Algorithmica (1997) 17: 199–208

Algorithmica

Three-Dimensional Graph Drawing[1]

R. F. Cohen,[2] P. Eades,[2] Tao Lin,[3] and F. Ruskey[4]

Abstract. Graph drawing research has been mostly oriented toward two-dimensional drawings. This paper describes an investigation of fundamental aspects of three-dimensional graph drawing. In particular we give three results concerning the space required for three-dimensional drawings.

We show how to produce a grid drawing of an arbitrary n-vertex graph with all vertices located at integer grid points, in an $n \times 2n \times 2n$ grid, such that no pair of edges cross. This grid size is optimal to within a constant. We also show how to convert an orthogonal two-dimensional drawing in an $H \times V$ integer grid to a three-dimensional drawing with $\lceil\sqrt{H}\rceil \times \lceil\sqrt{H}\rceil \times V$ volume. Using this technique we show, for example, that three-dimensional drawings of binary trees can be computed with volume $\lceil\sqrt{n}\rceil \times \lceil\sqrt{n}\rceil \times \lceil\log n\rceil$. We give an algorithm for producing drawings of rooted trees in which the z-coordinate of a node represents the depth of the node in the tree; our algorithm minimizes the *footprint* of the drawing, that is, the size of the projection in the xy plane.

Finally, we list significant unsolved problems in algorithms for three-dimensional graph drawing.

Key Words. Graph drawing, Algorithms, Three-dimensional.

1. Introduction. Recent hardware advances have promised to bring the cost of three-dimensional interfaces low enough to make them widely available, and initial research into three-dimensional applications has begun [11], [17], [13], [20], [7]. However, as yet this work has been done mainly by software engineers who use experimentation rather than mathematical proof to establish the effectiveness of their techniques. For example, very little fundamental work has been done to understand the bounds on the volume used when drawing graphs in three dimensions. This paper presents some elementary but fundamental mathematical results for three-dimensional graph drawing.

The *size* of a grid drawing can be measured in various ways; for two-dimensional drawings the most common measure is the *area* of the drawing [4], [1], [2], [9]. However, for visualization systems, the most relevant measure is the maximum distance which the drawing extends in each dimension. To define this notion more precisely, suppose that S is a set of points in three-dimensional space. The *rectangular hull* of S is the smallest rectangular prism (with sides parallel to the coordinate axis) which contains all of S. If the set of vertex positions in a three-dimensional graph drawing $\mathcal{D}$ is S, and the rectangular hull of S has dimensions $X \times Y \times Z$, then the *size* of $\mathcal{D}$ is the maximum of X, Y, and Z.

[1] This work was performed as part of the Information Visualization Group (IVG) at the University of Newcastle. The IVG is supported in part by IBM Toronto Laboratory.
[2] Department of Computer Science, University of Newcastle, University Drive, Callaghan, New South Wales 2308, Australia. {rfc.eades}@cs.newcastle.edu.au.
[3] CSIRO DIT, GPO Box 664, Canberra, ACT 2601, Australia. taolin@csis.dit.csiro.au.
[4] Department of Computer Science, University of Victoria, Victoria, B.C. V8W 3P6, Canada. fruskey@csr.uvic.ca. This work was performed while he was visiting the University of Newcastle.

Received December 12, 1994; revised August 7, 1995. Communicated by T. Nishizeki.

A *Fary drawing* is a straight-line drawings with no edge crossings. This paper is concerned with bounds on the size of three-dimensional Fary grid drawings.

The next section proves upper and lower size bounds for three-dimensional Fary grid drawings of general graphs. Section 3 gives a size-efficient technique to convert a planar orthogonal grid drawing to a three-dimensional Fary grid drawing. The *footprint* algorithm is described in Section 4; this is an efficient algorithm for producing three-dimensional drawings of rooted trees with a small *footprint*, that is, a small projection in the xy plane. In the final section we list some fundamental unsolved problems for three-dimensional graph drawing.

2. Three-Dimensional Grid Fary Drawings. In this section we show how to produce a three-dimensional Fary grid drawing of any graph. Our drawing of a graph with n vertices fits into an $n \times 2n \times 2n$ grid. This significantly reduces the previous best known upper bound for this problem of $n \times n^2 \times n^3$ (G. Di Battista, private communication).

We achieve this volume bound by providing for each n a *universal set* of grid points $\mathcal{U}_n = \{p_1, \ldots, p_n\}$; this set has the property that any four distinct points p_i, p_j, p_k, and p_l are not coplanar. Our drawing algorithm simply places each vertex v_i at point p_i. Since an edge crossing requires four coplanar points, the output is a Fary drawing. The set $\mathcal{U}_n$ is *universal* in the sense that $\mathcal{U}_n$ depends only on the number of vertices n and not on the graph itself.

The choice of $\mathcal{U}_n$ is motivated by some elementary mathematics. The *moment curve* M is a three-dimensional curve defined by the parameters

$$M(t) = (t, t^2, t^3).$$

It is not difficult to prove that the moment curve has the property that distinct chords can only intersect endpoints. That is, given four distinct points p_1, p_2, p_3, and p_4 on M, the segments $\overline{p_1 p_2}$ and $\overline{p_3 p_4}$ do not intersect. Therefore, given a graph G with n vertices, we can obtain a three-dimensional Fary grid drawing of G by placing each vertex v_i at $M(i)$. This drawing uses volume $n \times n^2 \times n^3$; such a drawing is very sparse and has poor resolution on a screen.

However, we improve this upper bound using the following simple algorithm:

Algorithm 1. 3-D DRAW
Input: A graph G with n vertices
Output: A 3-D drawing of G

1. Choose a prime p with $n < p \leq 2n$
2. **for** $i = 1$ **to** n **do** place v_i at point $p_i = (i, i^2 \bmod p, i^3 \bmod p)$

THEOREM 1. *Algorithm* 1 *gives a three-dimensional Fary grid drawing of an n vertex graph in* $n \times 2n \times 2n$ *volume in* $O(n)$ *time.*

Proof. Note that Step 1 can be implemented in linear time using a simple prime number sieve [22]. Thus the algorithm takes linear time.

Given a vector $x = (x_1, \ldots, x_r)$, the *Vandermonde* matrix $V(x)$ of x is an $r \times r$ matrix whose ijth entry is x_j^i. The determinant of $V(x)$ has a special form [10]:

$$|V(x)| = \prod_{1 \le j \le r} x_j \prod_{1 \le i < j \le r} (x_j - x_i).$$

Note that this determinant is 0 if and only if either $x_j = 0$ for some j, or there is a pair i, j of distinct indices for which $x_i = x_j$.

Suppose that p is a prime greater than each of four integers i, j, k, l. Consider the following Vandermonde-like determinant:

$$\Delta = \begin{vmatrix} 1 & 1 & 1 & 1 \\ i & j & k & l \\ i^2 \bmod p & j^2 \bmod p & k^2 \bmod p & \ell^2 \bmod p \\ i^3 \bmod p & j^3 \bmod p & k^3 \bmod p & \ell^3 \bmod p \end{vmatrix}.$$

Note that the points p_i, p_j, p_k, p_l are coplanar if and only if $\Delta = 0$[8]. Since $(a + b) \bmod p = (a \bmod p + b \bmod p) \bmod p$ and $(ab) \bmod p = (a \bmod p)(b \bmod p) \bmod p$ we can deduce that

$$\Delta \bmod p = \begin{vmatrix} 1 & 1 & 1 & 1 \\ i & j & k & \ell \\ i^2 & j^2 & k^2 & \ell^2 \\ i^3 & j^3 & k^3 & \ell^3 \end{vmatrix} \bmod p,$$

and using simple arithmetic noting the similarity with the Vandermonde determinant we can deduce that

$$\Delta \bmod p = (\ell - k)(\ell - j)(\ell - i)(k - j)(k - i)(j - i) \bmod p,$$

and thus $\Delta \neq 0$, since i, j, k, ℓ are distinct and $i, j, k, \ell < p$. If follows that the points p_i, p_j, p_k, p_l in Algorithm 1 are not coplanar. □

We can show further that, apart from a constant, no general drawing algorithm can achieve smaller size.

THEOREM 2. *Suppose that $\mathcal{D}$ is a three-dimensional Fary grid drawing of the complete graph G on n vertices, and $\mathcal{D}$ uses volume $X \times Y \times Z$. Then each of X, Y, Z is $\Omega(n)$.*

Proof. Suppose that $X \le n/5$. Consider the planes parallel to the yz plane which intersect the drawing of G. At least one such plane must contain at least five vertices; since G is complete, the subgraph on these five vertices is complete, and thus is not planar [3]. Thus $X > n/5$; a similar argument applies to Y and Z. □

Note that although Algorithm 1 is useful in establishing the theoretical limits for general three-dimensional graph drawing, it is not a very practical algorithm because it is insensitive to the particular graph being drawn. Specific classes of graphs, in fact, can be drawn in size less than that achieved by Algorithm 1; the next section describes such a technique.

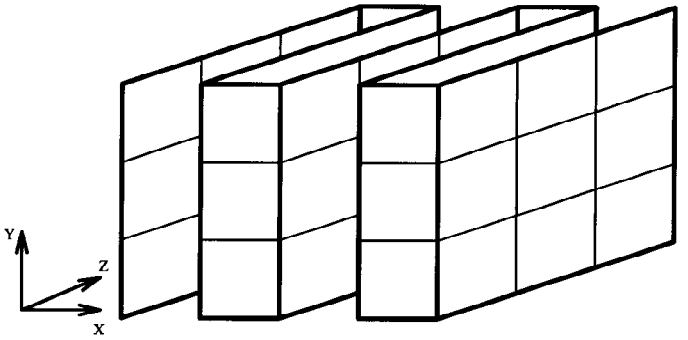

Fig. 1. Serpentine rollup.

3. Converting Orthogonal Grid Drawings to Three Dimensions. This section describes a general technique for converting a two-dimensional orthogonal grid drawing to a three-dimensional Fary drawing. Conceptually, the technique is quite simple: we just "roll up" the two-dimensional page. For specific classes of graphs, this technique gives a better size bound than the technique of the previous section.

For a positive integer r, the *serpentine rollup* σ_r maps the two-dimensional grid into the three-dimensional grid such that, for $x, y \geq 0$,

$$\sigma_r(x, y) = \begin{cases} (x \operatorname{div} r,\ y,\ x \bmod r) & \text{if } x \operatorname{div} r \text{ is even,} \\ (x \operatorname{div} r,\ y,\ x(r - x - 1) \bmod r & \text{if } x \operatorname{div} r \text{ is odd,} \end{cases}$$

where $x \operatorname{div} r = \lfloor x/r \rfloor$.

Note that the y-coordinate remains unchanged while the x-coordinate is "rolled-up" into the xz plane (see Figure 1).

The following lemmas follow directly from the definition of σ_r:

LEMMA 1. *If $\sigma_r(x_1, y_1) = \sigma_r(x_2, y_2)$, then $x_1 = x_2$ and $y_1 = y_2$.*

LEMMA 2. *Suppose that $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ are grid points with $x_1 < x_2$. Then either*

(a) $x(\sigma_r(p_1)) < x(\sigma_r(p_2))$, *or*
(b) $x(\sigma_r(p_1)) = x(\sigma_r(p_2))$ *and* $z(\sigma_r(p_1)) \neq z(\sigma_r(p_2))$.

LEMMA 3. *Suppose that $p_1 = (x_1, y_1)$, $p_2 = (x_2, y_2)$, and $p_3 = (x_3, y_3)$ are grid points with $x_1 < x_2 < x_3$ and $x(\sigma_r(p_1)) = x(\sigma_r(p_2)) = x(\sigma_r(p_3))$. Then either*

(a) $z(\sigma_r(p_1)) < z(\sigma_r(p_2)) < z(\sigma_r(p_3))$, *or*
(b) $z(\sigma_r(p_1)) > z(\sigma_r(p_2)) > z(\sigma_r(p_3))$.

Note that Lemma 1 implies that the inverse mapping $\sigma_r^{-1}(x, y, z)$ is defined for $x \geq 0$, $y \geq 0$, and $0 \leq z < r$.

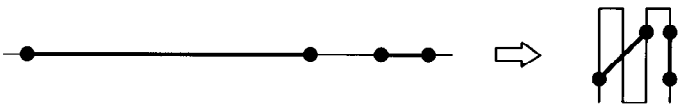

Fig. 2. The mapping to $\mathcal{D}3$ of horizontal edges.

Suppose $\mathcal{D}$ is a two-dimensional orthogonal drawing of a planar graph G. We assume that $\mathcal{D}$ has no edge bends since we can replace a bend with a dummy vertex. We produce a three-dimensional grid drawing $\mathcal{D}3$ of G by using the serpentine rollup, as follows. If vertex v is drawn at point p in $\mathcal{D}$, then we place v at $\sigma_r(v)$ in $\mathcal{D}3$. We then draw edges as straight line segments. Figure 2 illustrates the mapping from two horizontal edges in two dimensions to two edges in three dimensions.

LEMMA 4. *There are no edge crossings in* $\mathcal{D}3$.

Proof. Consider distinct edges $e = (v_1, v_2)$ and $f = (v_3, v_4)$ in G. Let (x_i, y_i) be the coordinates of the drawing of vertex v_i in $\mathcal{D}$. We show that the drawings of e and f do not cross in $\mathcal{D}3$.

We proceed by case analysis. Suppose in the drawing $\mathcal{D}$:

- *Edges e and f are drawn vertically.* In this case $x_1 = x_2$ and $x_3 = x_4$. If $x_1 \neq x_3$, then, by Lemma 2, $x(\sigma_r(x_1, y_1)) = x(\sigma_r(x_2, y_2)) \neq x(\sigma_r(x_3, y_3)) = x(\sigma_r(x_4, y_4))$ and so the edges do not cross. If $x_1 = x_3$, then we can assume without loss of generality that $y_1 < y_2 \leq y_3 < y_4$. Then the lemma holds since the mapping σ_r does not alter the y-coordinate of a point.
- *Edge e is drawn vertically and edge f is drawn horizontally.* Here $x_1 = x_2$ and $y_3 = y_4$. Suppose edges e and f cross in drawing $\mathcal{D}3$ but not in drawing $\mathcal{D}$. In $\mathcal{D}3$, edge e is contained in the line defined by $x = x(\sigma_r(x_1, y_1))$, $z = z(\sigma_r(x_1, y_1))$ and edge f is contained in the plane defined by $y = y(\sigma_r(x_3, y_3)) = y_3$. Therefore, the crossing must be at $p = (x(\sigma_r(x_1, y_1)), y_3, z(\sigma_r(x_1, y_1)))$. Then by Lemma 1, point $\sigma_r^{-1}(p)$ must be in both the drawings of e and f in $\mathcal{D}$, and is hence a crossing.
- *Edges e and f are drawn horizontally.* In this case $y_1 = y_2$ and $y_3 = y_4$. A crossing is possible in $\mathcal{D}3$ only if edges e and f are drawn with the same y-coordinate $y_1 = y_2 = y_3 = y_4$. Assume without loss of generality that $x_1 < x_2 \leq x_3 < x_4$. An edge crossing is not possible by the monotonicity of σ_r along the rollup described by Lemmas 2 and 3. □

From this lemma we can deduce the major property of the serpentine rollup transformation below.

THEOREM 3. *Suppose G is a planar graph and* $\mathcal{D}$ *is an orthogonal drawing of G with b bends using* $h \times w$ *area. Then for any integer* r, $1 \leq r \leq w$, *there is a three-dimensional grid drawing of G with b bends using* $r \times h \times \lceil w/r \rceil$ *volume.*

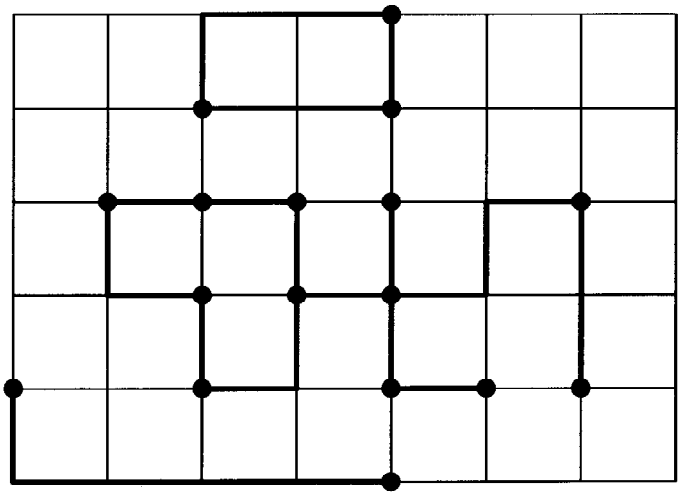

Fig. 3. An orthogonal grid drawing.

Thus two-dimensional orthogonal drawings can be efficiently "rolled up" to produce efficient three-dimensional Fary drawings. For instance, in [4] an algorithm is given for constructing an orthogonal grid drawing of a binary tree in an area of $n \times \log(n)$; we can deduce the following:

COROLLARY 1. *A three-dimensional Fary grid drawing of a binary tree with n nodes can be constructed in $O(n)$ time using $\sqrt{n} \times \sqrt{n} \times \log(n)$ volume.*

As another example, an orthogonal drawing of a planar graph with maximum degree 4 (as in Figure 3) can be constructed in an $n \times n$ grid, with $2n + 4$ bends and using linear time (see [15]). Thus we can deduce:

COROLLARY 2. *A three-dimensional grid drawing of a planar graph with maximum degree* 4 *and n nodes can be constructed in linear time using $n \times \sqrt{n} \times \sqrt{n}$ volume and having $O(n)$ total edge bends.*

4. The Minimum Footprint Algorithm. In this section we discuss three-dimensional grid drawings of rooted trees. We use the convention that the z-coordinate of a vertex is chosen to indicate its depth in the tree: the root has z-coordinate h, where h is the height of the whole tree, and a vertex of depth d has z-coordinate $h - d$. In other words, the vertices are placed in *layers*, where a layer is a plane $z = h - d$ containing the vertices of depth d. the edges are drawn as orthogonal polylines with two bends. Siblings are drawn on horizontal lines parallel to either the x- or the y-axis. These three-dimensional conventions are a direct generalization of common two-dimensional paradigms; two such drawings, one in two dimensions and one in three dimensions, are given in Figures 4 and 5.

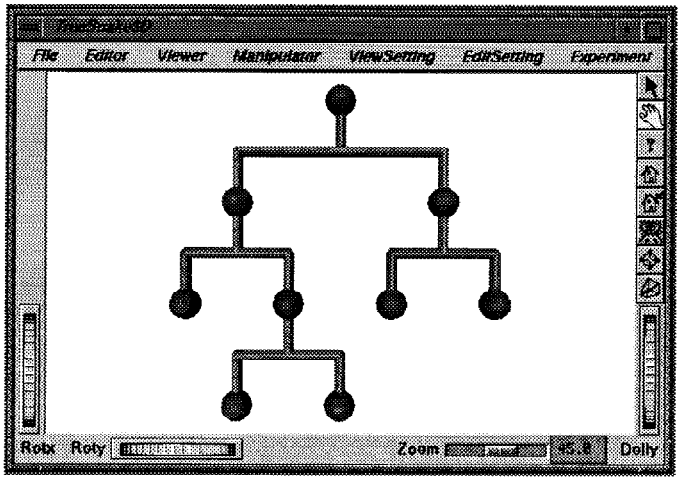

Fig. 4. A rooted tree in two dimensions.

We further require that subtrees are *separated* in the following sense. Suppose that u and v are two vertices in a tree drawing and neither is an ancestor of the other. Then we require that the rectangular hulls of the subtrees under u and v be disjoint. This requirement ensures that the subtrees are visually separated and, in applications such as browsing directory trees, assist navigation.

The above convention for three-dimensional tree drawing is called the *classical* convention, following the two-dimensional terminology of [6].

Note that the classical convention fixes the z-coordinate of each vertex in the drawing.

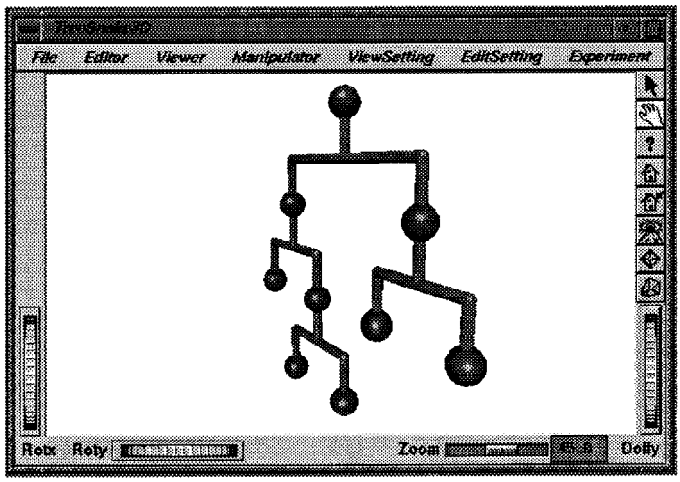

Fig. 5. A rooted tree in three dimensions.

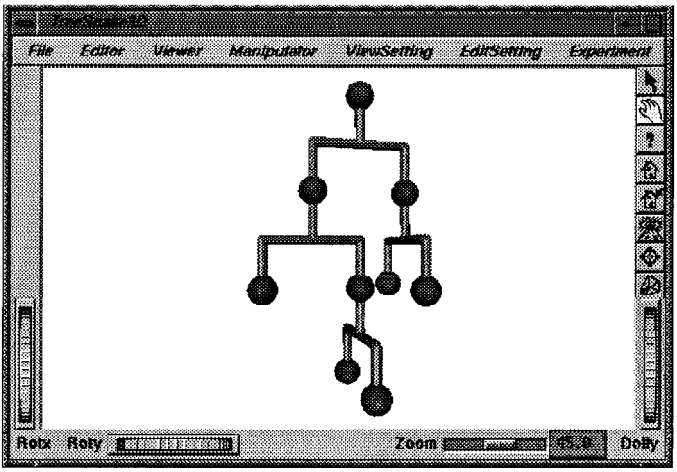

Fig. 6. A rooted tree in three dimensions.

The *footprint* of the drawing is the projection of the drawing onto the xy plane. If the rectangular hull of the footprint has dimensions $X \times Y$, then the *size* of the footprint is the maximum of X and Y.

The three-dimensional problem of drawing with a minimum-size footprint in three dimensions is the natural analogue to the well-studied problem of minimization of the width of a drawing in two dimensions (see [14], [21], [18], [19], [9], and [12]).

Note that according to the classical convention, the footprints of two disjoint subtrees must be disjoint. Further, if u has children v and w, then the footprint of the subtree under u is the rectangular hull of the union of the footprints of the subtrees under v and w. Thus the xy plane projection of a three-dimensional layout in the classical convention forms an *inclusion convention* drawing of the tree according to the terminology of [6]. For example, the techniques of [6] apply, and using the dynamic programming method of [5] we can deduce:

THEOREM 4. *A minimum footprint layout of a binary tree with n nodes can be found in time* $O(n^2)$.

We believe that the combination of the algorithmic techniques of [5] and the three-dimensional classical convention give a practical approach to drawing trees in three dimensions. Samples of three-dimensional classical convention drawings are shown in Figures 5 and 6.

5. Conclusions and Open Problems. In this paper we have demonstrated some initial results of an investigation of the fundamental algorithmic problems involved in drawing graphs in three dimensions. We believe that the results of Sections 2 and 3 are useful in defining the theoretical performance limits of layout algorithms in three dimensions.

The algorithm is Section 4 is a practical technique for drawing rooted trees in three dimensions.

However, the number of unanswered fundamental problems is quite large. Here we list a sample.

1. Is it possible to improve the lower bounds in this paper? In particular:
   (a) In our three-dimensional Fary grid drawing algorithm we find a collection of points with no four points coplanar. However, when drawing a complete graph, edge crossings can be avoided if any four coplanar points are arranged such that one point is interior to the convex hull of the remaining three. Is it possible to obtain an algorithm which uses this weaker condition? Is it possible to improve on Theorem 1? Is it possible to obtain a Fary drawing of a complete graph on n vertices in volume $n \times n \times n$?
   (b) Is it possible to obtain a Fary drawing of a binary tree with n nodes in volume $X \times Y \times Z$ where each of X, Y, Z is $O\left(\sqrt[3]{n}\right)$?
   (c) Is it possible to draw a tree of depth d with n vertices in the classical convention with footprint area $O(nd)$?
2. Find algorithms and lower bounds for three-dimensional Fary grid drawings of planar graphs. Are there planar graphs which require volume $X \times Y \times Z$ where the maximum of X, Y, Z is $\Omega(n)$? Note every planar graph can be drawn as the vertices and edges of a three-dimensional convex polyhedron [16]. Is it possible to obtain such a drawing with volume $O(n) \times O(n) \times O(n)$?
3. There has been a great deal of investigation of the problem of drawing a bipartite graph so that each part lies on a horizontal line. The corresponding problem of drawing a bipartite graph so that each part lies on a horizontal plane has not been investigated. It is not difficult to establish that to avoid crossings, K_{mn} requires area $\Omega(mn)$. However, no further results have been obtained.

References

[1] G. Di Battista, R. Tamassia, and I. G. Tollis. Area requirement and symmetry display in drawing graphs. *Proc. ACM Symp. on Computational Geometry*, pages 51–60, 1989.

[2] G. Di Battista, R. Tamassia, and I. G. Tollis. Area requirement and symmetry display of planar upward drawings. *Discrete and Computational Geometry*, 7:381–401, 1992.

[3] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications*. North-Holland, New York, 1976.

[4] P. Crescenzi, G. Di Battista, and A. Piperno. A note on optimal area algorithms for upward drawings, of binary trees. Technical Report 11.91, Dipartimento di Informatica e Sistemistica, Univ. di Roma "La Sapienza," 1991.

[5] P. Eades, T. Lin, and X. Lin. Minimum size h-v drawings. In *Advanced Visual Interfaces* (*Proceedings of AVI* 92, *Rome*), World Scientific Series in Computer Science, volume 36, pages 386–394, 1992.

[6] P. Eades, T. Lin, and X. Lin. Two tree drawing conventions. *International Journal of Computational Geometry and Applications*, 3(2):133–153, 1993.

[7] P. Eades, C. Stirk, and S. Whitesides. The techniques of Kologorov and Barzdin for three-dimensional orthogonal graph drawing. *Information Processing Letters*, to appear.

[8] J. D. Foley and A. van Dam. *Fundamentals of Interactive Computer Graphics*. Addison-Wesley, Reading, MA, 1982.

[9] A. Garg, M. T. Goodrich, and R. Tamassia. Area-efficient upward tree drawings. *Proc. ACM Symp. on Computational Geometry*, pages 359–368, 1993.

[10] D. Knuth. *The Art of Computer Programming*. Volume 1: *Fundamental Algorithms*. Addison-Wesley, Reading, MA, 1975.
[11] J. Mackinley, G. Robertson, and S. Card. Cone trees: Animated 3d visualizations of hierarchical information. *Proceedings of SIGCHI Conference on Human Factors in Computing*, pages 189–194, 1991.
[12] S. Moen. Drawing dynamic trees. *IEEE Software*, 7:21–8, 1990.
[13] P. Reid. Dynamic interactive display of complex data structures. In *Graphics Tools for Software Engineers*, pages 60–62. Cambridge University Press, Cambridge, 1989.
[14] E. Reingold and J. Tilford. Tidier drawing of trees. *IEEE Transactions on Software Engineering* 7(2):223–228, 1981.
[15] R. Tamassia. Planar orthogonal drawings of graphs. *Proc. IEEE Internat. Symp. on Circuits and Systems*, pages 319–322, 1990.
[16] W. T. Tutte. Convex representations of graphs. *Proceedings London Mathematical Society*, 10:304–320, 1960.
[17] O. Tversky, S. Snibbe, and R. Zeleznik. Cone trees in the uga graphics system: Suggestions of a more robust visualization tool. Technical Report CS-93-07, Brown University, 1993.
[18] J. Vaucher. Pretty printing of trees. *Software—Practice and Experience*, 10(7):553–561, 1980.
[19] J. Q. Walker II. A node-positioning algorithm for general trees. *Software—Practice and Experience*, 20(7):685–705, 1990.
[20] C. Ware, D. Hui, and G. Franck. Visualizing object oriented software in three dimensions. *CASCON* 1993 *Proceedings*, 1993.
[21] C. Wetherell and A. Shannon. Tidy drawing of trees. *IEEE Transactions on Software Engineering*, 5(5): 514–520, 1979.
[22] N. Wirth. *Systematic Programming*: *An Introduction*. Prentice-Hall, Englewood Cliffs, NJ, 1973.