - 83)/48$ edge-crossings over the spine of the book.*

References

[1] G. A. Atneosen: On the Embeddability of Compacta in N-Books, *PH. D. Thesis*, Michigan State Univ., 1968.

[2] F. Bernhart: The Book Thickness of a Graph, J. Combi. Theory, Ser. B 27, pp. 320–331, 1979.

[3] F. R. K. Chung, F. Thomson Leighton and A. L. Rosenberg: Embedding Graphs in Books: A Layout Problem with Applications to VLSI Design, SIAM J. Alg. Disc. Math. Vol. 8, No. 1, 1987.

[4] Z. Galil, R. Kannan, E. Szemeredi: "On non-trivial seperations for k-page graphs and simulations by nondeterministic one-tape Turning machines," 18th ACM Symp. on Theory of Computing, pp. 39-49, 1986.

[5] F. T. Leighton: "New Lower Bound Techniques for VLSI," Math. Systems Theory 17, pp. 47-70, 1984.

[6] T. T. Trarng, M. Markek-Sadowska, and E. S. Kuh: An Efficient Single-Row Routing Algorithm, IEEE Tran. Computer-aided Design, Vol. CAD-3, No. 3, 1984.

[7] A. T. White: Graphs, Groups and Surfaces, pp. 59. *North-Holland*, 1984.

[8] M. Yannakakis: Embedding Planar Graphs in Four Pages, J. Computer and System Sciences 38, pp. 36-67, 1989.

Dominance Drawings of Bipartite Graphs *

Hossam ElGindy, † Michael Houle, †Bill Lenhart, ‡ Mirka Miller, † David Rappaport, § and Sue Whitesides ¶

A partial order P of a finite set X is a transitive and non-reflexive binary relation on X. A partial order can be represented by a *transitive digraph* G on the elements of X. The *dimension* $d(P)$ of a partial order P is the minimum number of linear orders whose intersection is P [3]. There is a direct interpretation of $d(P)$ as it pertains to its associated graph. We can use vectors $\mathbf{x} = (x_1, x_2, ...x_k)$ to represent each vertex x of G, so that $x_i \leq y_i, i = 1, 2, ..., k$, (with strict inequality in at least one coordinate) if and only if y is reachable from x in G. Such an assignment of coordinates to vertices s called a *dominance drawing*, because all edges in the graph (and transitive closure) are geometrically characterized by the dominance relation [6]. We use the notation $d(G)$ to denote the dominance drawing dimension of the graph G.

In [7] it is proved that deciding whether $d(P) \leq 3$ is NP-complete. A characterization in [3] of partial orders of dimension two or less leads to a polynomial time recognition algorithm. Thus, let $\overline{G}$ denote the complement of an undirected version of the transitive digraph representing P. Then $d(G) \leq 2$ if and only if $\overline{G}$ is *transitively orientable*, that is, the edges of $\overline{G}$ can be oriented to obtain a transitive digraph. Transitive orientability can be tested in $O(\delta K)$ time, where δ denotes the maximum vertex degree and K the number of edges in the graph, [4] [5].

We present an algorithm that obtains a two dimensional dominance drawing of a transitive bipartite graph whenever such a drawing exists. The running time of the algorithm is proportional to the size of the input and is thus optimal. This compares favourably with the complexity of the algorithm in [4] [5]. Let $G = (S, T, E)$ be a transitive bipartite graph with $S = (s_1, s_2, \ldots, s_{|S|})$ a set of sources, and $T = (t_1, t_2, \ldots, t_{|T|})$ a set of sinks. We use $N(s) = \{t : (s_i, t_j) \in E\}$ to denote the *neighbourhood* of s. Let $\mathcal{N} = \{N(s) : s \in S\}$, and let $\mathcal{M} = \{N(s) - N(w) : s, w \in S, N(w) \subset N(s)\}$, then $\mathcal{I}(S,T) = \mathcal{N} \cup \mathcal{M}$. Let $\Pi(\mathcal{I}(S,T))$ denote the collection of all permutations of T, π, such that members of each subset $I \in \mathcal{I}(S,T)$ are contiguous in π. Our algorithm to obtain a two dimensional dominance drawing of the graph G is based on the following characterization of two dimensional transitive bipartite graphs.

Theorem 1 *The dimension of $G = (S, T, E)$ is less than or equal to two if and only if $\Pi(\mathcal{I}(S,T))$ is not empty.*

*Part of the work was carried out when the authors were participants of the Workshop on Layout and Optimal Path Problems at Hawks Nest, Australia, sponsored by the Department of Computer Science, The University of Newcastle, New South Wales, Australia

†Department of Computer Science The University of Newcastle, New South Wales, AUSTRALIA.

‡Department of Computer Science Williams College, Williamstown, USA.

§Department of Computing and Information Science Queen's University, Kingston, Ontario, K7L 3N6, CANADA.

¶School of Computer Science McGill University Montreal, Quebec, CANADA.

Consider a set X and a set of subsets of X, Ξ. Booth and Leuker [2] present the so called PQ-tree algorithms that can be used to determine the family of permutations of X so that every subset $\xi \in \Xi$ is contiguous in the family of permutations. The resulting computational complexity is linear in the size of the input. Thus it appears that an expedient solution to our problem is to compute the set $\mathcal{M}$ and subsequently the set $\mathcal{I}(S,T)$ and apply the PQ-tree algorithm. However, the size of $\mathcal{M}$ may be $O(|S|^2)$. This problem is not insurmountable as we can restrict our attention to a linear sized subset of $\mathcal{M}$. Consider the case where we have a maximal sequence $N(s_1) \subset N(s_2) \subset \cdots \subset N(s_k)$, then we only need to consider the set $N(s_k) - N(s_1)$. However, the task of computing this reduced subset of $\mathcal{M}$ approaches the conceptual difficulty of presenting a new approach without using PQ-trees. We present a new algorithm and skirt the problem of computing a reduced subset of $\mathcal{M}$. The data structure we use to represent the family of permutations $\Pi(\mathcal{I}(S,T))$ is influenced by the PQ-tree, but it is specially tailored for this problem and is much simpler.

We define a *boxlist* of a set T, $\mathcal{B}(T)$, as the empty list, or a linked list consisting of one or more boxes, where each box contains a subset of T, and the boxes form an exact cover of T, that is, the union of the boxes in the boxlist is T and each element of T appears in exactly one box. If we fix intra-box ordering of elements then a rear to front, or front to rear, traversal of $\mathcal{B}(T)$ corresponds to a permutation of T. A permutation π of T is *consistent* with $\mathcal{B}(T)$ if the elements within boxes can be ordered so that a traversal of $\mathcal{B}(T)$ is equal to π. Let $F(\mathcal{B}(T))$ be used to denote the set of all permutations that are consistent with $\mathcal{B}(T)$. Given a graph $G = (S,T,E)$ our algorithm begins with a boxlist representing all permutations of T, that is, the boxlist consists of exactly one box that contains T itself. If the graph has a two dimensional dominance drawing then the algorithm exits with a non-empty boxlist such that $F(\mathcal{B}(T)) = \Pi(\mathcal{I}(S,T))$ otherwise the algorithm exits with $\mathcal{B}(T) = \emptyset$, an empty list.

The principle operation performed on a boxlist is to add *constraints* to $\mathcal{B}(T)$ that are associated with a neighbourhood of a source, $N(s)$. The constraints are substrings, or *intervals* within the permutations of $\Pi(\mathcal{I}(S,T))$. Thus $\mathcal{B}(T)$ is constrained so that the interval associated with $N(s)$ will be contiguous in all permutations of $F(\mathcal{B}(T))$. We show that the adding of constraints can be scheduled so that it is easy to check whether an interval is contiguous within $\mathcal{B}(T)$, and that $N(s) - N(w)$ is contiguous for all w such that $N(w) \subset N(s)$. We maintain an overall linear complexity by avoiding explicit sorting.

Our algorithm development is summarized in the following theorem.

Theorem 2 *Given a connected transitive bipartite graph $G = (S,T,E)$, our algorithm returns $F(\mathcal{B}(T)) = \Pi(\mathcal{I}(S,T))$. The algorithm can be implemented to run in $O(|S| + |T| + |E|)$ time and space, and this is within a constant multiple of optimal.*

References

[1] G. Di Battista, W.-P. Liu, and I. Rival, *Bipartite Graphs, Upward Drawings, and Planarity,* Information Processing Letters, vol. 36 (1990) pp. 317-322.

[2] K. Booth and G. Leuker, *Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms* J. Comput. and Sys. Sci., 13 (1976) pp. 335-379.

[3] B. Dushnik and E. W. Miller, *Partially ordered sets,* Amer. J. Math., 63 (1941) pp. 600-610.

[4] M. Golumbic, Algorithmic Graph Theory and Perfect Graphs, Academic Press (1980).

[5] A. Pnueli, A Lempel, and S. Even, *Transitive orientation of graphs and identification of permutation graphs,* Canad. J. Math., 23 (1971) pp. 160-175.

[6] R. Tamassia, *Drawing algorithms for planar st-graphs* Australasian Journal of Combinatorics, vol. 2 (1990).

[7] M. Yannakakis, *The complexity of the partial order dimension problem,* SIAM J. Alg. and Disc. Meth., Vol. 3, No. 3 (1982) pp. 351-358.

Computing the Overlay of Regular Planar Subdivisions in Linear Time

Ulrich Finke and Klaus Hinrichs*

A *planar subdivision* is the embedding of a *planar graph* in the plane and therefore determines a partitioning of the plane [4]. It is an open problem, whether the overlay of two simply connected subdivisions can be processed in linear time and space [1]. A first step to solve this problem is the overlay algorithm for convex subdivisions which has the desired complexity [3]. Generalizations of this result are of theoretical and practical importance. We propose an topological sweepline algorithm which solves the problem of overlaying regular subdivisions in linear time and space. A regular subdivision is simply connected, the embedding of each edge curve must be a function in x, and each vertex v of the planar graph has at least one incoming edge from the left and one outgoing edge to the right.

The sweepline in our algorithm divides the overlay subdivision into two parts. The left part is the correctly processed overlay and the right part contains the unprocessed elements of the subdivisions. All edges between the left and right part intersect the topological sweepline. The basic idea of our algorithm is to handle only those edges that represent the seams between the subdivisions on the sweepline. By performing local sweepline transactions the algorithm sews up the subdivisions to generate the overlay result. The locality of the transactions makes it possible to parallelize the algorithm easily.

Algorithms for computing the overlay of planar subdivision are of great practical importance in geographic information systems and computational geometry [2, 5].

References

[1] B. Chazelle: Computational Geometry for the Gourmet - Old Fare and New Dishes, ICALP, 1991, 686 - 696.

[2] A. U. Frank: Overlay Processing in Spatial Information Systems, Proc. 8th Int. Symposium on Computer-Assisted Cartography (AUTO-CARTO 8), 1987, 16-31.

*FB 15, Informatik, Westfälische Wilhelms-Universität Einsteinstr. 62, D - 48149 Münster, Germany finke,hinrichs@math.uni-muenster.de

[3] L. J. Guibas, R. Seidel: Computing Convolutions by Reciprocal Search, 2nd ACM Symposium on Computational Geometry, 1986, 90 - 99.

[4] F. P. Preparata, M. I. Shamos: Computational Geometry - An Introduction. Springer-Verlag, New York, 1985.

[5] C. D. Tomlin: Geographic Information Systems and Cartographic Modeling, Prentice Hall, Englewood Cliffs, NJ, 1990.

Generation of Random Planar Maps

Alain Denise *

Methods of random generation are useful tools to study some properties of combinatorial structures. As regards graphs, such methods are efficient to verify or formulate conjectures, especially when the exhaustive generation of all the graphs which are to be studied would be unreasonable. These methods also allow to evaluate the performances of algorithms on such structures. Moreover, to increase one's own knowledge on some class of graphs, it is useful to be able to generate and to display these configurations. Thus, procedures of random generation are included in softwares of manipulation of graphs, which are used for research and teaching of graph theory and discrete mathematics [3]. The uniform generation of random graphs has been well studied for a few years, and efficient algorithms exist for some particular classes. See for exemple works of Tinhofer [7], Dixon and Wilf [5], Wormald [8], Jerrum and Sinclair [6].

The matter of our work is the uniform generation of random rooted planar maps with n edges. A planar map is the projection of a planar connected graph on a plane surface. A map is rooted if a vertex and an edge adjacent to it are distinguished. By using the encoding of planar maps due to Cori and Vauquelin [4], we reduce the problem to a problem of generation of words of a language close to the language of parenthesis systems. Then we use a rejection algorithm inspired by the methods of Barcucci, Pinzani and Sprugnoli [1, 2], in order to generate these words. We prove that the average complexity is $O(n^2)$, and we conjecture that it is $O(n\sqrt{n})$.

References

[1] E. Barcucci, R. Pinzani, and R. Sprugnoli. Génération aléatoire des animaux dirigés. In J. G. Penaud J. Labelle, editor, *Actes de l'Atelier Franco-Québécois de Combinatoire, publi LaCIM 10.* Université du Québec à Montréal, 1991.

[2] E. Barcucci, R. Pinzani, and R. Sprugnoli. Génération aléatoire de chemins sous-diagonaux. In C. Reutenauer P. Leroux, editor, *Actes du 4ème Colloque Séries Formelles et Combinatoire Algébrique, publi LaCIM 11.* Université du Québec à Montréal, 1992.

[3] O. Baudon. *Cabri-graphes, un cahier de brouillon interactif pour la théorie des graphes.* PhD thesis, Université Joseph Fourier, Grenoble, 1990.

*LaBRI, Université Bordeaux I, 33405 Talence Cedex, France. denise@labri.u-bordeaux.fr

[4] R. Cori and B. Vauquelin. Planar maps are well labeled trees. *Can. J. Math.*, 33(5):1023–1042, 1981.

[5] H. S. Wilf and J. D. Dixon. The random selection of unlabeled graphs. *J. Algorithms*, 4:205–213, 1983.

[6] M. Jerrum and A. Sinclair. Fast uniform generation of regular graphs. *TCS*, 73:91–100, 1990.

[7] G. Tinhofer. On the use of almost sure graph properties. *Lecture Notes in Computer Science*, 100, 1980.

[8] N. C. Wormald. Generating random regular graphs. *J. Algorithms*, 5:247–280, 1984.

Symmetric Drawings of Graphs

Joseph Manning*

Introduction

Perhaps one of the most important criteria for producing visually-informative drawings of abstract graphs is the display of axial and/or rotational symmetry, collectively known as *geometric symmetry*. Its importance stems from the fact that, given a symmetric drawing, an understanding of the entire graph can be built up from that of a smaller subgraph, replicated a number of times. This paper reviews several results on the definition, detection, and display of geometric symmetry.

Geometric symmetry in graphs may be defined by using either a geometric or an algebraic formulation. Geometrically, a graph is said to possess a particular symmetry if there exists some drawing of the graph which displays that symmetry. Note, however, that geometric symmetry is an inherent property of the abstract graph itself, rather than of any individual drawing. Algebraically, a geometric symmetry may be defined as an automorphism of the graph which satisfies certain conditions. Both definitions are equivalent. While all of the results below were obtained from the geometric definition, it appears that the algebraic definition may hold the greatest promise in attempting to expand these results to broader classes of graphs.

The fundamental problem of determining if a general abstract graph possesses any geometric symmetry, along with several variations, are all $\mathcal{NP}$-complete [5, 6]. Accordingly, the current research has focused on symmetry in planar graphs, since these constitute an important subclass of general graphs and frequently admit efficient solutions to otherwise intractable problems.

*Vassar College, U.S.A. manning@cs.vassar.edu

Algorithms

Optimal, linear-time algorithms, outlined below, have been developed for detecting and displaying both axial and rotational symmetries in the following classes of (planar) graphs:

- *Trees* [3, 6]: A tree ("free tree") is a connected acyclic graph. All symmetries of a tree must keep its center fixed. (A *center* of a tree is any vertex whose maximum distance from any leaf is minimized; every tree has either one center, or two adjacent centers; the latter case may be reduced to the former by introducing a new vertex on the edge joining the two centers.) Removing the center from a tree divides the remainder of the tree into a number of subtrees, and any symmetry of the overall tree must permute these among themselves, mapping subtrees to isomorphic subtrees. Using a variation of the linear-time tree-isomorphism test [1, Mp84], these subtrees are partitioned into isomorphism classes, from which the geometric symmetries are subsequently determined. A radial drawing of the tree, which displays these symmetries, is then constructed. Since it is impossible, in general, to display all of its axial symmetries in a single drawing of a given tree, the algorithm instead determines the maximum number of simultaneously-displayable axial symmetries, and constructs the corresponding "most symmetric" drawing.
- *Outerplanar Graphs* [4, 6]: An outerplanar graph is one which can be drawn in the plane with no edge crossings and with all vertices on the outer face. Every biconnected outerplanar graph has a unique Hamilton cycle, which may be found in linear time. By traversing its Hamilton cycle, the graph is transformed into a string, in such a way that geometric symmetries of the graph correspond to certain "symmetries" of the string. These, in turn, are found using an efficient pattern-matching algorithm [2], from which the symmetries of the graph are then recovered. By contrast with the situation for trees, all geometric symmetries of a biconnected outerplanar graph may be displayed in a single drawing, which the algorithm then constructs. For non-biconnected outerplanar graphs, geometric symmetries are enumerated by using a combination of the above algorithm, applied to the biconnected components, with the algorithm for trees, applied to the block-cutvertex tree, while a similar combination of drawing techniques is used to construct symmetric drawings.
- *Plane Embeddings of Planar Graphs* [6]: A planar graph is one which can be drawn in the plane with no edge crossings, and a plane embedding merely lists the order of edges emanating from each vertex, without specifying either the coordinates of the vertices or the shapes of the edges. A planar graph, with such an embedding, is first transformed into a number of "concentric" biconnected outerplanar levels, and the geometric symmetries of these levels are then found using the previous algorithm and "intersected" to give the symmetries of the original graph. This algorithm has particular relevance to drawing triconnected planar graphs, whose plane embeddings are unique up to the choice of outer face.

Many of the algorithms have been implemented, and test runs have shown that their optimal theoretical time complexities do indeed translate into fast practical algorithms. Running on a mid-range workstation, a graph with up to one hundred vertices can be processed almost instantaneously.

Extensions

Perhaps the most important challenge lies in extending these results to the entire class of planar graphs. An easier extension might be to series-parallel graphs, which form another proper subclass of planar graphs.

The incidence of geometric symmetry in graphs appears to decrease as the size of the graph increases. For example, while 58% of 10-vertex trees have at least one axial symmetry, only 16% of 15-vertex trees do. It appears useful to relax the requirement of strict symmetry and instead investigate using "near-symmetry" as a drawing criterion. An even more far-reaching generalization would be to explore the construction of (straight-edge) drawings in which the number of distinct edge lengths is minimized.

References

[1] A. Aho, J. Hopcroft, J. Ullman. "The Design and Analysis of Computer Algorithms". Addison-Wesley, 1974.

[2] D. Knuth, J. Morris, V. Pratt. "Fast Pattern Matching in Strings". *SIAM Journal on Computing 6*(2):323–350, Jun 1977.

[3] J. Manning, M. Atallah. "Fast Detection and Display of Symmetry in Trees". *Congressus Numerantium 64*:159–169, Nov 1988. Also available as Technical Report CSD-TR-562, Department of Computer Sciences, Purdue University, Dec 1985.

[4] J. Manning, M. Atallah. "Fast Detection and Display of Symmetry in Outerplanar Graphs". *Discrete Applied Mathematics 39*(1):13–35, Aug 1992. Also available as Technical Report CSD-TR-964, Department of Computer Sciences, Purdue University, Mar 1990.

[5] J. Manning. "Computational Complexity of Geometric Symmetry Detection in Graphs". *Lecture Notes in Computer Science 507*:1–7, Springer-Verlag, Jun 1991. Also available as Technical Report CSC-90-1, Department of Computer Science, University of Missouri–Rolla, Jan 1990.

[6] J. Manning. "Geometric Symmetry in Graphs". Ph.D. Thesis, Department of Computer Sciences, Purdue University, Dec 1990.

Recognizing Symmetric Graphs

Tomaž Pisanski*

Usually we can deal with graphs if they are small and we are able to grasp their pictorial representation. If a graph is stored and the information of its construction is lost the problem is how to recognize the graph: how to find its optimal or near-optimal construction. For a human the drawing of a graph may represent a way of recognizing the graph.

*IMFM, Department of Theoretical Computer Science, University of Ljubljana, Jadranska 19, 61111 Ljubljana, Slovenia. tomaz.pisanski@uni-lj.si

We implemented a series of algorithms for automatic drawing of graphs. The first method uses the idea of *spring embedding* by P. Eades [Ead84] that comes in several variants. Graduate student Danica Dolničar wrote a Pascal program that compares the algorithm of Kamada and Kawai [KK89] to the method of Fruchterman and Reingold [FR91].

In the second one we experimented with eigenvectors. We noticed that the 2nd, 3rd and 4th eigenvector can be used as the three coordinates for the vertices in the 3-dimensional space. Later we obtained some theoretical results that will be presented in a joint paper with John Shawe-Taylor.

For vertex transitive graphs we tried to use a combination in order to produce good results for large graphs. First we calculate the automorphisms of graph. We use B. McKay's Nauty to do the job. Then we select an automorphism π that has the smallest number of orbits. We contract the vertices in each orbit of π and thus obtain the factor graph. Then the factor graph is drawn using an automatic drawing algorithm. Finally the orbits are blown out so that the vertices are put on the cycles in the order specified by the permutation π. For instance, the Coxeter graph on 28 vertices that is otherwise hard to recognize is drawn in the familiar Y shape. If all the orbits are of the same size, say k, another drawing approach may be taken. The vertices of the k copies of the factor graph are placed on a circle in order to display rotational symmetry. Then the edges of the original graph are drawn as straight lines. The idea of displaying symmetry in graph drawing is certainly not new. The reader is referred to [2], [1], [6], and [7] for further information about the research on this topic.

It should be noted that computation-intensive algorithms are being programmed by students and researchers at IMFM in computer languages such as Pascal and C, however they are all bundled in a Mathematica [Wat89] package called VEGA. The whole system will become available for non-profit use in 1994.

References

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

[2] G. Di Battista, P. Eades, R. Tamassia, I. G. Tollis, "Algorithms for Drawing Graphs: an Annotated Bibliography", Preprint June 1993.

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

[4] T.M.J. Fruchterman and E.M. Reingold, "Graph Drawing by Force-directed Placement," Software-Practice and Experience , vol. **21** pp. 1129-1164, 1991.

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

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

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

[8] S. Wolfram, "Mathematica," Addison-Wesley, Redwood City, CA, 1991.

Algorithmic and Declarative Approaches to Aesthetic Layout

Peter Eades and Tao Lin*

Aesthetics for graph layout can be divided into three categories:

- *Global criteria*, such as minimizing the number of edge crossings, or maximizing symmetry.
- *Correctness criteria*, such as placing the employer above the employees in an organization tree drawing.
- *Preferred criteria*, which express the preferences of a specific user at a specific time. These criteria may include placing a particular node in the centre of the page, or using a particular aspect ration.

We can divide implementations of layout functions into two general categories: those with an algorithmic approach and those with a declarative approach.

The **algorithmic approach** is well documented in the survey [2]. This approach concentrates on achieving global criteria. Typically, the approach ignores the semantic or syntactic meaning of a specific diagram and only uses graph theoretic structure.

In a layout algorithm, the aesthetics are hard coded into the implementation of the function. It is not easy to change the requirements for a layout algorithm at run time. Such layout algorithms are not flexible, cannot cope with preferred criteria, and normally can only cope with a few fixed correctness criteria.

A **declarative layout function** handles the layout according to a flexible set of requirements specified by end-users or interface designers, even at run time. A system which uses declarative layout creation function has two components: an editor through which user can specify aesthetics and a mechanism for creating the layout. The aesthetics may be presented as constraints, rules, or parameters for a cost function. There are several mechanisms used in the declarative approach, such as constraint solvers, genetic systems, rule based systems, and simulated annealing.

The expressive power of constraints and rules ensure that a wide variety of requirements can be specified; thus the declarative approach maximizes flexibility.

However, the declarative approach has significant problems. It is difficult to choose the rules or constraints (it is difficult to foresee the effects of a constraint or rule, even in a moderately sized system). Implementations of declarative functions are very slow, and sometimes (due to their heuristic nature) do not achieve their aim, even there is a layout which satisfies relevant requirements. In particular, the declarative approach has some difficulty handling global aesthetics.

We conclude that neither algorithmic nor declarative approaches are suitable for sole adoption in graphic user interfaces. However, by integrating the approaches, one can use the advantages of one to compensate the disadvantages of the other.

If a layout algorithm is built on top of a declarative system, the integrated system seems very slow [3]. Declarative techniques are used successfully on top of algorithms in [TBB88, 1];

*Department of Computer Science, University of Newcastle, Newcastle, Australia. eades@cs.newcastle.edu.au. frank@cs.newcastle.edu.au.

however, in these cases the declarative techniques are tightly bound to the algorithms. We believe that an effective approach is to loosely tie some declarative functions on top of a layout algorithm. Briefly, this may be achieved by exploiting the nondeterminism in layout algorithms.

The integrated system may be used as follows:

- The *interface modeler* creates a generic toolbox of layout algorithms covering a wide range of global aesthetics. The algorithms leave points of nondeterminism to be exploited by constraints.
- The *interface designer* chooses a specific set of algorithms for a specific application. The algorithms are customized by specifying constraints so that they satisfy the correctness criteria of the application.
- The *end user* specifies further constraints to achieve preferred criteria.

This approach satisfies the same set of the global criteria as the underlying layout algorithm. However, the integrated approach is sufficiently flexible to support a broad range of correctness and preferred criteria.

We give examples of this approach using tree drawings.

References

[1] Karl-Friedrich Böhringer and Frances Newbery Paulisch. Using constraints to achieve stability in automatic graph layout algorithms. In *Proceedings of ACM/SIGCHI*, pages 43–51, 1990.

[2] P. Eades and R. Tamassia. Algorithms for drawing graphs: an annotated bibliography. Technical report, Department of Computer Science, Brown University, 1989. Accepted to the *Networks*.

[3] E.B. Messinger. *Automatic Layout of Large Directed Graphs*. PhD dissertation, Department of Computer Science, University of Washington, 1988. Published as Technical Report No. 88-07-08.

[4] Roberto Tamassia, Giuseppe Di Battista, and Carlo Batini. Automatic graph drawing and readability of diagrams. *IEEE Transactions on Systems, Man and Cybernetics*, 18(1):61–79, January/February 1988.

A Visual Approach to Graph Drawing *

Isabel F. Cruz,† Roberto Tamassia,† and Pascal Van Hentenryck†

This abstract describes research in progress on a new technique for the visual specification of constraints in graph drawing systems.

Work in graph drawing has traditionally focused on *algorithmic* approaches, where the layout of the graph is generated according to a prespecified set of general rules or aesthetic criteria (such as planarity or area minimization) that are embodied in an algorithm. Perhaps the most sophisticated graph drawing system based on the algorithmic approach is the one developed by Di Battista *et al.* [BBL92], which maintains a large database of graph drawing algorithms and is able to select the one best suited to the needs of the user.

The algorithmic approach is computationally efficient, however, it does not naturally support *constraints*, i.e., requirements that the user may want to impose on the drawing of a *specific* graph (e.g., clustering or aligning a given set of vertices). Previous work by Tamassia *et al.* [TBB88] has shown the importance of satisfying constraints in graph drawing systems, and has demonstrated that a limited constraint satisfaction capability can be added to an existing drawing algorithm. Recently, several attempts have been made at developing languages for the specification of constraints and at devising techniques for graph drawing based on the resolution of systems of constraints [Kam89, Mar91, HM90].

Current constraint-based systems have three major drawbacks:

- The specification of constraints is made through a detailed enumeration of facts from a fixed set of predicates, expressed in Prolog [Kam89] or with a set-theoretic notation [Mar91].
- Natural requirements, such as planarity, need complicated constraints to be expressed.
- General constraint-solving systems are computationally inefficient [HM90].

The above discussion indicates the need for a language to specify constraints that reconciles expressiveness with efficiency.

We believe that visual languages could provide a natural and user-friendly way to express the layout of a graph. For this purpose, we plan to design a variation of DOODLE, a visual language for the specification of the display of facts in an object-oriented database [Cru92, Cru93]. We envision the following goals, which differentiate our work from [Mar91] and [Kam89]:

- Visual specification of layout constraints: the user should not have to type a long list of textual specifications.
- Extensibility: the user should not be limited to a prespecified set of primitives.
- Flexibility: the user should not have to give precise geometric specifications, such as exact coordinates or precise geometric relations.

*Research supported in part by the National Science Foundation under grant CCR-9007851, by the U.S. Army Research Office under grant DAAL03-91-G-0035, and by the Office of Naval Research and the Advanced Research Projects Agency under contract N00014-91-J-4052, ARPA order 8225.

†Department of Computer Science, Brown University, Providence, RI 02912-1910. {ifc,rt,pvh}@cs.brown.edu

In addition to constraints, the visual language should also be able to express aesthetic criteria associated with optimization problems (e.g., crossing or area minimization) and to identify general drawing standards (e.g., layered drawing or upward drawing). Recent work by Eades and Lin [EL93] has similar objectives, but is not based on a visual specification.

For efficiency reasons, we envision using our visual language within a graph drawing system similar to Diagram Server [BGST90]. A crucial component of this system is a compiler that translates the visual specifications into a drawing algorithm synthesised from a database of drawing algorithms. The algorithms database will contain both polynomial- and exponential-time algorithms. The main purpose of the drawing compiler is to deduce from the specifications a combination of algorithms that solves the layout problem as efficiently as possible. The work in [BBL92] is particularly interesting in this context.

References

[BBL92] P. Bertolazzi, G. Di Battista, and G. Liotta. Parametric Graph Drawing. Technical Report 6/67, Consiglio Nazionale delle Ricerche, Rome, Italy, July 1992.

[BGST90] G. Di Battista, A. Giammarco, G. Santucci, and R. Tamassia. The Architecture of Diagram Server. In *Proc. of IEEE Workshop on Visual Languages*, 1990.

[Cru92] Isabel F. Cruz. DOODLE: A Visual Language for Object-Oriented Databases. In *ACM-SIGMOD Intl. Conf. on Management of Data*, pages 71–80, 1992.

[Cru93] Isabel F. Cruz. Using a Visual Constraint Language for Data Display Specification. In Paris Kanellakis, Jean-Louis Lassez, and Vijay Saraswat, editors, *First Workshop on Principles and Practice of Constraint Programming*, 1993.

[EL93] Peter Eades and Tao Lin. Algorithmic and Declarative Approaches to Aesthetic Layout. In *Proc. of Graph Drawing '93*, 1993.

[HM90] Richard Helm and Kim Marriott. Declarative Specification of Visual Languages. In *Proc. IEEE Workshop on Visual Languages*, 1990.

[Kam89] Tomihisa Kamada. *Visualizing Abstract Objects and Relations – A Constraint-Based Approach.* World Scientific, Singapore, 1989.

[Mar91] Joe Marks. A Formal Specification for Network Diagrams That Facilitates Automated Design. *Journal of Visual Languages and Computing*, 2:395–414, 1991.

[TBB88] R. Tamassia, G. Di Battista, and C. Batini. Automatic Graph Drawing and Readability of Diagrams. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-18(1):10–21, 1988.

Layout of Trees with Attribute Graph Grammars

Gaby Zinßmeister *

Graph grammars are a formalism for syntactically describing classes of graphs. Their production rules are used for graph rewriting to implement graph manipulations. There are a variety of application areas ranging from software development environments and compiler construction over pattern recognition and development of biological cell layers to specification of concurrent systems.

The central idea of our approach to layout graphs is viewing layout algorithms as attribute evaluators of attribute graph grammars, thus a layout algorithm is an attribute scheme plus an attribute evaluator. The main advantage is that different layouts can be specified simply with different attribute schemes.

We have modelled three widely used tree layout algorithms ([6, 7, 9]) with attribute graph grammars (AGG). A graph grammar (GG) for arbitrary trees is described. The three different layout algorithms are specified by different attribute schemes of the same grammar. We illustrate our approach with the Moen algorithm [6]. All three algorithms will be presented in the full paper.

Attribute Graph Grammars (AGG)

GGs are a generalization of string grammars which are well known from formal language theory. Kreowski and Rozenberg give an excellent survey of graph grammars [4, 5]. For our purposes we use so called context free node label controlled GGs (CFNLC) [2]. The productions of such a GG consist of one nonterminal node as the left-hand side of the rule, a graph over nonterminal and terminal nodes as the right-hand side and an embedding specification. A derivation step consists of choosing one occurrence of the left-hand side nonterminal A in the host graph H (which is the analogue of a sentential form), removing that instance, adding the right-hand side graph R and connecting the remainder of H to R following the embedding specification. The embedding specification describes which nodes of the neighbourhood of the instance of A in H are to be connected with which nodes of R. Thus edges may be deleted or added or their orientation may be inverted. The language produced by a GG is an (infinite) set of graphs.

The AGG approach used in this paper is a generalization of Knuth's attribute (string) grammars. Synthesized or inherited attributes are associated to nodes and attribute evaluation rules to the GG productions. Attribute evaluators may be constructed analogously (see [8]).For special GGs efficient (polynomial time) parsers can be constructed [3] and attribute evalutor generators have been implemented [8].

The GG for Arbitrary Trees

The GG for trees is defined as follows:
Let $T = (\Sigma_N = \{F, S\}, \Sigma_T = \{t\}, \Sigma_E = \{x\}, P, S = F)$ with the productions as in fig. 1.

*Wilhelm-Schickard-Institut, Sand 13, D-72076 Tübingen. zinssmei@informatik.uni-tuebingen.de

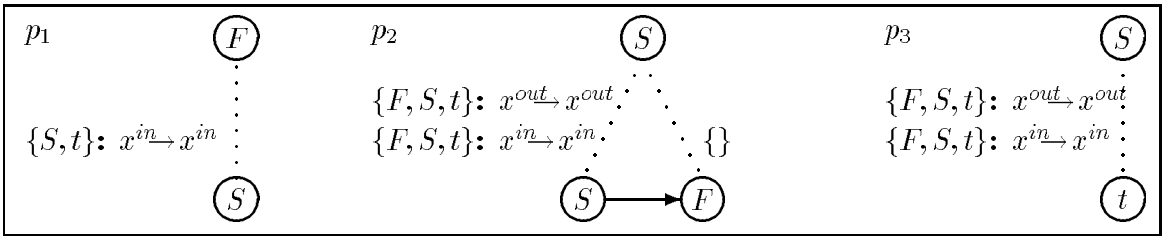


Figure 1: Productions P of T

S nodes are the 'generic' nodes with which we can add an arbitrary number of children to a treenode, whereas F nodes represent a fully derived subtree.

The upper nonterminal nodes are the left hand sides, the lower graphs with terminal and nonterminal nodes and solid edges are the right hand sides of the productions. The dotted lines indicate the replacement step. The mark $\{F,S,t\} : x^{in} \rightarrow x^{in}$ means that x-labelled edges from F-, S- and t-nodes to the left hand side node, are replaced by x-labelled edges from the same sources to the right hand side node specified by the dotted line (x^{out} denotes the reverse orientation of edges.) The empty set {} at a dotted line denotes that the target node is not connected to the host graph.

The terminal alphabet can obviously be extended, but is reduced to one node type for simplicity. There is only one edge label, so we can think about the terminal tree as one with unlabelled edges.

The Moen Algorithm as AGG

The main idea in attributing is to push information about subtree contours from bottom to top in the derivation tree (synthesized attributes), joining subtrees in the S-nodes and adjusting the root over the subtrees in the F-nodes. The relative position of a node to its sibling is calculated in this pass too. Afterwards in a second pass the missing relative positions of the first child of each subtree are pushed downwards in the derivation tree (inherited attributes). Absolute coordinates may be assigned to nodes in this second pass as well, but are omitted here for simplicity. Figure 6 lists the attributes and figure 6 shows the complete attribution. The functions written in italic are taken from the original Moen algorithm, except for different use of reference parameters and global values. The attributes are evaluable with a two pass left to right evaluation on the derivation tree.

Conclusion

The only other approach we know to drawing graphs using graph grammars is published by Brandenburg [1]. He augments graph grammar productions by placement rules in form of `left-of, above-of` constraints between *rhs*-nodes, a bounding box around the *rhs* and some connection points on that bounding box. This is well suited for graphs where replacing a (nonterminal) node results pushing the rest of the graph in x- and y- dimension to get space for the *rhs*-nodes at the (relative) old place of the *lhs*-node. Such graphs are for example syntax diagrams or series parallel graphs. But with the tree graph grammar above constraints in the sense Brandenburg will not lead to a reasonable layouts, because it is not

Attributes	Symbol	Type	Explanation
rootdims (syn)	F, S	integers	width, height and border of the root node
contour (syn)	F	polyline	contour around the tree derived from F
contoursubtrees (syn)	S	polyline	contour around the subtrees derived from S
lastsubtreeheight (syn)	S	integer	height incl. border of the subtreeroot derived last from S
sum (syn)	S	integer	sum of heights of the subtrees
dims (syn)	t	integers	width, height, border of the terminal node
offset (inh)	S, F, t	position	relative x-, y-coordinates
heightsubtrees (inh)	S	integer	height of subtrees derived from S and its siblings

Figure 2: Attributes for the Moen Algorithm

possible to define `left-of` relations between siblings. The reason is that sibling nodes are not related by edges and not produced within the same production, so no relation can be stated.

As further work we plan to develop a CFNLC GG for directed acyclic graphs (DAGs) and attribute it with some of the current layout algorithms for DAGs. Even though the literature ([3, 8]) describes generation of polynomial time parsers for subclasses of GGs and generation of attribute evalutors for AGGs, no running implementation is available. For that reason we are implementing a parser and an attribute evaluator for *TGG*. For interactive environments, where graphs are manipulated, incremental layout of graphs is necessary. This could be done by applying incremental attribute evaluation to the attribute graph grammar approach.

References

[1] F. J. Brandenburg. Layout Graph Grammars: the Placement Approach. In H. Ehrig, H.-J. Kreowski, and G. Rozenberg, editors, *Graph-Grammars and Their Application to Computer Science, 4^{th} Int. Workshop, Bremen, Germany, 1990*, LNCS 532, pages 144–156. Springer, 1991.

[2] D. Janssens and G. Rozenberg. Graph Grammars with Node-Label Controlled Rewriting And Embedding. In H. Ehrig, M. Nagl, and G. Rozenberg, editors, *Graph-Grammars and Their Application to Computer Science, 2^{nd} Int. Workshop, Haus Ohrbeck, Germany, 1982*, LNCS 153, pages 186–205. Springer, 1983.

[3] M. Kaul. Syntaxanalyse von Graphen bei Präzedenz-Graph-Grammatiken. Technical Report MIP-8610, Univ. Passau, 1986. (dissertation).

[4] H.-J. Kreowski and G. Rozenberg. On Structured Graph Grammars I. *Information Sciences*, 52:185–210, 1990.

[5] H.-J. Kreowski and G. Rozenberg. On Structured Graph Grammars II. *Information Sciences*, 52:221–246, 1990.

[6] S. Moen. Drawing Dynamic Trees. *IEEE Software*, 7(4):21–28, July 1990.

[7] E. M. Reingold and J. S. Tilford. Tidier Drawings of Trees. *IEEE Trans. Softw. Eng.*, SE-7(2):223–228, Mar. 1981.

[8] A. Schütte. *Spezifikation und Generierung von Übersetzern für Graph–Sprachen durch attributierte Graph–Grammatiken.* Reihe Informatik. EXpress Edition, Berlin, 1987. (dissertation).

p_1:	F.rootdims	:= S.rootdims
	F.contour	:= **IF** S.contoursubtrees = **empty THEN** *layout_leaf*(S.rootdims)
		ELSE *attach_parent*(S.contoursubtrees, S.rootdims,S.sum)
	S.offset	:= F.offset
	S.heightsubtrees	:= S.sum
p_2:	S_0.contoursubtrees	:= **IF** S_1.contoursubtrees = **empty THEN** F'.contour
		ELSE *merge1*(S_1.contoursubtrees,F'.contour)
	S_0.lastsubtreeheight	:= F'.rootdims.height + 2 * F'.rootdims.border
	S_0.rootdims	:= S_1.rootdims
	S_0.sum	:= **IF** S_1contoursubtrees = **empty THEN** S_0.lastsubtreeheight
		ELSE S_0.lastsubtreeheight + *merge2*(S_1.contoursubtrees,F.contour) + S_1.sum
	F.offset.x	:= **IF** S_1.contoursubtrees = **empty THEN**
		S_1.rootdims.border + const_parent_distance + S_1.rootdims.width
		ELSE 0
	F.offset.y	:= **IF** S_1.contoursubtrees = **empty THEN**
		(S_1.rootdims.height - S_1.heightsubtrees)/2 + S_1.rootdims.border
		ELSE *merge2*(S_1.contoursubtrees,F.contour) + S_1.lastsubtreeheight
	S_1.offset	:= S_0.offset
	S_1.heightsubtrees	:= S_0.heightsubtrees
p_3:	S.contoursubtrees	:= **empty**
	S.lastsubtreeheight	:= 0
	S.sum	:= 0
	S.rootdims	:= t.dims
	t.offset	:= S.offset

Figure 3: Attribute Scheme of T for the Moen Layout Algorithm

[9] Tree-Widget of the Athena Widget Set, X11R5 distribution of MIT, 1990.

The Display, Browsing and Filtering of Graph-trees

Sandra P. Foubister* [†] **and Colin Runciman** [†]

Context

We are writing an interpreter for a little lazy functional language. Implementation is by graph reduction [6]. The user is allowed to view, and explore, the program graph at every reduction step, or at less frequent intervals on request. The aim is to gain insight into the process of lazy graph reduction, where the order of reduction is not always intuitive. There are two main objectives: to explore the extent of sharing, and to be able to identify areas of inefficiency. The potential size and complexity of the graphs pose problems for display. This paper presents and discusses novel solutions to these problems.

Complexity

One way of simplifying the display is to avoid any crossing of arcs. There is no guarantee that a program graph will be planar – indeed, the features of a lazy language: sharing, recursion, and "knot tying" in general, make planarity unlikely. Rather than trying to display every arc in the graph, the solution being investigated is to use a spanning tree. This is enhanced with *display leaves* to represent arcs that would otherwise not be shown. Display leaves are labeled with a reference to the vertex to which they represent an arc. The problem of program *graph* display is thus limited to that of *tree* display. The special kind of tree being displayed is referred to as a *graph-tree*.

Size

The problem of size (compounded by the addition of display leaves) may be resolved in several ways: the scale of the display may be reduced, or only part of the graph may be shown at a time. In addition to these, a solution proposed here is that the size of the underlying *graph* be reduced, by grouping vertices together in *clusters*, so that the new graph has fewer vertices.

Graph-trees

The implementation of the programming environment is itself in a lazy functional language, namely Haskell [5]. In such a language one can define a *displayable graph-tree* type, `DGT`, that is convenient for subsequent display and browsing of the structure. It is parametrised on index, value and reference types. Indices uniquely identify vertices, values are vertex labels (not necessarily unique), and the reference type is the type of the display reference, typically an integer.

*Supported by the Science and Engineering Research Council of Great Britain

†Department of Computer Science, University of York, Heslington, York, YO1 5DD. {sandra,colin}@minster.york.ac.uk

```
data DGT i v r = DGT Xpos (Vertex i v r) [DGT i v r] | NoDGT
```

The `Xpos` is a provisional position on the `x` axis of the display that may be scaled to an actual `x` coordinate.
The `Vertex` is *either* a reference to a `DGT` not instantiated at this point, *or* a vertex value with its associated identifying index, and possibly a display reference.

```
data Vertex i v r = Ref (DGT i v r) | Val v i (Maybe r)
```

The list of `DGT`s within a `DGT` construction comprises a *predecessor* as well as the successors of the current `DGT`: we have a threaded structure that exploits the laziness of the defining language in its construction [1].
`NoDGT` is needed to represent the predecessor of the root of the (directed) graph. Each `DGT` has directly available sufficient information to redisplay the graph-tree with itself at the root.

Display

The requirements of the display are that it should be compact but also revealing of the structure. Various styles of presentation were considered, including the *tip-over* and *inclusion* conventions described in [4], and the possibility of showing the tree as a free tree (see [3]) (despite the existence of a root). For other purposes these may be suitable, but in our system the display of a graph-tree reflects the conventional display of applicative expressions as trees: interior vertices correspond to applications, with function and argument graphs as successors. Shared reference arcs may point back up the tree.

A modification of Vaucher's algorithm [8] is used to calculate `Xpos` entries as the `DGT` is being created. The final display is a spanning tree of the graph, with an extra node for each arc that is not part of the tree. The choice of spanning tree is determined by the order in which vertices are visited during the display routine. At present the order is that resulting from our variant of Vaucher's algorithm, but the resulting structure may not be ideal for filtering and browsing, and may not have the most satisfying appearance. However, determining an optimal spanning tree for display purposes may be infeasibly complex in our interactive setting [7].

Browsing

In addition to the main display, a *minigraph*, scaled to fit exactly onto a small window, is used as a map for browsing, as advocated by Beard and Walker [2]. The graph-tree has the shape it would have if labels were present, for concordance with the main display, but no labels are shown.

The main display is in a larger window, but on a fixed scale, so the graph-tree may have to be pruned. Arcs to vertices off the display, are truncated to form *stubs*. Clicking on the display of a vertex, or the end of a stub, or a display reference, brings the appropriate vertex to the root of the display. Clicking in the minigraph window permits the user to jump to another section of graph-tree.

Filtering

In order to reduce the number of vertices in the graph to be displayed, without violating the meaning of the original graph, the notion of a *homosemantic* graph is introduced. The idea is that a cluster of vertices with their interconnecting arcs becomes **one** vertex in a graph of clusters. This vertex inherits all the arcs from the vertices it incorporates that connect with the rest of the graph. The value of the new vertex integrates the values of its constituent vertices. The full structure of the original graph is **not** retained in the display, but the conditions under which clusters may assimilate others are defined in such a way that the graph has the same *meaning*. The implementation of such a filtering scheme raises various interesting questions about the definition of suitable "filters", and the ordering of compaction of the graph.

The system outlined above offers an effective way of observing even large and complex graphs. Our current goal is to provide users of our application with a flexible mechanism for *defining* filters, to achieve such views of the program graph as they find necessary.

References

[1] L. Allison. Circular programs and self-referential structures. *Software — Practice and Experience*, 19(2):99–111, 1989.

[2] David V. Beard and John Q. Walker II. Navigational techniques to improve the display of large two-dimensional spaces. *Behaviour and Information Technology*, 9(6):451 – 466, 1990.

[3] Peter Eades. Drawing free trees. Technical report, International institute for advanced study of social information science, Fujitsu laboratories Ltd., Japan, 1991.

[4] Peter Eades, Tao Lin, and Xuemin Lin. Two tree drawing conventions. *International Journal of Computational Geometry and Applications*, 1991.

[5] Joe Fasel, Paul Hudak, Simon Peyton Jones, and Phil Wadler. Special issue on the functional programming language Haskell. *ACM SIGPLAN Notices*, 27(5), 1992.

[6] Simon Peyton Jones and David Lester. *Implementing Functional Languages*. Prentice Hall, 1992.

[7] Kenneth J. Supowit and Edward M. Reingold. The complexity of drawing trees nicely. *Acta Informatica*, 18:377 –392, 1983.

[8] Jean G. Vaucher. Pretty-printing of trees. *Software — Practice and Experience*, 10:553–561, 1980.

A Layout Algorithm for Undirected Graphs

Daniel Tunkelang. *

We propose an algorithm for generating straight-line two dimensional layouts of undirected graphs. Our algorithm uses a combination of heuristics to obtain layouts which are near-optimal with respect to an "aesthetic" cost function. The heuristics improve on existing approaches by focusing on three aspects of the graph layout problem: computation of the aesthetic cost of a layout, order of node placement, and local optimization techniques. An implementation of our algorithm in C on an IBM RS-6000 workstation lays out most graphs of up to 100 nodes in under 10 seconds (many in less than 2 seconds) and consistently generates layouts which, with respect to the three aesthetic criteria described below, are better than those produced by the "force-directed" algorithm of Fruchterman and Reingold [FR91] and the simulated annealing algorithm of Davidson and Harel [DH91].

Di Battista et al. [DBETT93] discusses three aesthetic criteria for drawing graphs: edge lengths should be uniform; non-adjacent nodes should be far away from each other; and the number of edge crossings should be minimal. The first two criteria are characteristic of the *spring embedder* model proposed by Eades [Ead84] and further developed by Kamada and Kawai [KK89] and Fruchterman and Reingold [FR91]. Our algorithm's *aesthetic cost function* quantifies a weighted evaluation of all three criteria.

Our algorithm has three stages. In the first stage, it determines the order in which nodes will be placed by constructing a minimal height breadth-first spanning tree of the graph. This ordering enumerates the nodes of the graph from the center outwards. In the second stage, the algorithm places the nodes one at a time by sampling positions near the already placed neighbors of a node. After placing each node, the algorithm locally optimizes the layout near that node. The optimization process propagates itself through neighboring nodes; that is, whenever the local optimization procedure succeeds in improving the placement of a node, it calls itself recursively on all of the already placed neighbors of that node. After this process stabilizes at a local optimum, the algorithm proceeds, iterating through the list of nodes. The third stage fine-tunes the layout by again performing local optimization at every node.

Our algorithm's speed is largely the result of the way it computes the aesthetic cost function. First, the computation is incremental, so that a small change in the layout requires minimal recomputation. Second, the algorithm approximates the cost by ignoring interactions between far away, nonadjacent nodes (as in the "grid-variant" principle of Fruchterman and Reingold [FR91]). Third, the algorithm uses the uniform grid technique of Akman et al. [AFKN89] to compute edge crossings. Although these methods are not conceptually original, none of the published graph layout algorithms apply them in combination to computing the aesthetic cost function, which is the inner loop of computation.

Unlike most of the published layout algorithms for undirected graphs, which initially place all nodes randomly, our algorithm places nodes one at a time in a deterministic order. The inspiration for our method is a node-ordering strategy proposed by Watanabe [Wat89]. Our algorithm places the nodes in an order that reflects their centrality in the graph, thereby minimizing the constraints on the nodes which will eventually occupy the denser regions of

*Carnegie Mellon University. For a complete paper, please send email to Daniel.Tunkelang@cs.cmu.edu. This work is based on my Master's Thesis, which was supervised by Charles Leiserson at MIT and Mark Wegman at the IBM T J Watson Research Center.

the layout. This strategy exemplifies a general principle: a deterministic strategy based on knowledge of the problem is better than a random one based on ignorance.

The other innovation in our algorithm is its local optimization method. Whenever the algorithm improves the layout by moving a node, it propagates the local optimization process to that node's neighbors. That is, whenever the improvement procedure finds a way to reduce the cost associated with a particular node, it effects that improvement and then calls itself recursively on all of that node's neighbors. This approach provides several benefits. First, perfect initial placement is not so important, because the immediate attempt at local optimization fine-tunes the initial guess. Secondly, propagating optimization through neighbors tends to find the regions of the layout that need improvement and concentrate on them. Thirdly, this method of local optimization ensures that the algorithm gets what it pays for; the time it spends in local optimization is bounded in terms of the number of improvements the optimization generates. This method of local optimization, together with a method for rapid initial placement, makes the algorithm fast and effective.

The proof of our algorithm's merit is in its performance. Fruchterman and Reingold, as well as Davidson and Harel, graciously allowed implementations of their algorithms to be used for comparison with an implementation of ours. The test suite for comparison consisted of examples from their papers, as well as "textbook" examples of graphs and randomly generated graphs of up to 64 nodes. The proposed algorithm's running time is about the same as that of Fruchterman and Reingold and is much faster than the simulated annealing algorithm of Davidson and Harel. All three algorithms were aiming for the same aesthetic criteria (uniformity of edge lengths, distribution of nodes, and edge crossings). Our algorithm consistently produced the best layout with respect to these measures, especially the number of edge crossings. A drawback of our algorithm is that it does not explicitly consider the angles between adjacent edges, since measuring them would have required floating point computation. As a result, some of these angles are almost illegibly small. In general, however, our algorithm is very effective at producing low-crossing, aesthetically pleasing drawings of graphs of up to 64 nodes and average degree of up to four or five. Beyond this, the graphs become too dense for the proposed algorithm, and for those cited as well.

In summary, our algorithm improves on existing work by optimizing the inner loop of computation, intelligently choosing an order for node placement, and using a local optimization strategy that exploits local structure within a layout. We continue to explore the many open problems in graph layout.

References

[AFKN89] V. Akman, W. R. Franklin, M. Kankanhalli, and Narayanaswami. Geometric computing and uniform grid technique. *Computer-Aided Design*, 21(7):410–420, September 1989.

[DBETT93] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. Algorithms for drawing graphs: an annotated bibliography. Technical report, Brown University, June 1993.

[DH91] R. Davidson and D. Harel. Drawing graphs nicely using simulated annealing. Technical report, Department of Applied Mathematics and Computer Science, Weizmann Institute of Science, April 1991. Revised version.

[Ead84] P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.

[FR91] T. Fruchterman and E. Reingold. Graph drawing by force-directed placement. *Software — Practice and Experience*, 21(11):1129–1164, November 1991.

[KK89] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31:7–15, April 1989.

[Wat89] H. Watanabe. Heuristic graph displayer for g-base. *International Journal of Man-Machine Studies*, 30:287–302, 1989.

Drawing Ranked Digraphs with Recursive Clusters

Stephen C. North

Abstract not Available.

Graph Drawing Algorithms for the Design and Analysis of Telecommunication Networks

Ioannis G. Tollis and Chunliang Xia *

The problem of drawing a graph in the plane has received increasing attention recently due to the large number of applications [1]. Examples include VLSI layout, algorithm animation, visual languages, and CASE tools [2]. Vertices are usually represented by points and edges by simple open curves. In this paper we study techniques for visualizing telecommunication networks. The visualization of telecommunication networks is very useful in aiding the design process of minimum cost networks and the management of network operations [6]. We present linear time algorithms for drawing telecommunication networks (with optimal area) so that important properties are displayed.

The design and analysis of cost effective *survivable telecommunication networks* is a very important problem [3, 5, 7, 8, 10]. Most problems that aim towards minimizing the total cost of a network are NP-hard [4]. For that matter, computer tools to aid the design and analysis of telecommunication networks are in great demand. A central problem of such tools is how to draw a network on the computer screen such that important aspects of the network can be easily captured and an improved solution can be obtained by a user interactively.

The problem is defined as follows: Let $G = (V, E)$ be a telecommunication network with a set of nodes V (representing the sites of switches) and a set of links E (representing

*The authors are with the Department of Computer Science, The University of Texas at Dallas, Richardson, TX 75083-0688. tollis@utdallas.edu, xia@utdallas.edu.

the electrical wires or optical fiber links between nodes). The traffic requirements between the nodes are defined by an $n \times n$ matrix T, where $T(i,j)$ corresponds to the amount of traffic between nodes i and j. We need to design a network which (a) satisfies the traffic requirements, (b) can survive failures, and (c) the cost of the network is minimum. A network is *1-survivable* if it can survive the failure of a link e, i.e., the removal of link e does not disconnect the network and the traffic that originally travels through e can be accomodated on another path. The multi-ring architecture is considered as a cost-effective survivable network architecture due to its simplicity, improved survivability and bandwidth sharing [9, 10]. A *ring cover* of G is a set of rings (cycles) of G such that the rings are connected and every node in V is included in at least one ring. Apparently, a network with a ring cover is 1-survivable since the switches automatically send the required traffic around the ring if a link failure occurs.

Since the nodes of the network correspond to sites, they have geographic coordinates. Hence, the network can be drawn naturally with little effort. However, the important properties of the network that designers are interested in (such as rings) are not displayed. In this paper, we describe several algorithms for drawing telecommunication networks in order to to aid the design of cost-efficient networks. Given a ring cover of a network, our algorithms display it in such a way that rings are easily identifiable and possible problems can be easily spotted by network designers.

Ideally, we want to draw all rings as cycles, but this is not always possible if rings are not allowed to intersect. For instance, if three rings share a common node, then the cycles will intersect. In cases like this, we will use a geometric shape with a slight deviation from cycle, called *almost-cycle*, to represent rings. An almost-cycle is such a geometric shape that, except for very few nodes, almost all nodes of a ring are placed on the boundary of a cycle. Even if we allow rings to cross, not all ring covers admit such a representation. In this paper, we present a necessary condition for ring covers that admit such a representation.

As is the case in most graph drawing algorithms, the exsitence of unnecessary crossings is viewed as harmful to the readability of the drawings. Thus, minimizing such crossings is central to our approach. Also, we assume the exsitence of a *resolution rule*, that is, in the final drawing, any two nodes of the network must be kept far enough so that the human eye can tell them apart. This implies that the drawing cannot be arbitrarily scaled down. If we honor such a resolution rule and represent rings as cycles, there is a trivial lower bound of $\Omega(N^2)$ on the area required for the drawing, where N is the number of nodes in the network.

In order to capture the complexity of interaction among rings, a new graph G' is introduced. Given a network G and its ring cover C, a *contact node* is a node of G that is contained in at least two different rings of C. Let V' be the collection of all contact nodes, $G' = (C \cup V', E')$, where $E' = \{(r,v) \mid r \in C,\ v \in V' \text{ and } v \text{ is a node of ring } r\}$. Graph G' is called the *ring-contact node graph*. According to the definition of a ring cover, G' is a connected graph.

When G' is a tree, three different drawing algorithms are introduced: *outside drawing*, *inside drawing*, and *mixed drawing*. All of the algorithms create zero unnecessary crossings and take linear time.

In the rest we present our main results.

It seems natural to put two rings side by side when they share a contact node, since all rings will be drawn on the outer space, we call this style of drawing *outside drawing*.

Theorem 1 *Algorithm* outside drawing *results in a drawing which takes $O(N^2)$ area, and each ring is represented by an almost-cycle.*

Instead of placing two rings side by side when they share a contact node, we place one inside another. We call this style of drawing *inside drawing.*

Theorem 2 *Algorithm* inside drawing *results in a drawing which takes* $O(N^2)$ *area, and each ring is represented by a cycle.*

There are some cases where an inside drawing outperforms an outside drawing. There are also cases where an outside drawing outperforms an inside drawing. Hence, we combine the strength of the two to obtain a *mixed drawing.*

We assign a weight to each node v in G'. If v is a contact node, v has weight 0; if v is a ring, v has a weight equal to the number of nodes in the ring. Let N' be the length of the second longest path in G'.

Theorem 3 *Algorithm* mixed drawing *results in a drawing which takes* $O(N \times N')$ *area, and each ring is represented by a cycle.*

References

[1] G. Di Battista,P. Eades, R. Tamassia and I. G. Tollis, "Algorithms for Automatic Graph Drawing: An Annotated Bibliography," Dept. of Comp. Science, Brown Univ., Technical Report, 1993. Available via anonymous ftp from wilma.cs.brown.edu(128.148.33.66), files /pub/gdbiblio.tex.Z and /pub/gdbiblio.ps.Z

[2] 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.

[3] G.R. Dattatreya, J.P. Fonseka, K. Kiasaleh, I.H. Sudborough, I.G. Tollis, and S. Venkatesan, "Advanced Network Topologies for Network Survivability," Technical Report, UTD, January 1993.

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

[5] W.D. Grover, B.D. Venables, J.H. Sandham, and A.F. Milne, "Performance Studies of a Slef-Healing Network Protocol in Telecom Canada Long Haul Networks," *IEEE GLOBECOM 1990*, pp. 403.3.1-403.3.7.

[6] G. Kar, B. Madden and R. S. Gilbert, "Heuristic layout Algorithms for Network Management Presentation Services" *IEEE Network*, November, 1988.

[7] H. Sakauchi, Y. Nishimura, and S. Hasegawa, "A Self-Healing Network with an economical Spare-Channel Assignment," *IEEE GLOBECOM 1990*, pp. 403.1.1-403.1.6.

[8] J. C. Shah, "Restoration Network Planning Tool", Proc. 8th Annual Fiber Optic Engineers Conf. April 21, 1992.

[9] T.H. Wu, D.J. Collar, and R.H. Cardwell, "Survivable Network Architectures for Broadband Fiber Optic Networks: Model and Performance Comparisons," IEEE Journal of Lightwave Technology, Vol. 6, No. 11, November 1988, pp. 1698-1709.

[10] Tsong-Ho Wu and R. C. Lau, *A Class of Self-Healing Ring Architectures for SONET Network Applications*, IEEE Trans. on Communication, Vol. 40, No. 11, Nov. 1992.

A View to Graph Drawing Algorithms through GraphEd

Michael Himsolt *

We compare a collection of graph drawing algorithms implemented in our Graph^{Ed} system. We report on our experience from running these algorithms on a large number of examples both from the literature and by our own, and present our evaluation of the practical relevance of the algorithms and layout criteria.

The representation of complex structures as graphs is widespread. Graph drawing has gained increasing importance in many areas of Computer Science, but has proved to be a difficult task. Our Graph^{Ed} system is an approach to support solutions to this problem. Graph^{Ed} has been used by practitioners for database design, Petri nets and electrical circuits. One of its major applications is the implementation and evaluation of graph layout algorithms.

With its capabilities to create and edit graphs, Graph^{Ed} provides an effective environment to create and test large sets of examples. Since all drawing algorithms are built into one tool, it is easy to compare the effect of different algorithms on the same graph.

There is also a special module ("layout suite") that runs all applicable layout algorithms on one graph. It also writes statistical data such as the size of the graph, the space used, or the number of bends and crossings. We have created a large database of graphs and statistics with that module.

We regard testing many examples as an adequate and probably the best way to get precise data on the practical relevance of layout criteria and graph drawing algorithms. Currently, the following algorithms are implemented :

- Spring embedder (based on algorithms by Fruchtermann/Reingold and Kamada)
- Tree drawing (Walker)
- Dag drawing (Sugiyama/Tagawa/Toda)
- Planar drawing on a grid (Woods)
- Planar drawing on a grid with bends minimization (Tamassia)
- Planar straight-line drawing with convex faces (Chiba/Onoguchi/Nishizeki)
- Planar straight-line drawing on a grid (de Fraysseix/Pach/Pollack and Chrobak/Payne)
- Drawing Petri nets from term descriptions (Seisenberger)

We have tested these algorithms on a large number of examples, both from literature and by our own. We have tested arbitrary graphs as well as graphs with special structure (e.g. grids). Our experiences can be comprised as follows :

- Spring embedders produce good layouts for most graphs. They stress the display of isomorphic and symmetric substructures. A major drawback is the high runtime.
- From the practical point of view, tree drawing seems to be solved, since the algorithm reproduces the tree structure in the same way as the user would do it by hand.
- The algorithm of Sugiyama/Tagawa/Toda provides a good base for drawing dag's, but the layouts are not as good as trees.

*Universität Passau, 94030 Passau. himsolt@fmi.uni-passau.de

- Planar graphs are generally difficult to draw, because "planarity" alone does not explain the intrinsic structure of the graph. Moreover, some planar graph drawing algorithms depend on the actual planar embedding, which causes further problems.

  Tamassia's algorithm gives our best layouts, although it has the highest running time in this class. The drawings look pretty.

  Wood's algorithm gives suitable results for small graphs, but the number of bends is much higher as with Tamassia's algorithm.

  The algorithms of Chiba/Onoguchi/Nishizeki and de Fraysseix/Pach/Pollack are of limited practical use. They tend to cluster nodes and often destroy pleasing pictures.
- The Petri net algorithm takes agents as input, which are term descriptions of the nets. It produces very good layouts. This comes from the fact that the agents provide detailed information on the structure of the nets. The information is used to draw the graph as a designer would do.

  The good results stimulate our work on a general framework of graph grammar based layout algorithms.

¿From the experiments, our actual ranking of layout criteria is :

1. Distribute the nodes in a uniform fashion.
2. Display the intrinsic structure of the graph.
3. Display symmetric and isomorph substructures of the graph.
4. Use few edge crossings to draw the graph (none if the graph is planar).
5. Use few bends to draw the graph.
6. Place nodes and bends on a grid.

"Straight line edges" may or may not be a good criterium, depending on the other criteria used in a particular algorithm. It often imposes restrictions on the layout, like in the algorithms of Chiba/Onoguchi/Nishizeki and de Fraysseix/Pach/Pollack. We have found out that allowing a few bends, as in Tamassia's algorithm, is usually a good choice.

Graph^{Ed}is available with anonymous ftp from forwiss.uni-passau.de (132.231.1.10), `/pub/local/graphed`.

References

[1] G. Di Battista, P. Eades, R. Tamassia and I.G. Tollis "Algorithms for Drawing Graphs : An Annotated Bibliography" (contains references on most of the algorithms mentioned above)

[2] M. Himsolt, "Konzeption und Implementierung von Grapheneditoren", Dissertation, Universität Passau, 1993 (to be published)

[3] K. Seisenberger, "Termgraph : ein System zur zeichnerischen Darstellung von strukturierten Agenten und Petrinetzen", Diplomarbeit, Universitaet Passau, 1991

An Automated Graph Drawing System Using Graph Decomposition

C. L. McCreary, * C. L. Combs, † D. H. Gill, ‡ and J. V. Warren‡

Introduction

This paper presents a graph layout technique (CG) based on the hierarchial decomposition of graphs. One of the major differences between the graphs drawn by CG and other systems is that the vertices in CG's graphs are spaced in a balanced way both vertically and horizontally. Many other systems partition vertices into levels, and all vertices of the same level are placed on the same horizontal axis. Nodes tend to bunch toward the top of the graph is these systems. By using our graph decomposition technique, CG is able to determine balanced vertical spacing as well as balanced horizontal spacing. Edge crossings are reduced by a very efficient variant of the Barycentric method that exploits the subgraph hierarchy.

Graph Decomposition

Clan-based graph decomposition [2] is a parse of a directed acyclic graph (DAG) into a hierarchy of subgraphs called clans. A subset X of DAG G is a *clan* iff for all $x, y \in X$ and all $z \in$ G - X, (a) z is an ancestor of x iff z is an ancestor of y, and (b) z is a descendant of x iff z is a descendant of y.

A simple clan C, with more than three vertices, is classified as one of three types . It is (i) *primitive* if the only clans in C are the trivial clans; (ii) *in dependent* if every subgraph of C is a clan; or (iii) *linear* if for every pair of vertices x and y in C, x is an ancestor or descendant of y. Independent clans are sets of isolated vertices which can be visualized as horizontal neighbors. Linear clans are sequences of one or more vertices $v_i, v_{i+1}, ..., v_{j-1}, v_j$ where for i ¡ k, v_i is an ancestor of v_k, and can be seen as a vertical string. Any graph can be constructed from these simple clan as well as decomposed into a parse tree with clan components. Primitive clans do not fall into the clear-cut categories of vertices that should be laid out horizontally or laid out vertically. One procedure for further reduction of primitive clans is to form an independent clan of the source vertices of the primitive and decompose the remainder of the primitive. The independent clan is linearly connected to the rest of the clan. The parse tree of any completely decomposed graph is a bipartite tree where the internal vertices represent clans that are classified as either linear or independent.

The parse tree of the graph can be given a geometric interpretation. A bounding rectangle with known width and height can be associated with each clan. The parse tree hierarchy shows the embedding of the bounding rectangles.

A simple two-dimensional algebra defines the *bounding rectangles*. Singleton DAG vertices (or equivalently parse tree leaves) have unit square bounding rectangles. Linear clans require

*Dept. of Computer Science and Engineering, Auburn University, Auburn, AL. mccreary@eng.auburn.edu
†Equifax Incorporated, Atlanta, Georgia.
‡The MITRE Corporation, McLean, Virginia.

an area whose length is the sum of the lengths of the component clans and whose width is the maximum width of the component clans. Independent clans require an area whose width is the sum of the widths of the component clans and whose length is the maximum of the lengths of the component clans. To achieve an aesthetically pleasing layout, the vertices are centered within the bounding rectangles. Since clans are defined as groups of vertices with identical connections to the rest of the graph, clans can easily be contracted to a single vertex. Any vertex not in the clan that was connected to a clan vertex will be connected to the contracted vertex. By allowing segments of the graph to be contacted, the user can simplify graphs for viewing by contracting those parts which are not relevant to her investigation. Contracted vertices can be expanded to show the original clan configuration.

The Barycentric Method Adapted to Clans

The Barycentric method [6], a heuristic for reducing the number of edge crossings in two consecutive levels of a graph, is modified by considering adjacent clans instead of adjacent vertices. The process proceeds by rearranging adjacent parse tree children of the largest unprocessed linear clan. Because groups of vertices (clans), rather than individual vertices, are subject to rearrangement at each step, the method is much more efficient than the standard Barycentric method.

By inspecting the structure of a graph through graph decomposition, an aesthetically pleasing and natural layout of the graph vertices can be constructed. By adapting existing edge routing techniques [5, 3, 1, 4], CG is able to draw arcs that have few unnecessary edge crossings and that are smooth and straight.

References

[1] E. R. Gansner, S. C. North, and K. P. Vo. Dag - a program that draws directed acyclic graphs. *Software - Practice and Experience*, 18(11):1047–1062, November 1988.

[2] D. H. Gill, T. J. Smith, T. E. Ohasch, C. L. McCreary, and I. V. Warren R. E. K. Stirewalt. Sparial-temporal analysis of program dependence graphs for usefull parallelisim. *Journal of Parallel and Distributed Computing*, to appear, 1993.

[3] D. Jablownowski and V. A. Guana Jr. Gmb: A tool for manipulating and anomating graph data structures. *Software - Practice and Experience*, 19(3):283–301, March 1989.

[4] L. A. Rowe et al. A browser for directed graphs. *Software - Practice and Experience*, 17(1):61–76, January 1987.

[5] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchichal system structures. *IEEE Transactions on Syst., Man, and Cybernetics*, 11(2):109–125, February 1981.

[6] J. N. Warfield. Crossing theory and hierarchy mapping. *IEEE Transacrions on Syst., Man. and Cybernetics*, 7(7):505–523, July 1977.

Maximum Planar Subgraphs and Nice Embeddings: Practical Layout Tools

Michael Jünger and Petra Mutzel *

In automatic graph drawing, a given graph has to be layed out in the plane, possibly according to a number of topological and aesthetic constraints. Nice drawings for sparse nonplanar graphs can be achieved by determining a maximum planar subgraph and augmenting an embedding of this graph. Finding maximum planar subgraphs is NP-hard, and therefore this technique appeared not to be practical.

We attack the problem with techniques of polyhedral combinatorics. The polytope $\mathcal{PLS}(G)$ of G is defined as the convex hull over all incidence vectors of planar subgraphs of G and called the planar subgraph polytope. The problem of finding a planar subgraph P of G with weight $w(P)$ as large as possible can be written as the linear program $\max\{w^T x \mid x \in \mathcal{PLS}(G)\}$, since the vertices of the polytope $\mathcal{PLS}(G)$ are exactly the incidence vectors of the planar subgraphs of G. In order to apply linear programming techniques to solve this linear program one has to represent $\mathcal{PLS}(G)$ as the solution of an inequality system. Due to the NP-hardness of our problem, we cannot expect to be able to find a full description of $\mathcal{PLS}(G)$ by linear inequalities. But even a partial description of the facial structure of $\mathcal{PLS}(G)$ by linear inequalities is useful for the design of a "branch and cut"-algorithm, because such a description defines a relaxation of the original problem. Such relaxations can be solved within a branch and bound framework via cutting plane techniques and linear programming in order to produce tight bounds. For a partial description by inequalities we only have to concentrate on proper faces of maximal dimension of $\mathcal{PLS}(G)$, so-called facet-defining inequalities. One of the main results of our investigation of the facial structure of the planar subgraph polytope is the fact that all the subdivisions of K_5 or $K_{3,3}$ turned out to be facet-defining for $\mathcal{PLS}(G)$.

We have designed a branch and cut algorithm using facet-defining inequalities for $\mathcal{PLS}(G)$ as cutting planes. In a cutting plane algorithm, a sequence of relaxations is solved by linear programming. After the solution x of some relaxation is found, we must be able to check whether x is the incidence vector of a planar subgraph (in which case we have solved the problem) or whether any of the known facet-defining inequalities are violated by x. If no such inequalities can be found, we cannot tighten the relaxation and have to resort to branching, otherwise we tighten the relaxation by all facet-defining inequalities violated by x which we can find. Then the new relaxation is solved, etc. The process of finding violated inequalities (if possible) is called "separation" or "cutting plane generation". Although the vectors x coming up as solutions of LP-relaxations in the above outlined process have fractional components in general, they are often useful to obtain information on how a high-valued planar subgraph might look like. We exploit this idea with a greedy type heuristic with respect to the solution values of the edges. So, in addition to the upper bounds $w^T x$ on the value of a maximum planar subgraph, we also obtain a lower bound $w^T \overline{x}$ from the planar subgraph incidence vector $\overline{x}$ derived heuristically from x. The lower bound heuristic as well as the cutting plane generation are based on a planarity testing algorithm of Hopcroft and Tarjan [1].

In our computational experiments we solved several problems from the literature to optimality. Among them there is a graph given by Tamassia, Di Battista and Batini in a paper

*Institut für Informatik, Universität zu Köln, Pohligstraße 1, 50969 Köln, Germany. mjuenger@informatik.uni-koeln.de, mutzel@informatik.uni-koeln.de.

about automatic graph drawing ([TBB88]). In order to get the maximum planar subgraph of the graph the algorithm removed four of the 62 edges. The computation took 24 seconds on a SUN SPARCstation 10 model 20. For the graph given by Kant in [K92] on 45 nodes and 85 edges the algorithm found an optimum solution with 82 edges in 7 seconds.

In order to explore the limits of our branch and cut algorithm, we tested it on a series of randomly generated graphs. We could observe that the easiest problem instances are those on sparse graphs which are almost planar and dense graphs. We could observe that our code is able to solve all problem instances with up to 40 edges to optimality. Even though we cannot solve all instances of bigger sizes to optimality, our approach allows us to give quality guarantees which state that our solutions are less than $p\%$ below the optimum, where p is given when the computation stops after a certain amount of time. The quality guarantee turns out to be typically less than 10% for random problems with up to 80 edges.

References

[HT74] Hopcroft, J., and R.E. Tarjan, "Efficient planarity testing", *J. ACM 21* (1974) 549–568

[K92] Kant, G., "An O(n^2) Maximal Planarization Algorithm based on PQ-trees", Utrecht University (1992)

[TBB88] Tamassia, R., G. Di Battista, and C. Batini,"Automatic graph drawing and readability of diagrams", *IEEE Transactions on Systems, Man and Cybernetics 18* (1988) 61–79

Heuristics for Planarization by Vertex Splitting

Peter Eades and Xavier Mendonça

Intuitively, a vertex v may be "split" by making two copies v_1 and v_2 and attaching the edges incident with v to either v_1 or v_2. The operation is illustrated in Figure 1.

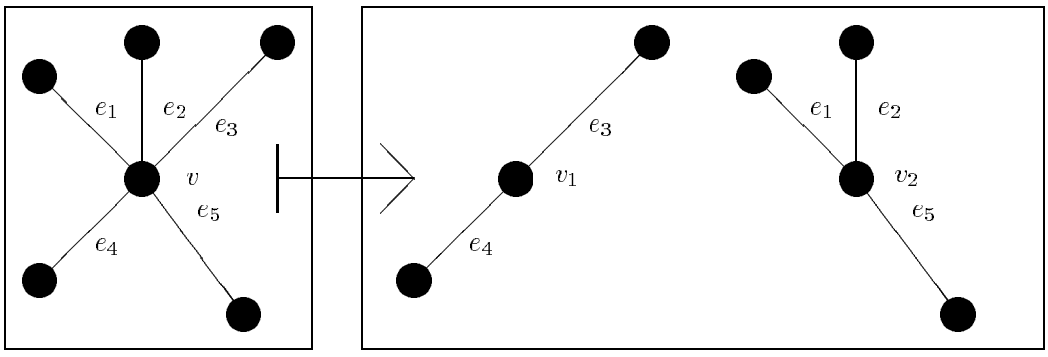


Figure 1: The spliting operation

This simple operation is introduced to change the graph a little to make it amenable to layout.

Effective layout algorithms impose restrictions on the input graph structure. The two main sources of this problem are:

layout algorithm design. The layout algorithm is limited to special classes of graphs. For instance, there is a wealth of layout algorithms for planar graphs; however, these algorithms are useless for nonplanar graphs.

task difficulty. The layout task has high complexity and we must impose restrictions on the input to be able to handle the task within reasonable computational resources. For instance, finding a planar drawing of a graph in which all edges have length one is, in general, NP-hard. However, when the input is restricted to trees, there are trivial layout algorithms for this aesthetic.

The problem of transforming the input graph to conform with the restrictions is an important concern. These transformations should change the graph as little as possible so the graph does not lose it "identity". For instance, to "planarize" a graph we may delete a small number of edges, or add a small number of dummy vertices (at crossings). Many optimisation problems of minimising the number of such transformations are NP-complete [Men93], but effective heuristics are available for some.

We are concerned with the *splitting* transformation described above. Manual layout techniques sometimes involve making a copy of a node in order to simplify layout (see, for example, [Lim83]). We aim to investigate the automation of layout techniques using the splitting operation.

We are particularly interested in four basic aesthetic criteria: planarity, edge length, symmetry and straight line drawing.

We present a heuristic for planarization by splitting which we call *SPLIT-PLANARIZE.* The SPLIT-PLANARIZE heuristic is based on Lempel, Even, and Cederbaum's planarity testing algorithm [LEC66], its implementation using PQ-trees [BL76], and the PLANARIZE algorithm of Jayakumar, Thulasiraman and Swamy [JTS89].

We also present two other algorithms which use vertex splitting for different objectives. The first algorithm, TENSION-SPLIT, is a heuristic which consists of a modification of a spring system. A tension is calculated for each eligible vertex, and after each local minimization step a vertex with high tension is split. The objective of the TENSION-SPLIT algorithm is to produce a layout of a graph G' (transformed from G by splitting) in which the Euclidean distance between the pairs of vertices u and v in G' is equal to the length of the edges uv.

Finally, an optimization criterion to perform splitting using simulated annealing is presented. This is a very interesting approach since several different criteria can be applied to produce different embeddings.

References

[BL76] K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. *J. Comp. Syst. Sci.*, 13(3):335–379, Dec. 1976.

[JTS89] R. Jayakumar, K. Thulasiraman, and M. N. S. Swamy. $o(n_2)$ algorithms for graph planarization. *IEEE Trans. Computer-Aided Design*, 8(3):257–267, Mar. 1989.

[LEC66] A. Lempel, S. Even, and I. Cerderbaum. An algorithm for planarity testing of graphs. In *Theory of Graphs, Int. Symp.*, pages 215–232, Rome, Italy, July 1966. P. Rosenstiel (Ed.), Gordon & Breach.

[Lim83] M.I.M. Holdings Limited. *Organization Charts.* M.I.M. Holdings Limited, Queensland Australia, 1983. Internal Memorandum.

[Men93] C. F. X. De Mendonça. *A Layout System for Information System Diagrams.* PhD dissertation, The University of Queensland, Department of Computer Science, June-August 1993.

Planar Graph Embedding with a Specified Set of Face-Independent Vertices

Takao Ozawa*

Introduction

It is known that there are many ways, in general, to embed a biconnected planar graph in the plane. In this paper we introduce a new graph embedding problem as defined below, and give a very efficient solution algorithm to it. Let G be a biconnected planar graph with vertex set V.

Problem FIVS-EMB: Given a subset U of V, find, if possible, an embedding of G in the plane such that no two vertices of U appear on a face boundary(each vertex in U is covered by a distinct face).

In graph theory two vertices are said mutually independent if they are not the end vertices of an edge. Extending the concept of independence in relation with edges to that in relation with faces, we say that two vertices not appearing on a face boundary are face-independent. If we want some edges, in addition to vertices, being face-independent, we only have to place a new vertex on each of the edges so that each edge is converted to a series connection of two edges incident to the new vertex, and then include the newly-added vertices in U. In integrated circuit layout it may happen that some elements should not be placed closely to avoid mutual interference. We can say that the graph embedding of FIVS-EMB takes such a constraint into consideration in a simplified way.

Solution Algorithm

Our solution algorithm to problem FIVS-EMB is based on the vertex addition algorithm for planarity testing [1], and is implemented using PQ-trees [2]. The vertices of G are labeled with the *st-numbers*, and thus we have V={*1, 2, .., n*} where n is the number of vertices in V. Let G_k be the subgraph of G induced by the vertex set $V_k = \{1, 2, ... k\}$. Roughly speaking, the vertex addition algorithm successively embeds G_k for k=*1, 2, .., n* in the plane.

Now, a vertex pair $\{x,y\}$($\{x,y\}$ is not equal to $\{1,n\}$) is called a separation pair if the removal of the pair results in a disconnected graph. Let C be a connected component of the resultant graph containing vertices whose st-numbers are between x and y. The subgraph

*Department of Applied Mathematics and Informatics, Ryukoku University, Seta, Ohtsu, Siga 520-21, Japan. ozawa@rins.ryukoku.ac.jp

of G which is obtained by adding x and y to C is called an $\{x,y\}$-split component. There may be two or more split components for the pair $\{x,y\}$, and there may be an edge or edges connecting vertices x and y. Let *S(x,y)* be the set which consists of all $\{x,y\}$-split components and edges connecting x and y, if any. Different embedding can be obtained by the following operations. (1) *permutation:* changing the embedding order of the split components and edges belonging to *S(x,y)*, and (2) *reflection:* reflecting biconnected split components in *S(x,y)*.

For PQ-trees operations on nodes corresponding to the above operations on *S(x,y)* are defined. A subgraph formed by *S(x,y)* with a fixed embedding order of split components and edges in it is called a composite split component.

In our solution algorithm we try to find an embedding of G_k which satisfies the condition of problem FIVS-EMB. Difficulty arises when there are two or more ways of embedding satisfying the condition and the entire embedding of G_k can not be finalized. We present two major sub-algorithms coping with this difficulty. The first one of them finds, by applying the operations of permutation and reflection, the most desirable embedding of G_k while leaving some part of the embedding undecided. In order to implement this sub-algorithm labels which indicate the existence of vertices in U on the boundaries of split or composite split components, are attached to the nodes of the PQ-tree representing G_k, and the sub-algorithm finds the most desirable labels of nodes while bubbling up the PQ-tree. The second major sub-algorithm decides, using an auxiliary bipartite graph, the embedding of split or composite split components which are previously formed and contained in the relevant split component, but whose embedding has not been finalized. We also present a sub-algorithm for finding the vertices contained in a split component while dealing with PQ-trees. This sub-algorithm is necessary for carrying out the second major sub-algorithm mentioned above.

The time complexity and the space complexity of our solution algorithm are both *O(n)*.

References

[1] A. Lempel, S. Even and I. Cederboum, "An algorithm for planarity testing of graphs," in Theory of Graphs, International Symposium, Rome July 1966, pp. 215-232, Gorden and Breach, N. Y., 1967.

[2] K. S. Booth and G. S. Lueker, "Testing for consecutive ones property, interval graphs and graph planarity using PQ-tree algorithms," J. Computer and System Sciences, vol. 13, pp. 335-379, 1976.

Implementation of the Planarity Testing Algorithm by Demoucron, Malgrange and Pertuiset

Bjorn Sigurd Benestad Johansen *

Previous Planarity Algorithms. Hopcroft & Tarjan [1] and Booth & Leuker [2] have presented (quite complicated) linear planarity algorithms. A much simpler algorithm has been presented by Demoucron et al [3]. The algorithms by Hopcroft & Tarjan and Demoucron et al will be referred to henceforth as HT and DMP respectively.
Although DMP is basically simple, it is abstract, and no data structures or details are given. The goal of this authors implementation of DMP, DMP–HT, has been to provide a fast–yet–simple planarity algorithm model.

A Straightforward Implementation of DMP. As described in Bondy & Murty [4], a straight–forward implementation of DMP requires sub–routines for:
1) Finding a cycle in the input–graph G.
2) Determining the fragments (in Bondy&Murty called bridges) of $\overline{G_i}$ in G and their vertices of attachment to $\overline{G_i}$, where $\overline{G_i}$ is a plane subgraph of G.
3) Determining the boundary of each region in $\overline{G_i}$.
4) Determining, for each fragment F, the regions of $\overline{G_i}$ where F is drawable.
5) Finding a path p in some fragment F of $\overline{G_i}$, between two of F's Vertices of Attachment.

Previous Implementations of DMP. Rubin [5] has given an $O(n^2)$ implementation of DMP, implementing to a large extent the 5 Sub–routines above. Rubin claims that his implementation compares favorably to HT, when run on a class of maximum planar test–graphs. Yeh [6] has subsequently investigated practical improvements to the implementation by Rubin.

DMP–HT. HT and DMP both include paths, one at a time, in the plane embedding being built. This observation led to DMP–HT in which techniques from HT partly are used to implement DMP.
DMP–HT utilize a totally different approach then the ones from Bondy&Murty and Rubin/Yeh. DMP–HT is based on a linear Initial Phase and a Test Phase.
Initial Phase. The first part of the Initial Phase is almost identical to the first part of HT. (That is, two values are computed for each vertex, and the adjacency lists are sorted.) Then the input–graph G is transformed to a tree T, where the vertices in T represent edge–disjoint subgraphs of G, and the union of the subgraphs equals G. The Initial Phase removes the need for explicitly implementing any of the 5 Sub–routines above.
Test Phase. A plane embedding $\overline{G_i}$ is (implicitly) being built. As in DMP, $\overline{G_i}$ is initially empty, then a cycle is (implicitly) added, then one and one path is (implicitly) added. The inclusion of a path p through a fragment F in a region r of $\overline{G_i}$ will essentially only require:

Loop1. Determine the truth value of one or two statements and performing three or four imperatives for each new fragment emerging as a result of the the inclusion of p in G_i.

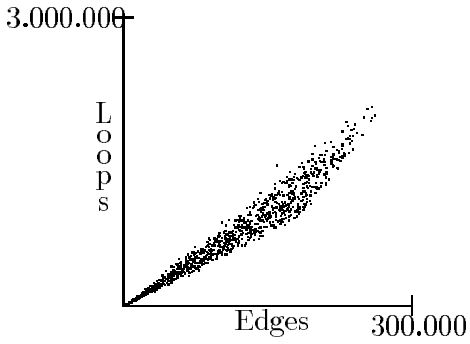
*Department of Informatics, University of Oslo, Norway. bjornjoh@ifi.uio.no

Loop2. Determine the truth value of one or two statements and performing two or four imperatives for each fragment ($\neq F$) which was drawable in r.

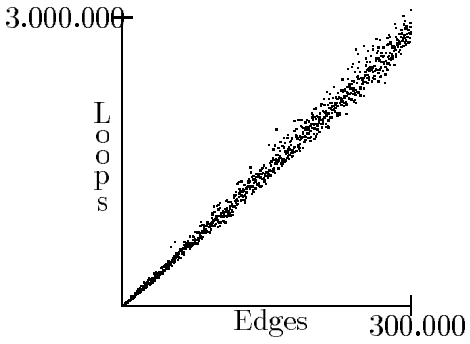
Comparison with HT. DMP–HT and HT seem to contain approximately the same number of lines of code.
DMP–HT has not been programmed in the same language and on the same computer as in [1]. Programmed in Simula on a Sparc–10, DMP–HT tested maximum planar graphs with 1000 vertices in approximately 2.5 seconds. This is probably inferior to HT, but may be satisfactory for practical purposes. But DMP–HT has a worst–case time ratio of $O(n^2)$, and for special graphs unsatisfactory results may occur. As in HT, the space requirements are of order $O(n)$.
DMP–HT may be viewed as conceptually simpler then HT, even though the proof of DMP–HT is fairly long. The proof shows, that a fragment F which is drawable in preferably only one region is directly accessible, that a path p through F is directly accessible from T, and that new fragments (see Loop1) are also directly accessible from T, how new edges can be added to $\overline{G_i}$ thus maintaining only sorted regions, and how the fact that all regions sorted can be utilized when determining in which regions new (Loop1) and old (Loop2) fragments are drawable. In DMP–HT every fragment will be determined as drawable in a maximum of two regions, and the inclusion of a path in $\overline{G_i}$ will result in three (rather then two) new regions. This will reduce the expected average number of drawable fragments in each region, thus reducing the expected running time of DMP–HT.
Also, DMP–HT directly tests graphs containing cut–vertices, and a plane embedding is implicitly given. An expansion to explicitly give a plane embedding in terms of regions seems to be straightforward. This plane embedding may also be constructed in a static fashion: Every path included in the plane embedding being built, can be embedded permanently in the plane.

Testresults. DMP–HT was tested empirically on two classes of planar graphs:
Class I. Construction of a 2–connected planar graph consisting of $n, n \geq 3$, vertices: First a cycle consisting of between (randomly) 3 and $max(3, \frac{n}{2})$ vertices was generated. Then, a random region r was repeatedly chosen to include a vertex v in r, and to generate an edge between v and a vertex in the boundary $V(r)$ of r with the probability $\frac{2.5}{|V(r)|}$. If $deg(v) < 2$ prevails, delete all the edges incident with v, and repeatly generate edges as above until $deg(v) \geq 2$. Finally, all the adjacency lists are to be randomized.
Class II. Maximum planar graphs were generated as above, with the exception that the initial cycle contained exactly 3 vertices, and exactly 3 new edges were generated when one vertex v was included in a random region r.
DMP–HT was empirically tested on 1,000 graphs generated from each of the two classes. Each of the graphs contained between 1,000 and 100,000 vertices. For each graph G the number of repetitions of the interior loops (Loop1 and Loop2) was counted, and plotted against $|E(G)|$. As can be seen, DMP-HT behave in an almost linear manner for these graphs:

a) "Random" Planar Graphs



b) "Random" Maximal Planar Graphs



In the complete paper, DMP–HT methodology is presented, validation of DMP–HT is given, and a complete implementation of DMP–HT is presented, assuming that the graph to be tested for planarity is stored on file as a set of adjacency–lists.

References

[1] J. E. Hopcroft & R. Tarjan, "Efficient Planarity Testing" J. Assoc. Comput. Mach. 21 (1974) 549–568

[2] K. S. Booth & G. S. Leuker, "Testing the Consecutive Ones Property, Interval Graphs, and Graph Planarity Using *PQ*–tree algorithms" J. Comput. Syst. Sci. 13 (1976) 335–379

[3] G. Demoucron, Y. Malgrange & R. Pertuiset, "Graphes Planaires: Reconnaissance et Construction de Represetations Planaires Topologiques" Revue Française de Recherche Opẽrationelle, Vol. 8, No. 3 (1st quarter, 1964), 43–47

[4] J. A. Bondy & U. S. R. Murty, "Graph Theory with Applications" Macmillan, London, 1976

[5] Frank Rubin, "An Improved Algorithm for Testing the Planarity of a Graph" IEEE Transactions on Computers, Vol. c–24, No 2, February 1975

[6] Dashing Yeh, "Improved Planarity Algorithms" Bit 22, 1 (1982), 2–16.

A Unified Approach to Testing, Embedding and Drawing Planar Graphs

Joel Small *

Let $G = (V, E)$ be a simple, undirected graph with vertex set V and edge set E. We first offer improvements to Williamson's version of the Hopcroft-Tarjan (HT) planarity testing algorithm while maintaining the **O**(n) execution time. Then, with the data structures and graph embedding in place from this algorithm, we augment the data structures and use the computed embedding to calculate and draw a rectilinear layout of the graph. The

*Naval Command, Control and Ocean Surveillance Center, San Diego, California. jsmall@nosc.mil

algorithms for augmenting data structures and drawing the graph run in **O**(n) time as well. Furthermore, the augmented data structures lend themselves to efficiently generating many nonisomorphic drawings.

The planarity testing algorithm exploits the bridge-cycle recursion for biconnected graphs introduced by HT. Briefly the idea is to choose a cycle C in G. Edges in $E(G)$ (the edge set of G) not in $E(C)$ are called the *simple bridges* of G with respect to C if both endpoints lie on C. Remove C together with all edges incident to C from G. The resulting graph is a collection of connected components. These components together with the simple bridges are called the *bridges* of G with respect to C. In any plane drawing of G, the cycle C will partition the plane into two regions, a finite and an infinite region. Each bridge must lie entirely in one region. The bridge graph of G with respect to C, denoted BRGR(G,C), is a graph whose vertices are the bridges of G with respect to C. There is an edge between two vertices B_1 and B_2 in BRGR(G,C) if B_1 and B_2 must be drawn on opposite sides of C in any plane drawing of G. The idea for an efficient algorithm to test if G is planar will be to choose a cycle C, then construct the bridges of G with respect to C. Add edges from C to each bridge to make it biconnected (these are called the *augmented bridges*), and recursively test each augmented bridge to determine if it is planar. Finally compute BRGR(G,C) and check that this graph is bipartite. If it is then this gives a valid means for locating the bridges about C to get an embedding of G.

The bridge-cycle recursion is exemplified by a data structure called the pathtree of G, denoted $PATR(G,T)$, [3, Def. 7.14]. T is a lineal spanning tree of the graph G. PATR(G,T) is a rooted, ordered tree whose vertices are paths in G defined using T. The vertices of this tree define a partition for both the vertex set $V(G)$ and the edge set $E(G)$. PATR(G,T) can be traversed in postorder to test the graph for planarity. We offer some improvements to previous versions of algorithms for traversing the pathtree. These improvements are primarily in computing and maintaining the spanning forests for bridge graphs and in simplifying the tests necessary to update the spanning forests when adding new vertices to the bridge graph. This work has facilitated the actual implementation of the algorithms.

As a part of the planarity testing, vertices of the pathtree can be colored to define an embedding for the graph if it is planar, or identify an obstruction if it is not. If the graph is planar, we show how to augment the pathtree and use this augmented pathtree to draw a rectilinear layout of the graph where vertices are represented by horizontal intervals and edges are drawn using vertical intervals (also called a weak visibility representation in [2]). The edge partition defined by the pathtree is maintained in the drawing by placing edges in the same block on the same vertical line. The general idea to obtain the drawing is described next.

Suppose we are given a planar graph G along with a PATR(G,T) that has been colored to identify a planar embedding. We assign a left-right orientation to the vertices of the pathtree that identifies the orientation of the paths as they are drawn in the plane. Next, we can use the PATR(G,T) in conjunction with the orientation to assign a valid *st* numbering to the vertices. Finally, we compute the horizontal placement for the vertices (paths of G) of the pathtree. Vertices of the graph G are placed vertically according to their st number. We plan to describe the algorithms to accomplish the drawings in more detail at the meeting.

We have written a versatile, interactive graph software package called GAP, a Graph Analysis Program, to study graphs and graph algorithms. Versions are available for both PCs and SUN workstations. The data structures and algorithms we describe for testing, embedding and drawing planar graphs have been implemented as part of this package. The five primary components for doing this are to perform a depth first search and compute low numbers (DFS), sort the vertices using a lexicographic bucket sort (LEXSORT), compute

the pathtree (PATR), build the spanning forests for bridge graphs and test for odd cycles (PLTEST), and last, augment the pathtree and compute a rectilinear layout (RECDRAW). We ran the program on maximal planar graphs generated according to the following algorithm:

procedure RandomGraph(Graph G, NumberOfVertices nv)
1. $G \leftarrow K_3$
2. tv $\leftarrow$ 3
3. While tv $<$ nv do
3.1 Select a face f at random.
3.2 Add a new vertex v to G.
3.3 tv $\leftarrow$ tv + 1.
3.4 Add edges {v,w_1}, {v,w_2}, {v,w_3} to G, where w_1, w_2, w_3 are the vertices lying on f.
3.5 Update the list of faces of G

The table below shows the mean time, in seconds, for each of the components on 100 maximal planar graphs having 500, 1000, 2000, 4000 and 8000 vertices.

$\|V\|$	DFS	LEXSORT	PATR	PLTEST	RECDRAW	TOTAL
500	0.04	0.22	0.18	0.49	0.07	1.00
1000	0.09	0.45	0.40	1.03	0.14	2.11
2000	0.18	0.95	0.80	2.05	0.30	4.28
4000	0.36	1.99	1.73	4.61	0.60	9.29
8000	0.73	3.92	2.47	9.88	1.10	18.10

BIBLIOGRAPHY

1. Rosenstiehl, P. and Tarjan, R. E., Rectilinear Planar Layouts and Bipolar Orientations of Planar Graphs, *Discrete and Computational Geometry*, **1** (1986), 343-353.
2. Tamassia, R. and Tollis,I. G., A Unified Approach to Visibility Representations of Planar Graphs, *Discrete and Computational Geometry*, **1** (1986), 321-341.
3. Williamson, S. G., *Combinatorics for Computer Science*, Computer Science Press, 1985
4. Williamson, S. G., Embedding Graphs in the Plane - Algorithmic Aspects, *Annals of Discrete Mathematics 6: Combinatorial Mathematics, Optimal Designs, and Their Applications* (J. Srivatsava, Ed.), North Holland, 1980, 349-384.

A Simple Linear-Time Algorithm for Embedding Maximal Planar Graphs

Hermann Stamm-Wilbrandt*

Introduction

All existing algorithms for *planarity testing / planar embedding* can be grouped into two principal classes. Either, they run in linear time, but to the expense of *complex* algorithmic concepts or *complex* data-structures, or they are easy to understand and implement, but require more than linear time ([1]).

In this paper, a new linear-time algorithm for embedding maximal planar graphs is proposed. This algorithm is both easy to understand and easy to implement. The algorithm consists of two phases, both of which use only simple, local graph-modifications.

In addition to planar embedding, the new algorithm allows to test graphs for maximal planarity. We will also demonstrate how to generate *random (maximal) planar graphs* using this algorithm. The algorithm proposed constitutes a first step towards a simple, linear-time solution for embedding general planar graphs.† A full version of this abstract may be found in [5]. Notice that our results make use of a seminal paper dealing with *planar 3-bounded orientations* by Chrobak and Eppstein [2].

One of the referees made us aware of a paper by R.C. Read [4].‡

Basic definitions

The terminology used in this paper is adopted from [3]. Let $G = (V, E)$ be a planar graph, and $ADJ[v]$ for all $v \in V$ denote the adjacency lists of G. An *embedding* of G is defined as an ordering of the adjacency lists of G, such that for each vertex $v \in V$ the order of the edges in $ADJ[v]$ corresponds to a counter-clockwise traversal of the edges in any fixed *embedding of G in the plane*. A simple planar graph is called *maximal planar*, if the addition of any new edge results in a non-planar graph. The *smallest maximal planar graph* w.r.t. the number of vertices is defined as the complete (maximal planar) graph on 4 vertices K_4. Thus, a maximal planar graph does not contain any vertex of degree less than 3. This is because such graphs are triconnected. The triconnectivity also enforces a unique embedding in the plane w.r.t. a chosen outer face.

A key concept for the new algorithm is the notion of *reducible vertices*:

Definition 1 A vertex v of $G = (V, E)$ will be called *small*, if $deg(v) < 18$, otherwise it will be called *large*. A vertex $v \in V$ is *reducible* if it satisfies one of the following conditions: (a) $deg(v) \leq 3$ or (b) $deg(v) = 4$ and v has at least 2 small neighbors, or (c) $deg(v) = 5$ and v has at least 4 small neighbors.

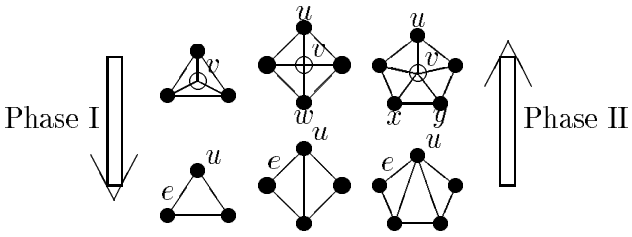
*Institut für Informatik III, Universität Bonn. hermann@holmium.informatik.uni-bonn.de

†This goal has been recognized as a siginificant open problem in [1].

‡His methods are similar to those described here, but they need a (topological) embedding of the input graph, and they compute a *straight line drawing* of it in linear time and quadratic space. Our algorithm runs in linear time and linear space, determines the (topological) embedding and can additionally incorporate Read's coordinate determination naturally to result in an *embedding in the plane*

Lemma 1 Each planar graph G has at least 4 reducible vertices.

Definition 2 The *reduction* of a reducible vertex v in a maximal planar graph G is obtained by the following process:
If $deg(v) = 3$, then simply remove v from G.
If $deg(v) \in \{4, 5\}$ then let $u \in N(v)$ be a vertex with exactly two neighbors in $N(v)$. Now remove v and add new edges $\{u, x\}$ to G for all $x \in (N(v) - (\{u\} \cup N(u)))$.
As described below, this process is the basis for the first phase of the proposed algorithm and is illustrated in the figure on the right.



Lemma 2 Let $G = (V, E)$ be a maximal planar graph with $|V| > 4$, and let v be any reducible vertex of G. The reduction of v in G results in G', which is again a maximal planar graph.

Definition 3 The *inverse reduction* for vertex v and graph G' is the operation inverse to the reduction of v in G. Inverse reductions are applied during the second phase, as illustrated in the figure.

Lemma 3 The *inverse reduction* for vertex v in an embedding of G' leads to an embedding of G.

The Embedding Algorithm

Both the *reduction* and the *inverse reduction* operator take constant time. This speed, however, requires special data structures omitted here. Notice that updating the set of reducible vertices takes constant time for each single reduction, too [5].

```
In:   maximal planar graph G = (V, E).
Out:  embedding of G.                    /* ordering of adjacency lists */
Proc: set of vertices R;                 /* initially containing all reducible vertices of G */
      stack of list of vertices S;       /* initially empty */
      while (|V| > 4) do            {    choose reducible vertex v from R;
                                         push {v ∪ N(v)} on S;
                                         do reduction of v in G;
                                         update R to contain all reducible vertices of G again; }
      embed the resulting G;             /* G is now the K4 */
      while (S is not empty) do     {    pop v and N(v) from S;
                                         do the inverse reduction of v in (the embedding of) G; }
```

The main result of this paper is given by the following theorem.

Theorem 1 Each operation of the above algorithm can be done in constant time. Hence the algorithm has linear time complexity.

Since both the reduction and the inverse reduction are considerably simple, the resulting algorithm is easy to understand and to implement. To our knowledge the algorithm is the first *simple* algorithm with linear time complexity.

Extensions of the Basic Algorithm

The proposed algorithm can be easily extended to an algorithm which tests whether a graph is maximal planar. The necessary modifications include some additional tests. The resulting algorithm still operates in linear time.

A slightly modified version of this algorithm can also be used to generate *random maximal planar graphs* of n vertices. More specifically, starting with an embedding of $G = (V, E) = K_4$, the following step has to be repeated $n - 4$ times:

> Choose an edge e of G, a vertex $u \in e$ and a type of reverse reduction $t \in \{3, 4, 5\}$ randomly. Do the inverse reduction of a new vertex v at edge e and vertex u of type t (see above figure).

Notice that the resulting graph G is an embedding. An embedding of a random planar graph with n vertices and m edges can be obtained by deleting $3n - 6 - m$ edges randomly from the random maximal planar graph G on n vertices mentioned above. To the best of our knowledge all existing algorithms for generating *random maximal planar graphs* consist exclusively of reduction type 3.

References

[1] Battista, G.D., Eades, P., Tamassia, R., Tollis, I.G., 'Algorithms for Drawing Graphs: an Annotated Bibliography', /pub/gdbiblio.tex.Z from wilma.cs.brown.edu .

[2] Chrobak, M., Eppstein, D., 'Planar orientations with low out-degree and compaction of adjacency matrices', Theor. Comp. Sci., **86**, 243-266 (1991).

[3] Even, S., Graph Algorithms, Computer Science Press, 1979.

[4] Read, R.C., 'A new method for drawing a planar graph given the cyclic order of the edges at each vertex', Congressus Numerantium, **56**, 31-44 (1987).

[5] Stamm-Wilbrandt, H., 'A Simple Linear-Time Algorithm for Embedding Maximal Planar Graphs', Technical report IAI-TR-93-5, Institut für Informatik III, Universität Bonn.

The Left-Right Algorithm for Planarity Testing and Embedding

H. de Fraysseix and P. Rosenstiehl.

Abstract not Available.

Connexity of Bipolar Orientations

P. O. de Mendez

Abstract not Available.