Incremental Orthogonal Graph Drawing in Three Dimensions

Achilleas Papakostas and Ioannis G. Tollis*

Dept. of Computer Science
The University of Texas at Dallas
Richardson, TX 75083-0688
email: papakost@utdallas.edu, tollis@utdallas.edu

Abstract. We present two algorithms for orthogonal graph drawing in three dimensional space. For graphs of maximum degree six, the 3-D drawing is produced in linear time, has volume at most $4.66n^3$ and each edge has at most three bends. If the degree of the graph is arbitrary, the vertices are represented by solid 3-D boxes whose surface is proportional to their degree. The produced drawing has two bends per edge. Both algorithms guarantee no crossings and can be used under an interactive setting (i.e., vertices arrive and enter the drawing on-line), as well.

1 Introduction

Graph drawing addresses the problem of automatically generating geometric representations of abstract graphs or networks. For a survey of graph drawing algorithms and other related results see the annotated bibliography of Di Battista, Eades, Tamassia and Tollis [6]. Various algorithms have been introduced to produce orthogonal drawings of planar [1, 10, 20] or general [1, 10, 13, 15, 19] graphs of maximum degree three or four. For drawings of general graphs, the required area can be as little as $0.76n^2$ [13, 15], the total number of bends is no more than $2n + 2$, and there are at most two bends any edge [1, 13, 15]. There has been a recent trend in Graph Drawing to visualize graphs in the three dimensional space [2, 3, 5, 7, 8, 9, 17, 18]. Although the number of applications that require such a representation for graphs is still limited [2, 9, 12, 18], there is no doubt that 3-D Graph Drawing will find many applications in the future.

In [4] it is shown that any n-vertex graph has a 3-D drawing in a $n \times 2n \times 2n$ grid, so that all vertices are located on grid points, and no two edges cross. In the same paper, a technique to convert an orthogonal 2-D drawing of area $H \times V$ to a 3-D straight-line drawing of volume $\lceil\sqrt{H}\rceil \times \lceil\sqrt{H}\rceil \times V$ is also presented. Graph drawing in three dimensional space Naturally, orthogonal drawing in three dimensional space has also received attention recently [7, 8, 9, 17]. A 3-D orthogonal drawing typically has the following properties:

– vertices are points with integer coordinates in three dimensional space,

* Research supported in part by NIST, Advanced Technology Program grant number 70NANB5H1162.

- each edge is a polyline sequence of consecutive straight line segments; each one of these line segments is parallel either to the x-axis, y-axis, or z-axis,
- the meeting point of two consecutive straight line segments of the same edge is a bend and has integer coordinates, and
- line segments coming from routes of two different edges are not allowed to overlap.

A very interesting upper bound on the volume for 3-D orthogonal drawings of graphs of maximum degree six is shown in [8]. More specifically, the volume for such drawings is at most $O(\sqrt{n}) \times O(\sqrt{n}) \times O(\sqrt{n})$, while each edge has at most seven bends, and no two edges cross. This improves the result in [7], where the volume upper bound was the same but the drawings allowed up to 16 bends per edge. If we require that each edge have at most three bends, then another algorithm is presented in [8] that requires volume exactly $27n^3$ (the produced drawings have no crossings). Both algorithms run in $O(n^{\frac{3}{2}})$ time. Note that Kolmogorov and Bardzin [11] show an existential lower bound of $\Omega(n^{\frac{3}{2}})$ on the volume occupied by 3-D orthogonal drawings.

In this paper we present one algorithm for producing 3-D orthogonal drawings of graphs of maximum degree six, and a second algorithm that produces 3-D orthogonal drawings of graphs of arbitrary degree. Note that there has not been any previous work that dealt with the theory of 3-D orthogonal drawing of graphs of arbitrary degree. Both algorithms are based on the "Relative-Coordinates" paradigm for vertex insertion [14, 16]. As such, both algorithms support interactive environments where vertices arrive and enter the drawing on-line. An important feature of this work is that both algorithms guarantee no edge crossings.

Given an n-vertex graph G of maximum degree six, our first algorithm produces a 3-D orthogonal drawing of G whose volume is at most $4.66n^3$, in linear time. Moreover, each edge of the drawing has at most three bends. Hence, our algorithm outperforms the algorithm of [8] in terms of both running time and volume of the drawing. Our second algorithm uses solid three dimensional boxes to represent vertices. The surface of each such box is proportional to the degree of the represented vertex. The produced 3-D orthogonal drawings have at most two bends per edge, and volume $O((\frac{m}{3} + O(n))^3)$, where m is the number of edges of the drawing.

2 Preliminaries

Clearly, for each graph of maximum degree six, there is a 3-D orthogonal drawing. The system of coordinates typically used in three dimensional space is based on three axes x, y, z so that each one of them is perpendicular to the other two (see Fig. 1a). Three different planes are formed by the three possible ways we can pair these axes: The xz-plane is defined by the x, z-axes, the yz-plane is defined by the y, z-axes, and the xy-plane is defined by the x, y-axes. Each one of these planes is called a *base* plane; each base plane is perpendicular to the other two.

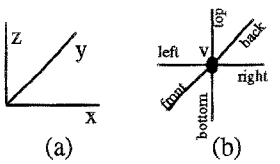

Fig. 1. (a) Coordinates system for 3-D drawing, (b) possible directions from where an edge can enter v.

Each vertex of a 3-D drawing has six possible directions around it from where incident edges may enter the vertex. The two directions parallel to the z-axis are *top* (extending towards the positive part of the z-axis) and *bottom* (extending towards the negative part of the z-axis). *Front* and *back* directions are parallel to the y-axis and they extend towards the negative and positive parts of the y-axis, respectively. The remaining two directions are parallel to the x-axis and are called *left* (extending towards the negative part of the x-axis) and *right*. See also Fig. 1b.

Two directions parallel to the same axis are *opposite* directions. Two directions parallel to two different axes are *orthogonal* directions. If there is no edge entering a vertex v from a specific direction of v, this direction is called *free direction* of v. A *plane* free direction is a left, right, front, or back free direction. Consider vertices $v_1, v_2, \cdots v_r$, where $r \geq 2$, having plane free directions $fd_1, fd_2, \cdots fd_r$ which extend towards the same direction (e.g., they are all left free directions). The set of the fd_i's forms a *beam*. If the fd_i's are left (resp. right, front, back) free directions, then their beam is a left (resp. right, front, back) beam. Vertices $v_1, v_2, \cdots v_r$ are the *origins* of the beam. Two beams are opposite (resp. orthogonal) if the free direction of one beam is opposite (resp. orthogonal) to the free direction of the other.

The volume of a 3-D drawing is the volume of the smallest rectangular parallelepiped that encloses the 3-D drawing. Also recall [14, 16] the following terminology: The *current* drawing is the drawing before the insertion of a new vertex v; the number of vertices of the current drawing that will be connected with v through new edges, is v's *local* degree. We call these vertices *adjacent* vertices of v.

3 Drawing Graphs with Maximum Degree Six

In this section we present our incremental algorithm for producing orthogonal drawings of graphs of maximum degree six in the three dimensional space. The incremental nature of our algorithm comes from the fact that a user is allowed to insert vertices (along with edges to other existing vertices) into the current drawing in any order. The algorithm supports such vertex insertions at any moment t, as long as each request observes the following rules:

- we start the drawing from scratch, that is the very first current drawing is the empty graph,
- the degree of any vertex of the current drawing at any time t is at most six, and
- the graph represented by the current drawing is always connected.

Our technique follows the Relative-Coordinates scenario [14, 16]. This means that the decision about where to place a new vertex and how its incident edges will be routed depends entirely on the free directions around the adjacent vertices. The properties of the Relative-Coordinates scenario [14, 16] are also properties of the 3-D drawings produced by our algorithm and guarantee a "smooth" transition from the current drawing to the next. The notation $u \rightarrow p \rightarrow p'$ means that from vertex u we draw a straight line segment that intersects plane p perpendicularly, and from the intersection point we draw another segment to plane p' that intersects p' perpendicularly as well. We use the notation $p_{a,v}$, where $a = x, y$, or z and v some vertex, to denote the plane which is perpendicular to the a-axis and contains vertex v. As we will see later, our 3-D orthogonal drawing is built in an upward fashion (i.e., it grows along the positive z-axis). For this reason, we always keep the following basic rule during the interactive drawing process:

Basic Rule: No vertex has a bottom free direction in any current drawing.

Most of the edges we route follow one of five fundamental routes, described below, depending on their available free directions. Assume that w and w' are two vertices of the current drawing and w has higher z-coordinate than w'. In the first three fundamental routes, edge (w', w) always enters w' from its left (or other plane) free direction. In the remaining two fundamental routes, edge (w', w) enters w' from its top free direction. Note that these fundamental routes can generalize to other situations (besides the examples shown in Fig. 2).

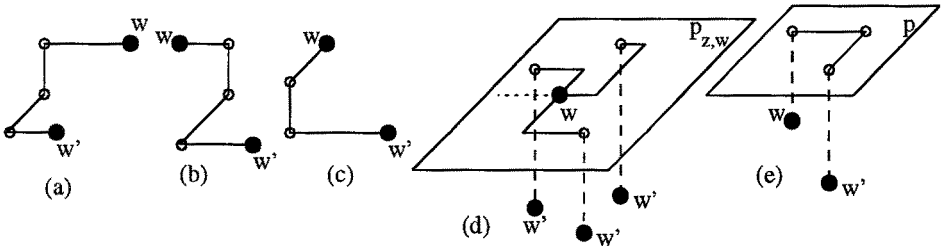

Fig. 2. (a) First, (b) Second, (c) Third Fundamental Routes, (d) Same-Plane, (e) Over-The-Top Routes.

- **First Fundamental Route:** Edge (w', w) enters w from its left free direction. We open up a new plane p to the left of the leftmost plane of

the current drawing. Edge (w', w) is routed with three bends as follows: $w' \rightarrow p \rightarrow p_{y,w} \rightarrow p_{z,w} \rightarrow w$. This is shown in Fig. 2a. The small empty circles of this figure denote the three bends of the route.

- **Second Fundamental Route:** Vertex w has lower x-coordinate than w', and edge (w', w) enters w from its right free direction. We open up a new plane p parallel to the yz-plane and one unit to the right of w. Edge (w', w) is routed with three bends (see Fig. 2b), as follows: $w' \rightarrow p \rightarrow p_{y,w} \rightarrow p_{z,w} \rightarrow w$.
- **Third Fundamental Route:** Vertex w has lower x-coordinate and higher y-coordinate than w', and edge (w', w) enters w from its front free direction. No new plane is opened up and we route edge (w', w) with two bends (see Fig. 2c) as follows: $w' \rightarrow p_{x,w} \rightarrow p_{z,w} \rightarrow w$.
- **Same-Plane Route:** Edge (w', w) may enter w from any one of its plane free directions. We draw a straight line segment from w' intersecting plane $p_{z,w}$ perpendicularly. The remaining portion of edge (w', w) is routed exclusively in $p_{z,w}$, and may enter w from any one of its plane free directions with at most two bends (if two bends are required, then a new plane parallel either to the xz or yz-plane has to be inserted). This means that the whole route has at most three bends. In Fig. 2d we show three examples of the portions of three routes in plane $p_{z,w}$.
- **Over-The-Top Route:** Edge (w', w) enters w from its top free direction. A new plane p parallel to the xy-plane is inserted in the drawing, one unit above w. Edge (w', w) is routed with three bends (see Fig. 2e) as follows: $w' \rightarrow p \rightarrow p_{x,w} \rightarrow p_{y,w} \rightarrow w$. In other words, we draw a straight line segment intersecting p perpendicularly, route the edge in p bringing it directly on top of w with one bend, and then just draw the line segment from that point to w.

3.1 Overview of the Algorithm - Preprocessing

Assume that we start with an empty graph. The following gives an overview of the algorithm for placing the next vertex v in the current drawing. The steps of this algorithm are analyzed in this and the following subsections. Let v_1 be the first vertex to be inserted. Vertex v_1 has local degree zero. If v_2 is the second vertex to be inserted, then v_2 has local degree one and is connected with v_1. In Fig. 3a, we show the first two vertices inserted in an empty drawing. There are three observations to make about Fig. 3a. First, edge (v_1, v_2) has three bends. Second, a total of seven new planes are inserted in the empty drawing. Third, neither v_1 nor v_2 has a bottom free direction.

1. IF v is the first or second vertex to be inserted, THEN place them as discussed above.
2. ELSE
   (a) Find v's adjacent vertices $u_1, \cdots u_l$ in the current drawing.
   (b) FOR each adjacent vertex determine its connector (use the procedure described below).

(c) Find which Routing Case (see next section) v's insertion falls into.
(d) WITHIN a Routing Case:
  i. Determine the anchor vertex u_a and the cover vertex u_c.
  ii. Place v.
  iii. Route edge (u_a, v).
  iv. Route the remaining edges (u_i, v) except (u_c, v), using the three Fundamental Routes and/or the Same-Plane Route.
  v. Route edge (u_c, v).

Let v be the next vertex to be inserted in the current drawing and l ($1 \leq l \leq 6$) be v's local degree. We find the l adjacent vertices $u_1, u_2, \cdots u_l$ of v. According to the Basic Rule, v must not have a bottom free direction after v is placed and all its l incident edges are routed. This means that exactly one of these edges must enter v from the bottom. The vertex which is the other endpoint of this edge is called *anchor* vertex, and is denoted by u_a. If $l = 6$, then the last one of v's incident edges to be routed enters v from its top free direction. The other endpoint of this edge is called *cover* vertex, and is denoted by u_c.

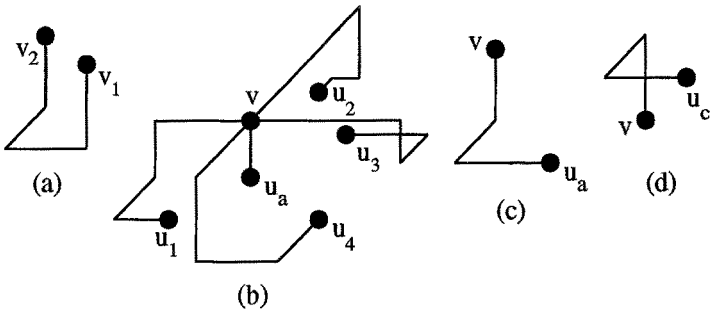

Fig. 3. (a) Inserting the first two vertices, (b) a Routing Case 1 example, (c) (u_a, v) of Routing Case 1 when u_a does not have top connector, (d) (u_c, v) of Routing Case 2 when u_c does not have top connector.

For each adjacent vertex u_i, we must pick one of its free directions which will be used for routing edge (u_i, v). The free direction picked for each u_i is called u_i's *connector*. Once a connector for an adjacent vertex u_i is determined, it remains the same throughout the whole process of placing v and routing its incident edges. If a connector of some u_i is a right (left, front, back, top) free direction, then it is called *right (left, front, back, top) connector*. Opposite, orthogonal, and plane connectors are defined in the same way as for free directions. Also, a beam of connectors is defined similarly to the beam of free directions. Let c_i be u_i's connector. We run the following procedure to determine the connector of each u_i.

1. Choose a free direction fd_i for each u_i so that:
   (a) The number of pairs $< fd_i, fd_j >$ $(i \neq j$ and $1 \leq i, j \leq l)$ where fd_i and fd_j are opposite, is the smallest possible.
   (b) fd_i is top free direction, only if u_i has only this free direction left.
2. IF there are no two opposite beams among the fd_i's, THEN
   (a) FOR each u_i:
      i. $c_i := fd_i$.
3. ELSE IF there are two opposite beams B_1 and B_2, THEN
   (a) Consider the beam with the smallest cardinality; say B_1.
   (b) FOR each origin u_i of B_1:
      i. IF u_i's top free direction is available, THEN $c_i :=$ top connector.
      ii. ELSE $c_i := fd_i$.
   (c) FOR each u_i that is NOT an origin of B_1:
      i. $c_i := fd_i$.

3.2 Vertex Placement and Edge Routing

As we will see in this subsection, many of v's incident edges are routed using the fundamental routes. When this is the case, vertex v corresponds to w, and the adjacent vertex u_i, which is the other end of the route, corresponds to w'. Depending on the types of connectors that v's adjacent vertices have, we distinguish three Routing Cases, which are briefly discussed below (see [17] for a detailed description):

Routing Case 1: There is no beam among connectors c_i. Anchor u_a is selected among the adjacent vertices with top connectors. Edge (u_a, v) is a simple straight line segment from u_a to v (see Fig. 3b). If there is no adjacent vertex having top connector, any adjacent vertex can be the anchor vertex u_a. Then, edge (u_a, v) is routed with two bends as shown in Fig. 3c. This generalizes to cases where u_a has a different plane connector, through a rotation. The remaining edges (u_i, v) where u_i is not the anchor, are routed using the fundamental routes.

Routing Case 2: There is at least one beam among connectors c_i and there are no two opposite beams. If there is at least one adjacent vertex with top connector and $l = 6$, then this vertex is the cover u_c. If $l = 6$ and there is no adjacent vertex with top connector, then cover u_c is the adjacent vertex with highest z-coordinate which belongs to a beam. Then, we find the beam B_{max} having the highest cardinality without counting u_c. Assume that B_{max} is a left beam (the following discussion generalizes through rotation). Anchor u_a is always one of B_{max}'s origins. More specifically, it is the vertex whose y-coordinate is the median of the y-coordinates of all B_{max}'s origins. Edge (u_a, v) is routed with two bends (see Fig. 3c).

Vertex v is connected with the rest of its adjacent vertices using the fundamental routes. If cover u_c does not have top connector, then, by the way it was chosen, it has the following properties: (a) u_c has higher z-coordinate than v (see [17]), and (b) u_c has either left (if it is an origin of B_{max}), or back (if it is an origin of the other beam different from B_{max}) connector. If it has left connector,

it is routed with two bends as shown in Fig. 3d (routing is similar when u_c has back connector).

Routing Case 3: There are two opposite beams among the u_i's. Clearly, in this routing case, we have that $l \geq 4$. B_{max} is the beam with the highest cardinality. Let us assume that B_{max} is a left beam (the discussion generalizes through rotation). Let B be the beam which is opposite to B_{max}. Anchor u_a is the median of the origins of beam B_{max} with respect to their y-coordinates. Edges connecting v with the origins of the two beams are basically routed using the fundamental routes. There are two exceptions to that:

The first exception deals with the situation where exactly one edge (u_B, v) (where u_B is an origin of B) is routed on top of the current drawing, all the way from the rightmost to the leftmost side of the drawing (see Fig. 4a). The second exception deals with the situation where $l = 6$ and there is no adjacent vertex with top connector. Vertex u_c is an origin of B, and edge (u_c, v) is routed with three bends as shown in Fig. 4b. Note that in this situation, vertex v is placed in plane p_{y,u_c}.

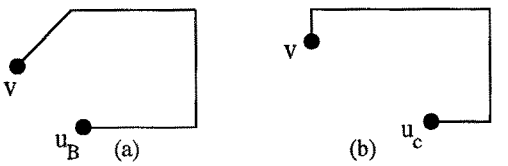

Fig. 4. Routing Case 3: (a) edge routing on top of current drawing, (b) edge (u_c, v) when u_c has no top connector.

In Fig. 5 we show the 3-D orthogonal drawing of K_7 produced by our algorithm. The numbers in the vertices denote the order in which the vertices were inserted. The volume of this drawing is $8 \times 8 \times 8 = 512 \leq 1.5n^3$, where $n = 7$. Observe that out of K_7's 21 edges, there is one edge with no bends, 12 edges require two bends each, and the remaining eight edges are routed with three bends each. Due to space limitations, we only present our two main theorems for orthogonal graph drawing in three dimensions of graphs of maximum degree six. For more details, see the full paper [17].

Theorem 1. *There is a 3-D orthogonal graph drawing algorithm for graphs of maximum degree six that allows on-line vertex insertion so that the following hold at any time t:*

- *after each vertex insertion, the coordinates of any vertex or bend of the current drawing shift by a small constant amount of units along the x, y, z-axes, effectively maintaining the general shape of the drawing,*
- *there are at most three bends along any edge,*
- *no two edges cross,*

- *the volume of the drawing is at most* $4.66n^3(t)$*, where* $n(t)$ *is the number of vertices in the drawing at time* t*, and*
- *vertex insertion takes constant time (if the screen needs to be refreshed after each vertex insertion, then it takes linear time).*

Our incremental algorithm can be used to produce a 3-D orthogonal drawing of a graph by first numbering its vertices and then inserting each vertex one at a time. By maintaining the drawing implicitly each insertion takes constant time. Hence we have the following:

Theorem 2. *Let* G *be an* n*-vertex connected graph of maximum degree six. There is a linear-time algorithm that produces a 3-D orthogonal drawing of* G*, so that each edge has at most three bends, no two edges cross, and the volume of the drawing is at most* $4.66n^3$*.*

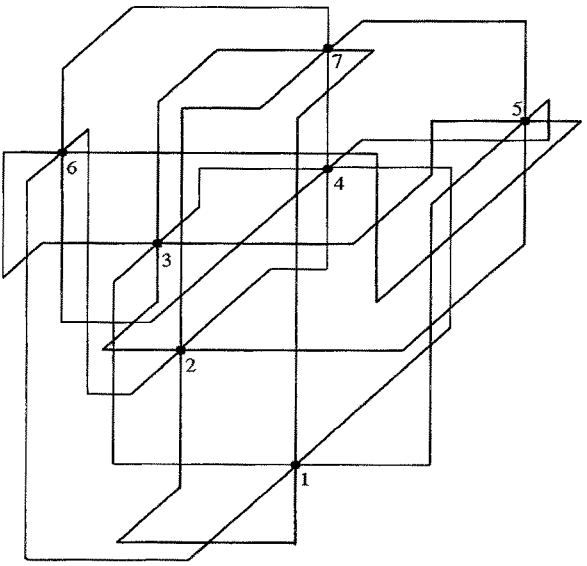

Fig. 5. 3-D orthogonal drawing of K_7 produced by our algorithm.

4 Drawing High Degree Graphs

In this section we give a very brief description of our model to support high degree three dimensional orthogonal graph drawing based on the Relative-Coordinates scenario. Our model allows vertices to arrive on-line and the degree of the vertices to increase arbitrarily. We represent vertices using three dimensional boxes.

When a vertex is inserted into the drawing, it is represented by a cubic box of size depending on the degree of the vertex. Edges that are adjacent to a vertex are attached to the surface of its box. The points on the box surface where edges can be attached are called *connectors*, and they have integer coordinates. As a result of edge routing, edges may require to attach to specific sides of incident boxes. If there are no available connectors on that side, we need to grow the box creating new connectors on that side. Our model for representing vertices in three dimensional space supports box growing. Figure 6a illustrates how box growing works. Finally, note that although the box of every vertex starts out having a cubic configuration, it may grow its size in various different ways in the course of the drawing process.

Our algorithm produces a 3-D drawing considering and placing one vertex at a time. Assume that v is the next vertex to be inserted in the current 3-D drawing. Let k be v's local degree, and let $u_1, u_2, \cdots u_k$ be v's adjacent vertices. For each vertex u_i we find the sides of box u_i that have available connectors. Then we find the side of the adjacent boxes on which most of these boxes have at least one available connector. This is the side where the edges connecting v with each u_i will be attached.

The next step is to create the box representing v and place it in the current 3-D drawing. Box v is a cube. Each newly inserted box is placed in such a way so that none of its connectors has the same x, y or z-coordinate as any other connector of any box of the current drawing. In [17] we describe in detail how v's position is computed. After v is inserted, the edges that connect v with its adjacent vertices are routed. We show that each edge can be routed with two bends without crossing other edges [17].

Theorem 3. *There is an algorithm to produce 3-D orthogonal drawings of graphs (not necessarily connected) which allows vertices to arrive on-line. The drawings have the following properties at any time t:*

- *vertices are represented by boxes and the surface of each box is at most six times the current degree of the vertex,*
- *each edge has two bends,*
- *no two edges cross,*
- *the volume is $O((\frac{m(t)}{3}+n(t))^3)$, where $m(t)$ and $n(t)$ are the number of edges and vertices at time t, and*
- *vertex insertion takes constant time.*

In practice, we expect the volume to be smaller than the upper bound given in the above theorem. This is because our analysis assumes that for each vertex insertion the boxes of all the adjacent vertices need to grow. We expect a box to grow very infrequently, since each box has several connectors on its sides. If the each inserted vertex is adjacent to no more than 16 vertices then the volume is bounded by $(\frac{m(t)}{3} + 2n(t))^3$. Figure 6b shows the 3-D orthogonal drawing of K_5 as produced by our algorithm. The box numbers denote the vertex insertion order.

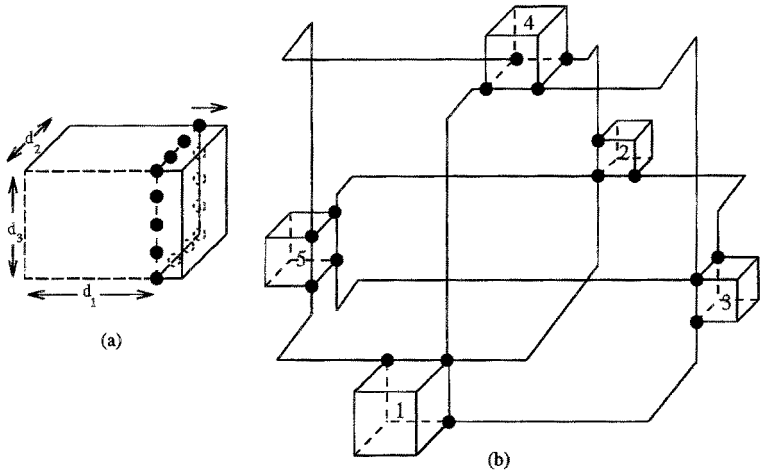

Fig. 6. (a) Box growing creates new connectors (denoted by solid and dotted circles), (b) 3-D orthogonal drawing of K_5, using boxes to represent vertices.

5 Conclusions and Open Problems

We presented incremental algorithms for producing orthogonal graph drawings in three dimensional space. The first algorithm deals with graphs of maximum degree six, and the produced drawings have volume at most $4.66n^3$. This improves the best known [8] volume requirement of exactly $27n^3$, while maintaining the same upper bound for the number of bends per edge. Our algorithm runs in linear time.

The second algorithm introduces 3-D orthogonal drawing for graphs of degree higher than six. Vertices are represented by solid three dimensional boxes whose surface is proportional to the degree of the vertex. The volume of the drawings is bounded by $O((\frac{m}{3}+n)^3)$ and each edge has only two bends.

Improving the upper bounds on the volume while keeping the number of bends per edge to three or less, is an interesting open problem.

References

1. T. Biedl and G. Kant, *A Better Heuristic for Orthogonal Graph Drawings*, Proc. 2nd Ann. European Symposium on Algorithms (ESA '94), Lecture Notes in Computer Science, vol. 855, pp. 24-35, Springer-Verlag, 1994.
2. M. Brown and M. Najork, *Algorithm animation using 3D interactive graphics*, Proc. ACM Symp. on User Interface Software and Technology, 1993, pp. 93-100.
3. I. Bruss and A. Frick, *Fast Interactive 3-D Visualization*, Proc. of Workshop GD '95, Lecture Notes in Comp. Sci. 1027, Springer-Verlag, 1995, pp. 99-110.

4. R. Cohen, P. Eades, T. Lin, F. Ruskey, *Three Dimensional Graph Drawing*, Proc. of DIMACS Workshop GD '94, Lecture Notes in Comp. Sci. 894, Springer-Verlag, 1994, pp. 1-11.
5. I. Cruz and J. Twarog, *3D Graph Drawing with Simulated Annealing*, Proc. of Workshop GD '95, Lecture Notes in Comp. Sci. 1027, Springer-Verlag, 1995, pp. 162-165.
6. G. Di Battista, P. Eades, R. Tamassia and I. Tollis, *Algorithms for Drawing Graphs: An Annotated Bibliography*, Computational Geometry: Theory and Applications, vol. 4, no 5, 1994, pp. 235-282. Also available via anonymous `ftp` from `ftp.cs.brown.edu`, `gdbiblio.tex.Z` and `gdbiblio.ps.Z` in `/pub/papers/compgeo`.
7. P. Eades, C. Stirk, S. Whitesides, *The Techniques of Kolmogorov and Bardzin for Three Dimensional Orthogonal Graph Drawings*, TR 95-07, Dept. of Computer Science, University of Newcastle, Australia, 1995. Also to appear in Information Processing Letters.
8. P. Eades, A. Symvonis, S. Whitesides, *Two Algorithms for Three Dimensional Orthogonal Graph Drawing*, Proc. of Workshop GD '96, Lecture Notes in Comp. Sci. 1190, Springer-Verlag, 1996, pp. 139-154.
9. A. Garg and R. Tamassia, *GIOTTO3D: A System for Visualizing Hierarchical Structures in 3D*, Proc. of Workshop GD '96, Lecture Notes in Comp. Sci. 1190, Springer-Verlag, 1996, pp. 193-200.
10. Goos Kant, *Drawing Planar Graphs Using the Canonical Ordering*, Algorithmica, vol. 16, no. 1, 1996, pp. 4-32.
11. A. N. Kolmogorov and Y. M. Bardzin, *About Realization of Sets in 3-dimensional Space*, Problems in Cybernetics, 1967, pp. 261-268.
12. J. MacKinley, G. Robertson, S. Card, *Cone Trees: Animated 3d visualizations of hierarchical information*, In Proc. of SIGCHI Conf. on Human Factors in Computing, pp. 189-194, 1991.
13. A. Papakostas and I. G. Tollis, *Algorithms for Area-Efficient Orthogonal Drawings*, Technical Report UTDCS-06-95, The University of Texas at Dallas, 1995.
14. A. Papakostas and I. G. Tollis, *Issues in Interactive Orthogonal Graph Drawing*, Proc. of Workshop GD '95, Lecture Notes in Comp. Sci, 1027, Springer-Verlag, 1995, pp. 419-430.
15. A. Papakostas and I. G. Tollis, *A Pairing Technique for Area-Efficient Orthogonal Drawings*, Proc. of Workshop GD '96, Lecture Notes in Comp. Sci. 1190, Springer-Verlag, 1996, pp. 355-370.
16. A. Papakostas, J. Six and I. G. Tollis, *Experimental and Theoretical Results in Interactive Graph Drawing*, Proc. of Workshop GD '96, Lecture Notes in Comp. Sci. 1190, Springer-Verlag, 1996, pp. 371-386.
17. A. Papakostas and I. G. Tollis, *Incremental Orthogonal Graph Drawing in Three Dimensions*, Technical Report UTDCS-02-97, The University of Texas at Dallas, 1997. (available through www.utdallas.edu/~tollis)
18. S. Reiss, *An engine for the 3D visualization of program information*, J. Visual Languages and Computing, vol. 6, no. 3, 1995.
19. Markus Schäffter, *Drawing Graphs on Rectangular Grids*, Discr. Appl. Math. 63 (1995) pp. 75-89.
20. R. Tamassia and I. Tollis, *Planar Grid Embeddings in Linear Time*, IEEE Trans. on Circuits and Systems CAS-36 (1989), pp. 1230-1234.