アルゴリズム 12 − 2
(1989. 11. 20)

# Complexity Issues in Drawing Directed Graphs

Peter Eades
Geometric Algorithms Laboratory
Department of Computer Science, University of Queensland
St. Lucia Queensland Australia 4067

## Abstract

We review a well established methodology for drawing directed graphs nicely, that is, so that they are easy to understand and remember. The methodology inspires several interesting questions about the algorithmic tools required. This paper poses some of these questions, and comments on progress toward answers.

## 1. Introduction

Many recent development tools for software engineering and programming involve the representation of entities and their relationships as directed graphs. The availability of inexpensive but powerful graphics workstations has lead to systems which allow the user to visually display and manipulate these directed graphs. The usefulness of this visualisation depends on the layout of the graph: a "nice" drawing can be a great aid to a software designer, but a poor drawing can be confusing or misleading. Thus there has been considerable interest in *aesthetic layout algorithms*, that is, algorithms for drawing directed graphs so that they are easy to understand and remember. An annotated bibliography of these algorithms is available [6].

The current interest has mainly been from practitioners who are more concerned with the architecture of the visualisation process than the complexity of the algorithms. In this paper we outline some of the complexity issues which arise in aesthetic layout of directed graphs.

A general method for drawing directed graphs according to the four aesthetic criteria listed below has been developed by a number of authors (for example, [14], [16], [19]) and has been integrated into several graph drawing systems.

$C_1$: *All arcs should follow the same general direction.*

$C_2$: *Vertices should be distributed evenly over the page.*

$C_3$: *There should be as few arc crossings as possible.*

$C_4$: *Arcs should be as straight as possible.*

The general method is a "successive refinement" technique: a drawing is refined according to each criterion in turn, from $C_1$ to $C_4$. There are four steps $S_1$, $S_2$, $S_3$, $S_4$, where $S_i$ attempts to fix those aspects of the drawing which relate to criterion $C_i$, while leaving other aspects of the layout free.

Each of these steps attempts to optimise some quantifications of the associated criterion. By stating these quantifications in precise terms, we can show that for each criterion there is at least one optimisation problem which is NP-hard. We examine the complexity of some of the optimisation problems arising from the four steps.

Fast algorithms are essential for interactive graphics; this fact, together with the NP-hardness results mentioned above, justifies *heuristic* implementations of Steps $S_1$ to $S_4$. Many such algorithms are available, but they are mostly proposed by practitioners who are not concerned with rigorous analysis. In this paper we review mathematical results such as "performance guarantees" for the performance of these heuristics, and indicate areas where analysis is yet to be done.

As an aside, we introduce a new heuristic algorithm for Step $S_3$. Basically, by solving a system of linear equations we can ensure that the $x$ coordinate of each vertex is at a weighted average of the $x$ coordinates of its neighbors. We can show that the drawing so obtained has several desirable properties besides avoiding crossings: for instance, it displays some symmetry.

In the last section some brief remarks are made about drawing *planar* digraphs.

## 2. The Four Steps

### 2.1. Step $S_1$: Remove Cycles

An acyclic digraph can be drawn so that all arcs are monotonic in the same direction (say downward). For a digraph with cycles, we need to reverse some arcs so that it becomes acyclic.

A set $R$ of arcs in a digraph $G = (V, A)$ is a *feedback arc set* if the reversal of every arc in $R$ makes $G$ acyclic. In Step $S_1$, we choose a feedback arc set $R$, and form the acyclic digraph $G'$ from $G$ by reversing the arcs of $R$. Steps $S_2$, $S_3$, and $S_4$ can then be applied to obtain a drawing of $G'$ with all arcs pointing downward. If we then reverse the arcs of $R$ again, we get a drawing in which all arcs except those of $R$ are pointing downward. Of course we want $|R|$ to be as small as possible.

We denote the smallest size of a feedback arc set of $G$ by $r(G)$.

Unfortunately, the problem of computing $r(G)$ is NP-complete ([8]), even for graphs which are quite sparse. (For instance, it is NP-complete for orientations of graphs of maximum degree 3.) This justifies heuristics.

We would like a heuristic with a proven performance bound $K$; that is, we would like to answer the following problem.

**Problem 1: Feedback Arc Set Approximation**
*INSTANCE:* A digraph $G$.
*OUTPUT:* A feedback arc set $R$ of $G$ with $|R| \leq Kr(G)$.

We would like to know whether Problem 1 is NP-hard.

Some results are presented in [4]. However, the results of [4] concentrate on "tournaments" (orientations of complete graphs). In practice, tournaments are seldom drawn; most digraphs required in diagrams are reasonably sparse. It would be particularly interesting to determine the complexity of Problem 1 on sparse digraphs (for example, where $|A| < c|V|$ for some constant $c$).

### 2.2. Step $S_2$: Layering

A *layering* of a digraph $G = (V, A)$ is a partition of $V$ into subsets $L_1, L_2, \cdots, L_h$, such that if $u \to v \in A$ where $u \in L_i$ and $v \in L_j$ then $i > j$. A digraph with a layering is a *layered network*. The *height* of the network is the number $h$ of layers; the *width* is $w = \max_{1 \leq m \leq h} |L_m|$.

Such networks are conventionally drawn so that all vertices in layer $L_m$ lie on the horizontal line $y = m$, as in Figure 1.

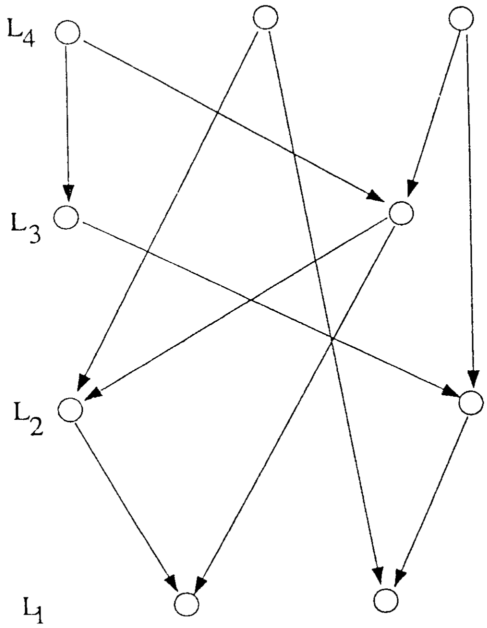

*Figure 1: a layered network*

Note that a layered network must be acyclic, and the drawing convention ensures that all arcs point downward.

The *span* of an arc $u \rightarrow v$ with $u \in L_i$ and $v \in L_j$ is $i-j$. The network is *proper* if no arc has a span greater than one.

Step $S_2$ constructs a proper layered network from an acyclic digraph while attempting to keep the vertices spread evenly over the page.

If we assume that each vertex is approximately the same size, and requires a minimal separation distance to the next vertex in both horizontal and vertical directions, then we can measure the width of the drawing as the width of the layering, and the height of the drawing as the height of the layering. Thus to achieve criterion $C_2$ we need a layering of given width $w$ and height $h$, where $w$ and $h$ measure the dimensions of the page to be used. Thus we are interested in the following problem.

**Problem 2: Minimum Size Layer Assignment**
*INSTANCE:* An acyclic digraph $G$, integers $W$ and $H$.
*OUTPUT:* A layering of $G$ with width at most $W$ and height at most $H$.

Unfortunately, Problem 2 is NP-hard, for the following reason. Suppose that each vertex of an acyclic digraph $G$ represents a task to be performed on one of the processors of a multiprocessor. One unit of time is required for each task. The arcs of $G$ represent precedence constraints between the tasks. The *multiprocessor scheduling problem* is to assign the tasks to a set of $w$ processors so that all tasks are completed in time $h$. This can be done if and only if a layering of width $w$ and height $h$ can be found. Thus the multiprocessor scheduling problem is essentially the same as the layer assignment problem.

The layered network must be made proper, because Step $S_3$ below assumes that there are no arcs with span greater than one. (because it is difficult to handle crossings between long arcs). Thus we replace each such arc $u \rightarrow v$ with a path $u = v_0 \rightarrow v_1 \rightarrow \cdots \rightarrow v_{j-i} = v$, adding the dummy vertices $v_1, v_2, \cdots, v_{j-i-1}$. In Figure 2, dummy vertices have been added to the network of Figure 1.

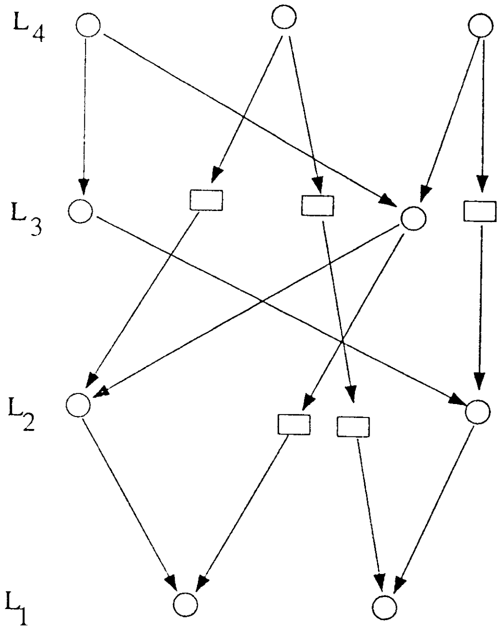

*Figure 2: dummy vertices*

Dummy vertices cause problems for three reasons. Firstly, the time used by step $S_3$ depends on the total number of vertices, dummy plus real. More vertices means more time, thus it is desirable to reduce the number of dummy vertices. Secondly, bends in the arcs in the final drawing occur only at dummy vertices. Although some straightening can be achieved by other means, again it is desirable to alleviate the problem by reducing the number of dummy vertices. Finally, short arcs are easier to follow than long arcs.

The number of dummy vertices for a layering $L_1, L_2, ..., L_h$ is $\sum(i-j)$, where the sum is over all arcs $u \rightarrow v$ with $u \in L_i$ and $v \in L_j$. This quantity is called the *total length* of the layering.

One can use total length as the the optimisation goal for the layering. If so, then we are interested in the following problem:

**Problem 3: Dummy Vertex Minimisation**
*INSTANCE:* An acyclic digraph $G$.
*OUTPUT:* A layering of $G$ with minimal total length.

Surprisingly, Problem 3 can be solved efficiently by linear programming techniques (see [10]). It would be interesting to know how close a solution to Problem 3 is to a solution to Problem

2; that is, whether the linear programming technique can be used as a heuristic for Problem 2.

## 2.3. Step $S_3$: Reduce Crossings

The number of arc crossings in a drawing of a proper layered network does not depend on the precise position of vertices but only on the ordering of the vertices within each layer. Thus the problem of reducing arc crossings is the combinatorial one of choosing an appropriate ordering for each layer, not the geometric one of choosing an $x$-coordinate for each vertex. Although this combinatorialisation considerably simplifies the problem, it is still difficult: the proof method of [9] implies that the problem of minimising arc crossings in a layered network is NP-hard, even if there are only two layers.

A kind of "layer-by-layer sweep" heuristic can be applied as follows. First, an ordering of layer $L_1$ is chosen. Then for $i=2,3,...,h$, the ordering of layer $L_i$ is held fixed while re-ordering layer $L_{i+1}$ to reduce crossings between layer $L_{i+1}$ and layer $L_i$.

There are several variations of this basic method (see [17], for example), but each variation presupposes a solution to the a problem of the following form: given a fixed ordering of layer $L_i$, choose an ordering of layer $L_{i+1}$ to minimise the number of arc crossings. See Figure 3.

The minimum number of crossings for a 2-layered network $G$ with a fixed order for the lower layer is denoted by $opt(G)$. Thus we want to solve the following problem:

**Problem 4: Two-Layered Crossing Minimisation**
*INSTANCE*: A layered network $G$ with two layers $L_0, L_1$, and an ordering of $L_0$.
*OUTPUT*: An ordering of $L_1$ which gives $opt(G)$ crossings.

Unfortunately, this Problem 4 is NP-hard [8].

Some effective heuristics are available for Problem 4. Two of these, called the "barycenter" and "median" heuristics, are based on the intuition that to avoid crossings, each vertex should be "close" to the vertices to which it is attached.

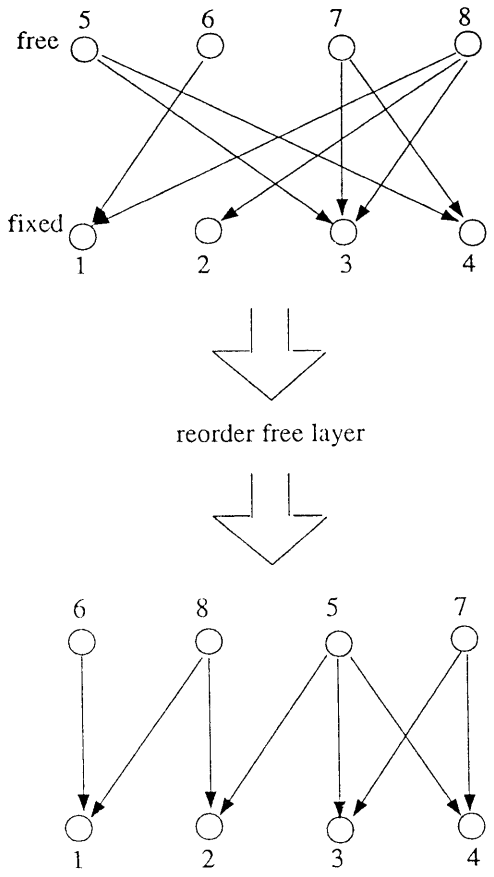

*Figure 3: fixed and free layers*

In the *barycenter* heuristic [19], for each vertex $v$ we compute the average of the $x$ coordinates the vertices which are adjacent to $v$. We denote the number of crossings given by this algorithm on a graph $G$ by $bar(G)$.

In the *median* heuristic [8], we compute the $x$ coordinate of vertex $v$ as the median of the $x$ coordinates the vertices which are adjacent to $v$. Both are efficient (linear time and space) and very simple. The number of crossings given by the median heuristic is denoted by $med(G)$.

Extensive tests ([11], [13]) suggest that both methods are effective, at least for the restricted problem. The following Theorems (from [8]) give some analytic support to the experience.

**Theorem 1:** $\frac{med(G)}{opt(G)} \leq 3$.

**Theorem 2:** $\frac{bar(G)}{opt(G)}$ is $O(\sqrt{n})$.

It can be shown (see [8]) that both $med(G)$ and $bar(G)$ become arbitrarily close to $opt(G)$ as the density of $G$ increases, that is, as $|E| \rightarrow |V|^2$.

It is not clear whether assurances such as the Theorems above can be given for the more general problem with more than two layers. We would like to know the complexity of the following problem.

**Problem 5: Crossing Minimisation Approximation**

*INSTANCE*: A proper layered network $G$ with layers $L_0, L_1, \cdots, L_{m-1}$.

*OUTPUT*: An ordering of each layer such that the number $r$ of crossings satisfies $r \leq K\omega$, where $\omega$ is the minimum number of crossings and $K$ is a constant.

Even a partial solution to this problem would be interesting. For instance:

- with the input restricted in various ways (for example, to sparse graphs), or
- with $K$ a small function of the size of $G$, (for example, $K = O(\log|V|)$).

A possible candidate for such an algorithm is described in Section 3 below.

## 2.4. Step $S_4$: Straighten Arcs

Bends in arcs occur at the dummy vertices introduced at step $S_2$. It is desirable to reduce the angle of such bends by choosing an $x$-coordinate for each vertex, without disturbing the ordering created at step $S_3$. Sugiyama [16] has gone further, in attempting to make arcs as close to vertical lines as possible, subject to the ordering constraints. It is possible to state this problem as an optimisation problem:

$$\text{minimise} \sum_{u \rightarrow v} (x(u) - x(v))^2,$$

subject to linear constraints which preserve the ordering and a minimal horizontal distance between vertices. Unfortunately, it is not clear how this quadratic programming problem can be solved efficiently.

A heuristic approach, based on a "layer-by-layer sweep" paradigm, is presented in [16] and refined in [10].

## 3. A New Method for Step $S_3$

W. T. Tutte proposed the following method for drawing triconnected planar graphs. First choose an outer face $F$, and draw $F$ as a regular polygon. For every vertex $v$ not on $F$, place $v$ at the barycenter (average) of its graph-theoretic neighbours.

A variation of this method can be used to draw a proper layered network $G$ with layers $L_1, L_2, \cdots, L_h$ as follows.

*Algorithm Tutte*

(1) Choose two vertices $u_1, v_1$ from $L_1$, and two vertices $u_h, v_h$ from $L_h$.

(2) Fix the $x$-coordinates of the vertices in layers $L_1$ and $L_h$.

(3) For every other vertex $u$ with indegree $d^-(u)$ and outdegree $d^+(u)$, assign $x$-coordinate $x(u)$ to $u$, where

$$x(u) = \frac{\left[\sum_{u \leftarrow v} x(v)\right]}{2d^-(u)} + \frac{\left[\sum_{u \rightarrow w} x(w)\right]}{2d^+(u)}.$$

A connectivity precondition is needed for this algorithm to work; the following conditions are sufficient:

- all sinks (vertices of indegree 0) are in layer $L_h$, and all sources (vertices of outdegree 0) are in layer $L_1$, and
- the graph formed by connecting all sinks to a single new sink and all sources to a single new source is triconnected.

The equations at step 3 are linear and, in practice, quite sparse; they can be solved by a number of numerical techniques.

There are several several interesting properties of the layout computed by this algorithm. For instance, it can be shown that the algorithm can display some automorphisms of the graph as symmetries of the drawing.

In a sense, *Algorithm Tutte* is similar to the layer by layer sweep described above when the barycenter heuristic is used at each layer. It would be interesting to know whether any theorem of the form of Theorem 2 could be obtained for *Algorithm Tutte*.

## 4. Drawing Planar Digraphs

Finally, we make some remarks on drawing planar digraphs. The approach to this problem is quite different from the more general problem.

Firstly note that the feedback arc set problem has a polynomial time solution for planar graphs. Thus Step $S_1$ above can be achieved and we can restrict out attention to planar acyclic digraphs.

A digraph $G$ is *downward planar* if it is possible to draw $G$ so that all arcs point downward and no arcs cross. Not every planar acyclic digraph is downward planar: see Figure 4.

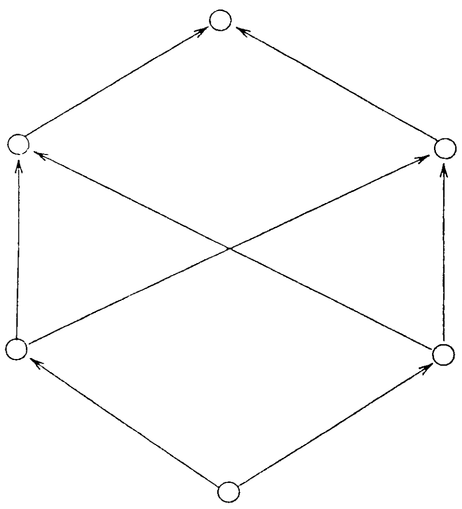

*Figure 4: planar acyclic digraph which is not downward planar*

There is a recent blossoming of literature on the theory of downward planar digraphs. For example, such graphs can be drawn with straight line arcs in time $O(n \log n)$: see [3]. Unfortunately the complexity of following problem is yet to be determined:

**Problem 6: Downward Planarity Test**
*INSTANCE*: An acyclic planar digraph $G$.
*QUESTION*: Is $G$ downward planar?

*Note*: this problem is often called the *upward planarity test* problem.

It is easy to see that for any digraph there is a set of arcs whose reversal makes $G$ downward planar. The complexity of following generalisation of the Problem 6 has not been determined:

**Problem 7: Downward Planar Feedback Arc Set**
*INSTANCE*: A planar digraph $G$.
*OUTPUT*: Find a minimum cardinality set $R$ of arcs of $G$ whose reversal makes $G$ downward planar.

## References

[1] C. Batini, G. Di Battista and R. Tamassia, "Automatic Graph Drawing and Readability of Diagrams", *IEEE Trans. on Systems, Man and Cybernetics* SMC-18 (1) (1988), 61-79.

[2] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*, Macmillan 1976.

[3] G. Di Battista and R. Tamassia, "Algorithms for Plane Representations of Acyclic Digraphs", *Theoretical Computer Science* 61 (1988), 175-198.

[4] P. Eades, X. Lin and W. Smyth, "Heuristics for the Feedback Arc Set Problem", Technical Report, Department of Computer Science, Curtin University of Technology, Perth, Australia.

[5] P. Eades, B. D. McKay and N. C. Wormald, "On and Edge Crossing Problem", *Proceedings of the 9th Australian Computer Science Conference*, Australian National University 1986, 327-334.

[6] P. Eades and R. Tamassia, "Algorithms for Drawing Graphs: an Annotated Bibliography", Tech. Report CS-89-09, Department of Computer Science, Brown University.

[7] P. Eades and N. C. Wormald, "Edge Crossings in Drawings of Bipartite Graphs", *Technical Report* 108, Department of Computer Science, University of Queensland 1989.

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

[9] M. R. Garey and D. S. Johnson, "Crossing Number is NP-Complete", *SIAM J. of Algebraic and Discrete Methods* 4 (3) (1983), 312-316.

[10] E. R. Gasner, S. C. North, K. P. Vo, "DAG - A Program that Draws Direct Graphs", manuscript.

[11] D. Kelly, *A View to Graph Layout Problems*, Masters Thesis, University of Queensland 1987.

[12] C. L. Lucchesi and D. H. Younger, "A Minimax Relation for Directed Graphs", *Journal of the London Mathematical Society* (2) 17 (1978), 369-374.

[13] E. Makinen, "Experiments in Drawing 2-Level Hierarchical Graphs", Report A-1988-1, Department of Computer Science, University of Tampere.

[14] L. A. Rowe, M. Davis, E. Messinger, C. Meyer, C. Spirakis, and A. Tuan, "A Browser for Directed Graphs", *Software Practise and Experience* 17 (1) (1987), 61-76.

[15] K. Sugiyama, "A Readability Requirement in Drawing Digraphs: Level Assignment and Edge Removal for Reducing the Total Length of Lines", Research Report 45, (1984) International Institute for Advanced Study of Social information Science, Fujitsu Ltd.

[16] K. Sugiyama, "A Cognitive Approach for Graph Drawing", *Cybernetics and Systems*, 18 (1987) 447-488.

[17] H. Trickey, "DRAG: A Graph Drawing System", *Proceedings of the Int. Conference on Electronic Publishing, Document Manipulation, and Typography*, Nice (1988), 171-182.

[18] W. Tutte, "How to Draw a Graph", *Proceedings of the London Mathematical Society*, 3 (13) (1963), 743-768.

[19] J. Warfield, "Crossing Theory and Hierarchy Mapping", *IEEE Trans. on Systems, Man and Cybernetics*, SMC-7 (7) (1977), 502-523.