International Journal of Computational Geometry & Applications
© World Scientific Publishing Company

AN ALGORITHM FOR DRAWING A HIERARCHICAL GRAPH

PETER EADES
Department of Computer Science, University of Newcastle
University Drive, Callaghan, NSW 2308, Australia

and

XUEMIN LIN
Department of Computer Science, University of Western Australia
Nedlands, WA 6009, Australia

and

ROBERTO TAMASSIA
Department of Computer Science, Brown University
Box 1910, Providence, Rhode Island 02912, USA

Received (received date)
Revised (revised date)
Communicated by Editor's name

1. INTRODUCTION

Graph layout algorithms are increasingly used in graphic user interfaces for visualization for information and software engineering[9,14,15]. The quality of the layout is the key to the success of the model: a nicely drawn graph is worth a thousand words, but a poor layout can be misleading.

Of course the quality of a layout is a subjective matter, but several aesthetic criteria are generally accepted. For instance, edge crossings should be avoided, and a symmetric drawing is desirable where possible. Other aesthetic criteria include the minimization of the number of bends in the edges[16], the minimization of the number of different slopes used by edges[3], and the minimization of the variance in the edge length[8]. In this paper we present a method for drawing "hierarchical directed graphs", which are digraphs in which each node is assigned a layer, as in Figure 1.

Hierarchical graphs appear in several graph drawing applications, where nodes are assigned layers for semantic reasons. More importantly, general methods for drawing directed graphs usually begin by transforming the input digraph into a hierarchical graph, then applying a hierarchical graph drawing algorithm[6,9,14].

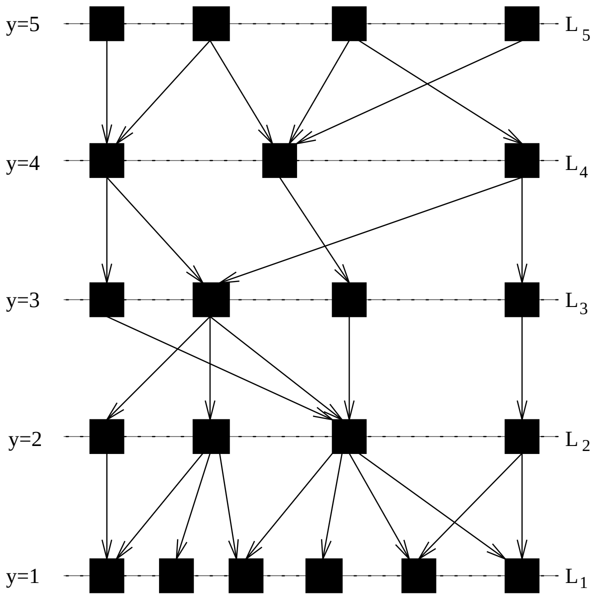

Fig. 1. A hierarchical directed graph

Our layout algorithm satisfies several important aesthetic criteria. For instance, it produces planar and symmetric drawings whenever such drawings are possible.

The algorithm is described in Section 3. The two properties above are precisely defined and proved in Sections 4 and 5. Our algorithm is related to some layout heuristics, originally due to Sugiyama *et al.*[15], which seek to form a drawing with a small number of crossings; in Section 6 we briefly describe the relationship, and report on the performance of our algorithm as a crossing minimization heuristic.

The planarity property proved in Section 4 holds for a restricted class of hierarchical graphs. In fact our algorithm works best for hierarchical graphs which are not too dense. For dense graphs, other techniques such as *edge concentration* [13] are more suitable. In the final section we discuss the limitations of our algorithm, and show how to extend the results of Section 4 to a slightly more general class of graphs.

2. TERMINOLOGY

Standard graph theoretic terminology[2] is used.

A *hierarchical graph* $H = (V, A, \lambda, k)$ consists of a directed graph (V, A), a positive integer k, and, for each node u, an integer $\lambda(u) \in \{1, 2, ..., k\}$, with the property that if $(u, v) \in A$, then $\lambda(u) > \lambda(v)$. For $1 \leq i \leq k$ the set $\{u : \lambda(u) = i\}$ is the ith *layer* of H and is denoted by L_i. Note that a hierarchical graph is acyclic.

The *span* of an arc (u, v) is $\lambda(u) - \lambda(v)$. An arc of span greater than one is *long*; and a hierarchical graph with no long arcs is *proper*. For most of this paper we deal with proper hierarchical graphs; in Section 7 we show how the results apply to graphs with long arcs.

The layers L_1 and L_k are the *boundary layers* of H. In many applications, all sources and sinks are in L_k and L_1 respectively; in this case H is a *boundary s-t graph.*

A hierarchical graph is conventionally drawn with layer L_i on the horizontal line $y = i$ as in Figure 1. If the graph is proper, then arcs are drawn as closed line segments. Thus a *drawing* of a proper hierarchical graph is a $1-1$ mapping α which assigns a point $\alpha(u) = (\alpha_x(u), \alpha_y(u))$ to each node $u \in V$. The drawing convention implies that $\alpha_y(u) = \lambda(u)$ and so effectively the only role of a drawing algorithm is to choose $\alpha_x(u)$.

A drawing is *planar* if no pair of nonincident arcs intersect. A planar drawing is *convex* if every internal face is a convex polygon. A hierarchical graph is *hierarchically planar* if it has a planar (hierarchical) drawing.

In the following it is often convenient to discussing mappings $\alpha : V \rightarrow R^2$ which are not necessarily $1-1$; such a mapping will be called a *sketch.* A *boundary sketch* β of H consists of an x-coordinate $\beta_x(u)$ for each node u in the boundary layers of H. A sketch α of H *extends* the boundary sketch β if $\alpha_x(u) = \beta_x(u)$ for each u in the boundary layers of H. Similarly we can define a *boundary drawing* and an *extension* of a boundary drawing. Our algorithm essentially assumes that nodes in the boundary layers have fixed positions; thus we say H is *hierarchically planar with respect to* β if it has a planar drawing which extends β.

Next we define a connectivity condition for proper boundary $s-t$ graphs; this condition generalizes 3-connectivity.

In a proper boundary $s-t$ graph H, a pair $\{u, v\}$ of distinct nodes L_i is *collapsible* if u and v are in the same nonboundary layer L_i, and there are two nodes $a \in L_j$ and $b \in L_l$ such that $j > i > l$, and every directed path from a sink to a source containing u always passes through a and b, and also every directed path from a sink to a source containing v always passes through a and b. Note that, intuitively, by replacing every arc in H by an elastic band, we have that if $\{u, v\}$ is collapsible, then by pulling a away from b, all the directed paths from a to b containing either u or v would collapse together.

A proper boundary $s-t$ graph H is *collapse free* if there are no multiple arcs and no collapsible pairs. If $\{u, v\}$ is collapsible, then the pair $\{a, b\}$ mentioned above is a *separation pair* of the underlying undirected graph G of H; thus if G has no multiple edges and is 3-connected, then H is collapse free.

We denote the indegree of u by d_u^- and the outdegree of u by d_u^+.

3. THE ALGORITHM

Our algorithm is inspired by an algorithm proposed by Tutte[17,18] for drawing triconnected planar graphs. For a proper boundary $s-t$ graph $H = (V, A, \lambda, k)$ we first choose a boundary drawing β for H, for example, by spacing L_1 and L_k equally on the lines $y = 1$ and $y = k$. Then we choose the x coordinate $x(u)$ for each nonboundary node u by placing u at a kind of weighted barycentre of its neighbours as follows:

$$x(u) = \frac{1}{2d_u^-} \sum_{(v,u)} x(v) + \frac{1}{2d_u^+} \sum_{(u,v)} x(v). \quad (1)$$

On the right hand side of these equations, the symbol $x(v)$ is $\beta_x(v)$ if v is a boundary node, and otherwise represents a variable for the x coordinate of v. Thus (1) is a system of m equations in m unknowns, where m is the number of nonboundary nodes. We will write the system as

$$Mx = b \quad (2)$$

where x is the (unknown) vector of x coordinates of nonboundary nodes. Note that the coefficient matrix M is closely related to the adjacency matrix of the graph: it has nonzero entries only on the diagonal, and at the same places as the adjacency matrix.

We show below that the system has a unique solution.

The output of our algorithm, which we shall call the *Degree Weighted Barycentre (DWB)* algorithm, is the sketch defined by the solution to the equations (1).

Note that the equations (1) are different from the equations used by Tutte[17,18]; in fact, the theorems in Sections 4 and 5 below would not hold for an algorithm that directly applied Tutte's equations to hierarchical graphs. Further, these results cannot be obtained by the method of Becker and Hotz[1].

A sample output of algorithm DWB is in Figure 2.

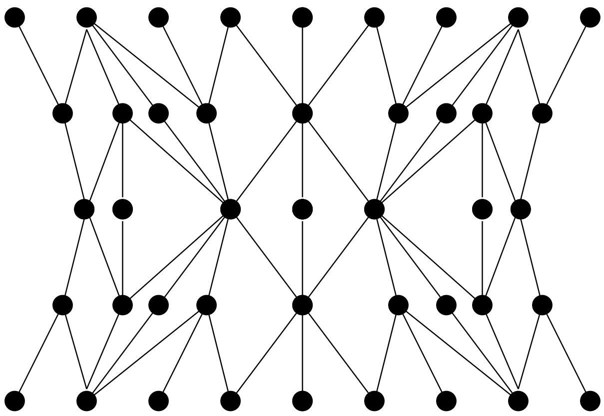

Fig. 2. A sample output of algorithm DWB

First we show that the system of equations has a unique solution.

Theorem 1 *If $H = (V, A, \lambda, k)$ is a proper boundary $s{-}t$ graph and β is a boundary sketch of H, then the equations (1) have a unique solution extending β.*

Proof. We only need to prove that if every sink and source v has been assigned $\beta_x(v) = 0$, then the unique solution of the equations (1) is $x(u) = 0$ for each u.

For each $i \in \{1, 2, ...k\}$, let m_i denote the maximal value of $|x(u)|$ over all $u \in L_i$. One can easily infer from equations (1) that, for $2 \leq i \leq k-1$,

$$m_i \leq \frac{1}{2}(m_{i-1} + m_{i+1}).$$

Since $m_1 = 0$ from the boundary conditions, we can deduce that, for $2 \leq i \leq k-1$,

$$m_i \leq \frac{i-1}{i} m_{i+1}. \tag{3}$$

Also, $m_k = 0$, and it follows that

$$m_{i+1} \leq \frac{k-i-1}{k-i} m_i. \tag{4}$$

The only solution to (3) and (4) is $m_i = 0$ for $2 \leq i \leq k-1$. □

An implementation of algorithm DWB could use Gaussian elimination to solve the equations (1), with an asymptotic time complexity as good as the best matrix multiplication algorithm. In practice, we have found that the following simple Gauss - Seidel iteration is quite fast:

Algorithm **DWB_GSITERATION**

1. Choose initial x coordinate $x(u)$ for each nonboundary node u;
2. Repeat the following until x converges:
   for $i := 2$ *to* $k-1$ *do*
   for each $u \in L_i$ *do* $x(u) := \frac{1}{2d_u^-} \sum_{(v,u)} x(v) + \frac{1}{2d_u^+} \sum_{(u,w)} x(w)$

The following equations characterize the iteration above:

$$x^{(j)}(u) = \frac{1}{2d_u^-} \sum_{(v,u)} x^{(j)}(v) + \frac{1}{2d_u^+} \sum_{(u,w)} x^{(j-1)}(w). \tag{5}$$

Here $x^{(0)}$ is the initial position and the value of x after j iterations of the main (outside) loop is $x^{(j)}$; if u is in the boundary, then $x^{(j)}(u) = x^{(j-1)}(u) = \beta_x(u)$.

The equations (5) are used in Sections 4 and 5 to prove properties of algorithm DWB.

Lemma 1 *The iteration (5) converges to the solution of (1).*

Proof. Define $\Delta^{j,i}$ and Δ_j by:

$$\begin{aligned} \Delta^{j,i} &= \max\{|x^{(j)}(u) - x^{(j-1)}(u)| : u \in L_i\}, \\ \Delta_j &= \max\{\Delta^{j,i} : 2 \leq i \leq k-1\}. \end{aligned}$$

Using (5) and the same kind of reasoning as for Theorem 1, one can show that for $j \geq 1$ and $2 \leq i \leq k-1$,

$$\Delta^{j,i} \leq \frac{2^{i-1}-1}{2^{i-1}} \Delta_{j-1},$$

and it follows that

$$\begin{aligned} \Delta_j &\leq \frac{2^{k-3}-1}{2^{k-3}} \Delta_{j-1} \\ &\leq \left(\frac{2^{k-3}-1}{2^{k-3}}\right)^{j-1} \Delta_1. \end{aligned}$$

Thus $\Delta_j \to 0$ as $j \to \infty$ and (5) converges to the solution of (1). $\Box$

4. PLANARITY AND CONVEXITY

In this section we prove the following theorem.

Theorem 2 *Suppose that a proper boundary $s-t$ graph $H = (V, A, \lambda, k)$ is collapse free. If H is hierarchically planar with respect to the boundary drawing β, then the drawing output by algorithm DWB is planar and convex.*

We need the following concepts to prove the theorem. Given two points $p_1 = (x_1, y)$ and $p_2 = (x_2, y)$, p_1 is *on the left (right)* of p_2 if $x_1 \le x_2$ ($x_1 \ge x_2$). If $x_1 < x_2$ ($x_1 > x_2$), then we say p_1 is *strictly on the left (right)* of p_2. Note that p_1 and p_2 are on the same horizontal line.

Suppose that a proper boundary $s-t$ hierarchical graph $H = (V, A, \lambda, k)$ is hierarchically planar, and a drawing α of H is hierarchically planar. A sketch $\overline{\alpha}$ of H *preserves the topology* of α if for each $i \in \{1, 2, ..., k\}$ and each pair $\{u, v\}$ of nodes in L_i, $\overline{\alpha}(u)$ is on the left of $\overline{\alpha}(v)$ whenever $\alpha(u)$ is strictly on the left of $\alpha(v)$. Note that a hierarchically planar drawing preserves the topology of itself.

The following lemma is simple but useful.

Lemma 2 *Suppose that a proper boundary $s-t$ hierarchical graph $H = (V, A, \lambda, k)$ is hierarchically planar, and a drawing α of H is hierarchically planar. Further, suppose that a drawing $\overline{\alpha}$ of H preserves the topology of α. Then $\overline{\alpha}$ is hierarchically planar.*

Proof. It is obvious that $\overline{\alpha}$ and α induce the same ordering of the nodes in the same layer of H. Thus $\overline{\alpha}$ and α have the same number of arc crossings. $\Box$

Consider the Gauss-Seidel iteration (5), and define a sketch $\alpha^{j,i}$ by:

$$\alpha_x^{j,i}(u) = \begin{cases} x^{(j)}(u), & \lambda(u) \le i \\ x^{(j-1)}(u), & \lambda(u) > i \end{cases}$$

for $i \in \{1, 2, \ldots, k\}$. Intuitively, $\alpha^{j,i}$ is the sketch after j iterations of the main loop and i iterations of the inside loop of the iteration characterized by (5). Note that $\alpha_x^{j-1,k-1} = \alpha_x^{j-1,k} = \alpha_x^{j,1}$.

From Lemma 1 we can infer that as $j \to \infty$, the sketch $\alpha_x^{j,k-1}$ converges to the sketch output by algorithm DWB.

Further, the iteration preserves the topology of a hierarchically planar drawing.

Lemma 3 *Suppose that a drawing α of a proper boundary $s-t$ graph $H = (V, A, \lambda, k)$ is hierarchically planar, and the sketch $\alpha^{j,i}$ ($1 \le i \le k-2$) preserves the topology of α. Then the sketch $\alpha^{j,i+1}$ also preserves the topology of α.*

Proof. Consider $\alpha^{j,i}$ and $\alpha^{j,i+1}$; only the locations of the nodes of L_{i+1} in $\alpha^{j,i}$ have been changed in $\alpha^{j,i+1}$. Since $\alpha^{j,i}$ preserves the topology of α, we need only to verify that for each pair $\{u, v\}$ of distinct nodes in L_{i+1}, if $\alpha(u)$ is strictly on the left of $\alpha(v)$ then $\alpha^{j,i+1}(u)$ is on the left of $\alpha^{j,i+1}(v)$.

For each node w in V, let N_w denote the node set $\{z : \ either\ (z, w) \in A\ or\ (w, z) \in A\}$.

Let $\{u, v\}$ be an arbitrary pair of distinct nodes in L_{i+1}, and let $\alpha(u)$ be strictly on the left of $\alpha(v)$. Since α is planar, we have that for each node $a \in N_u$ and each node $b \in N_v$, $\alpha(a)$ is strictly on the left of $\alpha(b)$. Note that $\alpha^{j,i}$ preserves the topology of α. Thus for each node $a \in N_u$ and each node $b \in N_v$, $\alpha^{j,i}(a)$ is on the left of $\alpha^{j,i}(b)$; this together with (5) ensure that $\alpha^{j,i+1}(u)$ is on the left of $\alpha^{j,i+1}(v)$. □

Proof of Theorem 2: If H is hierarchically planar with respect to β, then it has a hierarchically planar drawing α which extends β. Since the convergence is guaranteed by Lemma 1 irrespective of the initial sketch, we can use this drawing α as an initial sketch and apply the iteration (5). By Lemma 3 the sketch $\alpha_{H,\beta}$ output by algorithm DWB preserves the topology of α. Hence to prove planarity we only need to prove that if the input to algorithm DWB is a hierarchically planar proper boundary $s-t$ graph and is collapse free, then $\alpha_{H,\beta}$ is $1-1$.

Note that β is $1-1$ and $\alpha_{H,\beta}$ extends β. Thus if $\alpha_{H,\beta}$ is not $1-1$, then there are two nodes $u, v \in L_i$ $(i \neq 1, k)$ such that $\alpha_{H,\beta}(u) = \alpha_{H,\beta}(v)$. Without loss of generality, assume that $\alpha(u)$ is strictly on the left of $\alpha(v)$. Suppose that for $1 \leq j \leq k$, P_j denotes the set of nodes $w \in L_j$ such that there is a directed path from a source to a sink which passes through either u or v and contains w.

Since H is collapse free, without loss of generality, we may assume that $|P_j| \geq 2$ for $1 \leq j < i$. Note that α is hierarchically planar, and $\alpha_{H,b}$ preserves the topology of α. Thus for each node $a \in N_u$ and each node $b \in N_v$, $\alpha_{H,\beta}(a)$ is on the left of $\alpha_{H,\beta}(b)$. From (1), noting that $\alpha_{H,\beta}(u) = \alpha_{H,\beta}(v)$, we can deduce that in $\alpha_{H,\beta}$ the vertices of P_{i-1} are at the same location; and then similarly it follows that the vertices of P_{i-2} are at the same location (noting that $|P_{i-1}| \geq 2$). One can continue this argument to show that $|P_1| \geq 2$, contradicting the fact that β is $1-1$. Thus $\alpha_{H,\beta}$ is $1-1$.

Finally, note that the convexity of equations (1) implies the convexity of the output of algorithm DWB. □

5. SYMMETRY

In this section we briefly show that algorithm DWB above "displays symmetries"; intuitively, this means that if the input of algorithm DWB has an appropriate automorphism γ, then the output has a corresponding symmetry. A general model for symmetry in graph drawings is introduced by Lin[11] to make such intuitive notions precise; the application below of this model to hierarchical graphs is particularly simple.

A *geometric automorphism* of a hierarchical graph $H = (V, A, \lambda, k)$ is an automorphism γ of the underlying undirected graph such that $\gamma^2 = 1$ and either

- γ preserves the layers, that is, $\gamma(L_i) = L_i$, and γ has at most one fixed point in each layer (this is a *normal reflectional automorphism*); or
- γ reverses the layers, that is, $\gamma(L_i) = L_{k-i+1}$ for $1 \leq i \leq k$, and if k is odd, then γ fixes each node in $L_{(k+1)/2}$ (this is a *reverse reflectional automorphism*); or
- $\gamma(L_i) = L_{k-i+1}$ for $1 \leq i \leq k$, and there is at most one fixed point in $L_{(k+1)/2}$

when k is odd (this is a *rotational automorphism*).

A nontrivial isometry of the plane is a *symmetry* of a sketch of a hierarchical graph if it maps the image of a node to the image of a node, and maps the image of an edge to the image of an edge.

A sketch α *displays* a geometric automorphism γ if there is a symmetry s such that $s(\alpha(u)) = \alpha(\gamma(u))$ for each node u. If the sketch is $1-1$ (that is, α is a drawing), then we say that it *clearly displays* s.

It is not difficult to show that a symmetry of a drawing of a hierarchical graph induces a geometric automorphism of the hierarchical graph. More significantly, for the output of algorithm DWB, we can also show the converse.

Theorem 3 *Suppose that γ is a geometric automorphism of a proper boundary $s-t$ digraph H. If the boundary drawing β displays the restriction γ_B of γ to the boundary layers, then the output sketch of algorithm DWB displays γ.*

Proof. To prove this theorem, we need to introduce the following iteration which implements Gauss - Seidel iteration symmetrically:

Algorithm **DWB_SYMMETRIC_GSITERATIONS**

1. Choose an initial x coordinate $x(u)$ for each node u;
2. Repeat the following until x converges:
   2.1 *for* $i := 1$ *to* $\lfloor \frac{k-2}{2} \rfloor$ *do*
      for each $u \in L_{i+1}$ *do* $x(u) := \frac{1}{2d_u^-} \sum_{(v,u)} x(v) + \frac{1}{2d_u^+} \sum_{(u,w)} x(w)$;
      for each $u \in L_{k-i}$ *do* $x(u) := \frac{1}{2d_u^-} \sum_{(v,u)} x(v) + \frac{1}{2d_u^+} \sum_{(u,w)} x(w)$;
   2.2 IF k is odd THEN
      for each $u \in L_{(i+1)/2}$ *do* $x(u) := \frac{1}{2d_u^-} \sum_{(v,u)} x(v) + \frac{1}{2d_u^+} \sum_{(u,w)} x(w)$;

The following equations characterize the iteration above:

$$x^{(j)}(u) = \frac{1}{2d_u^-} \sum_{(v,u)} x^{(j)}(v) + \frac{1}{2d_u^+} \sum_{(u,w)} x^{(j-1)}(w), \tag{6}$$

when the iteration is implemented on $u \in L_{i+1}$; and

$$x^{(j)}(u) = \frac{1}{2d_u^-} \sum_{(v,u)} x^{(j-1)}(v) + \frac{1}{2d_u^+} \sum_{(u,w)} x^{(j)}(w), \tag{7}$$

when the iteration is implemented on $u \in L_{k-i}$; and

$$x^{(j)}(u) = \frac{1}{2d_u^-} \sum_{(v,u)} x^{(j)}(v) + \frac{1}{2d_u^+} \sum_{(u,w)} x^{(j)}(w), \tag{8}$$

when the iteration is implemented on $u \in L_{(k+1)/2}$. Here $x^{(0)}$ is the initial position and the value of x after j iterations of the main (outside) loop is $x^{(j)}$; if u is in the boundary, then $x^{(j)}(u) = x^{(j-1)}(u) = \beta_x(u)$.

Using the same method as for Lemma 1, we can prove that $x^j(u)$ produced by the iteration DWB_SYMMETRIC_GSITERATIONS converges to the solution of (1).

Given a geometric automorphism γ of H, one can easily construct a drawing $\alpha^{(0)}$ of H which extends β such that $\alpha^{(0)}$ displays γ. Note that every step of the iteration DWB_SYMMETRIC_GSITERATIONS characterized by (6) and (7) and (8) preserves the symmetry of the sketch. Thus, using $\alpha^{(0)}$ as an initial state for the iteration, the output sketch displays γ. □

6. MINIMIZING CROSSINGS

The main attraction of algorithm DWB is for symmetric and planar drawings. However, in this section we will briefly discuss the non-planar case.

Of course we would like to draw non-planar hierarchies with as few crossings as possible. Unfortunately, the problem of finding a drawing to minimize arc crossings in a hierarchical graph is NP-complete, even if there are only two layers[10].

Here we note that the crossing minimization problem is NP-complete even when the boundary drawing is fixed.

> **Fixed Boundary Crossings Problem**
> INSTANCE: A boundary $s-t$ graph H, a boundary drawing β of H, and an integer K.
> QUESTION: Is there a hierarchical drawing of H which extends β and has at most K arc crossings?

Eades and Wormald[7] have shown that for a hierarchical graph with two layers, if the positions of nodes in one layer are fixed, then the crossing minimizing problem is NP-complete. A simple polynomial transformation from the crossing minimization problem discussed by Eades and Wormald[7] to the Fixed Boundary Crossing problem above can be constructed; we leave details to the reader.

Theorem 4 *The Fixed Boundary Crossings Problem is NP-complete.* □

A number of crossing reduction heuristics are derived from the *barycentre method* of Sugiyama *et al.*[15] and its children[9,14] have been applied to this NP-complete problem. One variant of this method, outlined below, is very similar to the algorithm DWB_GSITERATION introduced in Section 3.

Algorithm **BARYCENTRE**

1. Choose initial x coordinate $x(u)$ for each node u (including boundary nodes);
2. Repeat the following until x does not change:
   2.1 *for* $i := 1$ *to* k *do*
   for each $u \in L_i$ *do* $t(u) := \frac{1}{2d_u^-}\sum_{(v,u)} x(v) + \frac{1}{2d_u^+}\sum_{(u,w)} x(w)$
   2.2 Sort the nodes of L_i into increasing order of t;
   2.3 Let the x coordinate of the jth node in L_i be j.

Although there is considerable discussion concerning the best choice of a variant for the barycentre method (see, for example, Rowe *et al.*[14]), it is clear that most variants are reasonably effective in reducing crossings.

The main difference between such barycentre methods and algorithm DWB is step 2.3: algorithm DWB_GSITERATION does not quantize the locations after each iteration. The quantization at step 2.3 makes BARYCENTRE difficult to analyse; for instance, it is not clear that it always terminates.

We believe that algorithm DWB can be used as a heuristic to reduce arc crossings. We have tested algorithm DWB and found that for graphs of less than 100 nodes and 300 edges, a reduction of around 50% may be expected. Note that the arc crossings can be further reduced using a greedy switching technique as a postprocess, as described by Mäkinen[12].

7. REMARKS

The theorems above can be generalized to cover improper hierarchical graphs, using a method which is standard in the graph drawing literature: each arc of span $k > 1$ is replaced by a path of $k - 1$ dummy nodes, then algorithm DWB is applied. In the final drawing, the dummy nodes are removed. Note that this strategy ensures that the *long arcs are drawn as straight lines* by algorithm DWB (because of the convexity of the equations (1)). It immediately establishes symmetry and planarity as well.

Di Battista and Nardelli[5] have given a linear algorithm to test planarity for hierarchical graphs (with some restrictions). Based on the order of the boundary nodes given by the di Battista - Nardelli algorithm, we may get a convex planar drawing with appropriate symmetries.

Algorithm DWB does not work well for dense graphs; if each layer is close to a complete bipartite graph, then a technique such as edge concentration[13] is necessary. Further, Algorithm DWB is limited by the necessity to choose a boundary sketch β; without a good choice of β, the worst case performance is poor and we are unable to provide performance guarantees such as those discussed by Eades and Wormald[7]. However, we have observed in tests that even a random choice of β usually gives a good drawing.

References

1. B. Becker and G. Hotz, "On the optimal layout of planar graphs with fixed boundary", *SIAM J. Computing* **16** (1987) 946-972.
2. J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*, (Macmillan, 1976).
3. J. Czyzowicz, A. Pelc and I. Rival, "Drawing orders with few slopes", Technical Report **TR-87-12**, Department of Computer Science, University of Ottawa, 1987.
4. G. Di Battista, P. Eades, R. Tamassia and I. Tollis, "Algorithms for Drawing Graphs: An Annotated Bibliography", *Comp. Geom. Theory Appl.*. **4** (1994) 235 – 282.
5. G. Di Battista and E. Nardelli, "Hierarchies and planarity theory", *IEEE Tran. Syst., Man, Cybern.* **18** (1988) 1035-1046.
6. P. Eades and X. Lin, "How to Draw a Directed Graph," Proc. IEEE Workshop on Visual Languages (*VL'89*), pp. 13-17, 1989.
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. P. Eades and N. C. Wormald, "Fixed edge length graph drawing is NP-hard", *Discrete Applied Mathematics* **28** (1990) 111-134.

9. E. R. Gansner, S. C. North and K. P. Vo, "DAG - a program that draws directed graphs", *Software Practice and Experience*, **18** (1988) 1047-1062.

10. M. R. Garey and D. S. Johnson, "Crossing number is NP-complete", *SIAM J. of Algebraic and Discrete Methods*, **4** (1983) 312-316.

11. X. Lin, "Analysis of algorithms for drawing graphs", Ph. D Thesis, Department of Computer Science, University of Queensland, 1992.

12. E. Mäkinen, "Experiments in drawing 2-Level hierarchical graphs", Technical Report **A-1988-1**, Department of Computer Science, University of Tampere, 1988.

13. F. J. Newbery, "Edge concentration: a method for clustering directed graphs", in *Proc. 2nd Int. Workshop on Software Configuration Management*, 1989, pp. 76-85.

14. L. A. Rowe, M. Davis, E. Messinger, C. Meyer, C. Spirakis and A. Tuan, "A browser for directed graphs", *Software Practice and Experience* **17** (1987) 61-76.

15. K. Sugiyama, S. Tagawa and M. Toda, "Methods for visual understanding of hierarchical system structures", *IEEE Tran. Syst., Man, Cybern.* **SMC-11** (1981) 109-125.

16. R. Tamassia, "On embedding a graph in the grid with the minimum number of bends", *SIAM J. Computing* **16** (1987) 421-444.

17. W. T. Tutte, "Convex representations of graphs", *Proc. London Math. Soc.* **10** (1960) 304-320.

18. W. T. Tutte, "How to draw a graph", *Proc. London Math. Soc.* **13** (1963) 743-768.