Journal of Graph Algorithms and Applications

http://www.cs.brown.edu/publications/jgaa/

vol. 3, no. 4, pp. 81-115 (1999)

Algorithms for Incremental Orthogonal Graph Drawing in Three Dimensions

Achilleas Papakostas

Network Management Division
NEC America
1525 W. Walnut Hill Ln.
Irving, TX 75038
papakos@asl.dl.nec.com

Ioannis G. Tollis

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

Abstract

We present two algorithms for orthogonal graph drawing in three dimensional space. For a graph with n vertices of maximum degree six, the 3-D drawing is produced in linear time, has volume at most $4.63n^3$ and has at most three bends per edge. 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.

Communicated by G. Di Battista and P. Mutzel.
Submitted: March 1998. Revised: November 1998 and April 1999.

Research supported in part by NIST, Advanced Technology Program grant number 70NANB5H1162, and by the Texas Advanced Research Program under Grant No. 009741-040.

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 book by Di Battista, Eades, Tamassia and Tollis [10]. An *orthogonal* drawing is a drawing in which vertices are represented by points of integer coordinates and edges are represented by polygonal chains consisting of horizontal and vertical line segments. Various algorithms have been introduced to produce orthogonal drawings of planar [2, 15, 20, 34, 36] or general [2, 24, 27, 33] graphs of maximum degree 4, and maximum degree 3 [20, 23, 24]. All these algorithms run in linear time, except for the algorithm in [34]. For drawings of general graphs, the required area can be as little as $0.76n^2$ [24, 27], the total number of bends is no more than $2n+2$ [2, 24, 27], and at most two bends can be on the same edge [2, 24, 27].

There has been a recent trend in Graph Drawing to visualize graphs in the three dimensional space. Although the number of applications that require such a representation for graphs is still limited [4, 17, 22, 31, 32, 37, 38], there is no doubt that 3-D Graph Drawing will find many applications in the future.

A number of software systems that produce straight-line 3-D drawings of graphs have been introduced. In the case of [5], the system is based on the *spring-embedder* paradigm [19]. Spring-embedders use a physical model based on vertices treated as currents exerting a repulsive force, while edges are modeled as forces attracting the vertices they combine.

Simulated Annealing has also been used [8] to produce straight-line 3-D drawings of graphs. The idea here is that there is a predefined cost associated with the current 3-D drawing of the graph, and the system moves to drawings of lower costs (while sometimes accepting higher cost drawings if they look 'nice'), until no further improvement is possible. Other special purpose systems are described in [11, 32].

Little is known about the theory of 3-D Graph Drawing. The concept of a visibility representation of a graph [35] has been extended to 3-D space, known as 3-D visibility representations of graphs. Research in this area [1, 3, 6, 16, 18] has revealed characterizations of several families of graphs and other theoretical results. In [7] it is shown that an n-vertex graph has a non-orthogonal 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.

Naturally, orthogonal drawing in three dimensional space has also received attention recently [4, 5, 8, 12, 13, 17, 32]. A 3-D orthogonal drawing typically has the following properties:

- Vertices are points with integer coordinates in three dimensional space.
- 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.
- 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 6 is shown in [13]. 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 [12], where the volume upper bound was the same but the drawings allowed up to 16 bends per edge. If we require that each edge has at most three bends, then another algorithm is presented in [13] 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 [21] show an existential lower bound of $\Omega(n^{\frac{3}{2}})$ on the volume occupied by 3-D orthogonal grid drawings of graphs of maximum degree 6.

In this paper we present an algorithm for producing 3-D orthogonal drawings of simple graphs of maximum degree 6, and a second algorithm that produces 3-D orthogonal drawings of simple 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 [25, 26, 28]. 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 6, our first algorithm produces a 3-D orthogonal drawing of G whose volume is at most $4.63n^3$, in linear time. Moreover, each edge of the drawing has at most three bends. Hence, our algorithm outperforms the algorithm of [13] 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 6, there is a 3-D orthogonal drawing according to the definition of the previous section. 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.

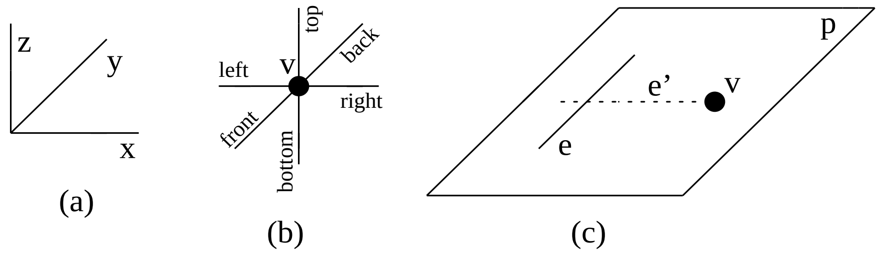


Figure 1: (a) Coordinates system for 3-D drawing, (b) possible directions from where an edge can enter v, (c) v's left free direction is blocked.

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* (extending towards the positive part of the x-axis), 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 free direction of v *is blocked by straight line segment* e, if we can draw a straight line from v along the free direction that intersects e. Such a situation is depicted in Fig. 1c: v and e are both placed in plane p which is parallel to the xy-plane. v's left free direction is blocked since line e' (which is parallel to the x-axis and extends towards the negative part of this axis) crosses line e.

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 *length* of a 3-D drawing is the maximum distance between two planes parallel to the yz-plane containing any part of the drawing. The *width* of a 3-D drawing is the maximum distance between two planes parallel to the xz-plane containing any part of the drawing. The *height* of a 3-D drawing is the maximum distance between two planes parallel to the xy-plane containing any part of the drawing. If a 3-D drawing has length l, width w and height h, its *volume* is $l \times w \times h$. It is actually the volume of the smallest rectangular parallelepiped that encloses the 3-D drawing.

Before we present our two algorithms, we repeat very briefly the following interactive graph drawing terminology: The current drawing is the drawing before the insertion of new vertex v; the number of vertices of the current drawing that are going to be connected with v through new edges, is v's local degree. We call these vertices *adjacent* vertices of v. Finally, a plane is *to the top of the topmost plane of the current drawing,* if it is parallel to the xy-plane and located one unit above the point of the current drawing with the highest z-coordinate. This notion extends similarly to the other directions.

3 Drawing Graphs with Maximum Degree Six

In this section we present our incremental algorithm for producing orthogonal drawings of graphs of maximum degree 6 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 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 6.
- The graph represented by the current drawing is always connected.

In the following two subsections, we describe how a new vertex is placed in the current drawing and how its (at most six) incident edges are routed. Vertices are represented by points. Our technique follows the Relative-Coordinates scenario. This means that the decision about where a new vertex will be placed 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 [25, 26, 28] 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 is 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 in 3-D follow one of five fundamental routes that we now explain. Assume that w and w' are two vertices of the current drawing. In the first three fundamental routes, edge (w', w) always enters w' from its left free direction.

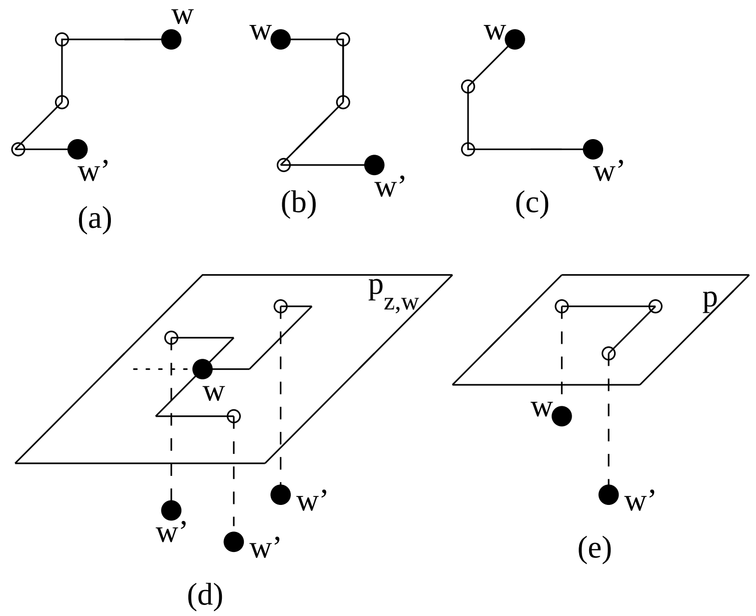


Figure 2: (a) First, (b) Second, (c) Third Fundamental Routes, (d) Same-Plane, (e) Over-The-Top Routes.

- **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: $w' \to p \to p_{y,w} \to p_{z,w} \to w$.

- **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: $w' \to p_{x,w} \to p_{z,w} \to w$.

Although we used specific free directions for both w and w' in order to describe the first three fundamental routes, we must stress that these routes generalize to other situations as well. More specifically, if we derive the symmetric of the route shown in Fig. 2a with respect to the xy-plane or yz-plane, we have new legal routes which still fall within the First Fundamental Route category. Also, rotating the configuration of Fig. 2a or a symmetric of it by a multiple of a right angle around the z-axis produces additional legal routes of the First Fundamental Route type. In the same way, we can use symmetry and/or rotation in the way described before to produce additional legal configurations for the Second and Third Fundamental Routes.

In the remaining two fundamental routes, edge (w', w) enters w' from its top free direction. We also assume that w has higher z-coordinate than 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: $w' \to p \to p_{x,w} \to p_{y,w} \to w$. 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.

Note that if more than one edge is routed to vertex w using the Same-Plane Route, we have to make sure that: (a) There are no crossings between any two portions of these edges lying in plane $p_{z,w}$, and (b) no portion of such an edge in $p_{z,w}$ blocks any one of w's remaining (i.e., after all edges are routed) free directions. These requirements can always be satisfied as long as: (a) the points in $p_{z,w}$ where the edges intersect $p_{z,w}$ have general position (i.e., no two points are in the same row or column of $p_{z,w}$), and (b) there are no portions of non Same-Plane Routes in plane $p_{z,w}$. In Fig. 2d, we show how three edges starting from generally positioned points are routed to w with at most two bends. Note that w's left free direction is not blocked as a result of the routing.

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 0. If v_2 is the second vertex to be inserted, then v_2 has local degree 1 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) Determine connectors (one for each adjacent vertex) by using the procedure described below.
   (c) Find which Routing Case v's insertion falls into.
   (d) WITHIN a Routing Case:
      i. IF Routing Case 1, THEN determine anchor vertex u_a.
      ii. IF Routing Case 2 or 3, THEN
         A. IF degree of v is 6, THEN determine cover vertex u_c.
         B. Determine anchor vertex u_a.
      iii. Place v.
      iv. Route edge (u_a, v).
      v. Route remaining edges (u_i, v) except (u_c, v), using the three Fundamental Routes and/or the Same-Plane Route.
      vi. IF Routing Case 1, THEN determine cover vertex u_c.
      vii. Route edge (u_c, v).
3. END.

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.

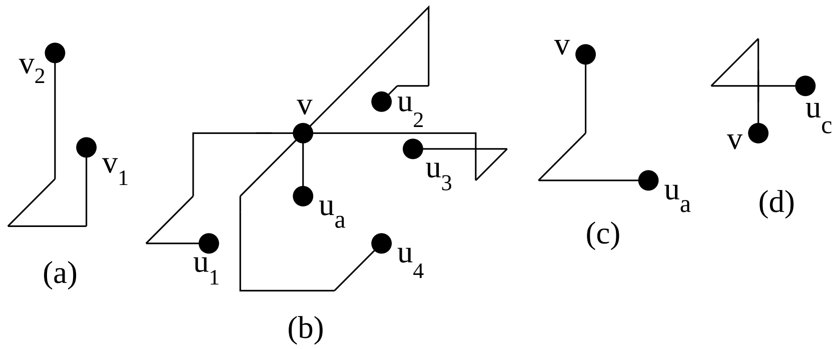


Figure 3: (a) Inserting the first two vertices, (b) a Routing Case 1 example, (c) (u_a, v) of Routing Case 1 when u_a does not have top connector, (d) (u_c, v) of Routing Case 2 when u_c does not have top connector.

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. $c_i := fd_i$.
   (b) RETURN.

3. 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. $c_i := fd_i$.
   (d) RETURN.
4. END.

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:

Routing Case 1: There is no beam among connectors c_i. We distinguish the following subcases for selecting the anchor vertex u_a:

1. First Subcase: There is at least one adjacent vertex with top connector. We consider the following cases:
   - There is exactly one adjacent vertex with plane connector. If this is left (front) or right (back) connector, then anchor u_a is the adjacent vertex with top connector whose $y(x)$-coordinate is the median of the $y(x)$-coordinates of all adjacent vertices with top connectors.
   - If there are exactly two adjacent vertices with plane connectors, we have two situations:
     - One plane connector is left (front) and the other one is right (back). Anchor u_a is the vertex with top connector whose $y(x)$-coordinate is the median of the $y(x)$-coordinates of all adjacent vertices with top connectors.
     - The two plane connectors are orthogonal. If one of them is a left (right) plane connector, then anchor u_a is the vertex with top connector having the lowest (highest) x-coordinate of all adjacent vertices with top connectors.

- If there are exactly three adjacent vertices with plane connectors, we find the plane connector which is orthogonal to the other two. If this is a left (resp. back, right, front) connector, then anchor u_a is the adjacent vertex with top connector having the lowest x-coordinate (resp. highest y-coordinate, highest x-coordinate, lowest y-coordinate) of all adjacent vertices with top connectors.
- In any other situation, any adjacent vertex with top connector can be the anchor u_a.

We insert a new plane p to the top of the topmost plane of the current drawing. Vertex v is placed at the intersection of planes p, p_{x,u_a}, p_{y,u_a}, directly above u_a. Edge (u_a, v) is a simple straight line segment from u_a to v (see Fig. 3b).

2. Second Subcase: There is no adjacent vertex having top connector. In this situation, any adjacent vertex can be the anchor vertex u_a. Assume that u_a has left connector. We insert three new planes p_1, p_2, p_3 so that p_1 (p_2) (p_3) is to the left (back) (top) of the leftmost (backmost) (topmost) plane of the current drawing. Vertex v is inserted at the intersection of planes p_1, p_2, p_3. Edge (u_a, v) is routed with two bends (see Fig. 3c) as follows: $u_a \rightarrow p_1 \rightarrow p_2 \rightarrow v$. This generalizes to cases where u_a has a different plane connector, through a rotation.

Any edge (u_i, v) where u_i is not the anchor is routed in one of the following ways: If u_i has plane connector, edge (u_i, v) is routed with three bends using the First Fundamental Route (see Fig. 3b for a complete example). If u_i has top connector, edge (u_i, v) is routed with two or three bends using the Same-Plane Route. Finally, note that if $l = 6$, cover vertex u_c has top connector and is routed with three bends using the Over-The-Top Route.

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 any such vertex can be 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.

We insert three new planes p_1, p_2, p_3 parallel to the three base planes, so that p_1 is to the left of the leftmost plane of the current drawing. If there is at least one adjacent vertex with top connector or if $l < 6$, then plane p_3 is to the top of the topmost plane of the current drawing. Otherwise, (i.e., $l = 6$ and there is no adjacent vertex with top connector), we insert p_3 one unit to the bottom of u_c (note that p_3 is parallel to the xy-plane). We distinguish the following subcases for plane p_2 which is parallel to the xz-plane:

1. First Subcase: There are no two orthogonal connectors. If B_{max} has cardinality 2, then plane p_2 is placed one unit to the front of adjacent vertex u_i which has top connector, is not the cover (if there is one), and has the highest y-coordinate among all adjacent vertices with top connectors. If B_{max} has cardinality higher than 2 or if such a vertex u_i does not exist, then plane p_2 is placed one unit to the back of u_a. Vertex v is placed at the intersection of planes p_1, p_2, p_3. Edge (u_a, v) is routed with two bends (see Fig. 3c) as follows: $u_a \to p_1 \to p_2 \to v$.

2. Second subcase: There is one pair of orthogonal beams, or there is only one beam and it is orthogonal to at least one connector. This means that B_{max} is orthogonal either to another beam (which we assume is a back beam) or to some connector c_i (which we assume is a back connector). Plane p_2 is placed to the back of the backmost plane of the current drawing. Placing v and routing edge (u_a, v) is done as in the First Subcase.

We use the First, Second, and Third Fundamental Routes to route the remaining edges (u_i, v) where u_i has a plane connector. These edges eventually attach only to plane free direction of v. Note that the Third Fundamental Route can be used only by vertices which are origins of B_{max}. Edges that come from adjacent vertices having top connectors are routed using the Same-Plane Route. If $l = 6$ and cover u_c has top connector, then edge (u_c, v) is routed using the Over-The-Top Route.

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, 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 (back) connector, it is routed with two bends as follows: $u_c \to p_1(p_2) \to p_2(p_1) \to v$. Figure 3d shows an example of this route when u_c has left connector.

Routing Case 3: There are two opposite beams among connectors c_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}. We first discuss the situation where (a) $l \leq 5$, or (b) $l = 6$ and there is at least one adjacent vertex with top connector (this vertex is cover u_c and edge (u_c, v) is routed using the Over-The-Top Route).

Anchor u_a is the median of the origins of beam B_{max} with respect to their y-coordinates. We open up three new planes p_1, p_2, p_3 as follows: p_1 is to the left of the leftmost plane of the current drawing, and p_3 is to the top of the topmost plane of the current drawing. Plane p_2 is parallel to the xz-plane and between the two origins of beam B. Vertex v is inserted at the intersection of planes p_1, p_2, p_3. We route edge (u_a, v) with two bends (see Fig. 3c) as follows: $u_a \to p_1 \to p_2 \to v$.

If B_{max} has two origins, then there can be at most one adjacent vertex u' with front, back, or top connector (u' is not the cover vertex). If u' has front or

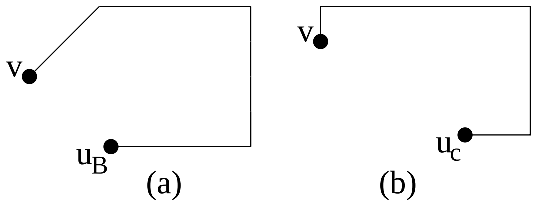


Figure 4: Routing Case 3: (a) routing the edge coming from vertex u_B when u_B is an origin of the beam opposite to B_{max}, (b) edge (u_c, v) when u_c does not have top connector.

back connector, then edge (u', v) is routed using the First Fundamental Route. If u' has top connector, then edge (u', v) is routed using the Same-Plane Route. In this case, edge (u', v) enters vertex v from its front (back) free direction if the y-coordinate of u' is lower (higher) than that of v.

Consider the case where B_{max} has three origins. If the y-coordinate of anchor u_a is lower (higher) than that of v, let u' be the origin of B_{max} whose y-coordinate is lower (higher) than that of u_a. Edge (u', v) is routed using the Third Fundamental Route entering v from its front (back) free direction.

The remaining incident edges of v are routed as follows: Let u'' be B_{max}'s origin whose edge has not been routed yet. We route edge (u'', v) using the First Fundamental Route. As a result of routing v's incident edges so far, either v's back or v's front free direction is available. If v's back (front) free direction is available, let u_B be the origin of B having higher (lower) y-coordinate than that of v. We open up a new plane p' to the right of the rightmost plane of the current drawing, and route edge (u_B, v) with three bends (see Fig. 4a) as follows: $u_B \rightarrow p' \rightarrow p_3 \rightarrow p_1 \rightarrow v$.

Note that edge (u_B, v) is routed on top of the current drawing, all the way from the rightmost to the leftmost side of the drawing. Although this edge passes directly over u_B, this will not create any crossings in the future since u_B's top free direction is not available. Let u'_B be the other origin of B. Edge (u'_B, v) is routed using the First Fundamental Route.

Let us now consider the situation where $l = 6$ and there is no adjacent vertex with top connector. If B has cardinality 3, then cover u_c is the origin of B which is between the other two origins of B with respect to the y-coordinate. If B has cardinality 2, then cover u_c is the origin of B with the lowest (highest) y-coordinate of the two if there is an adjacent vertex with front (back) connector. Anchor u_a is the median of the origins of B_{max} with respect to the y-coordinate. Planes p_1, p_3 are inserted as described in the previous case, and plane p_2 is identical to p_{y,u_c}. Vertex v is placed at the intersection of planes p_1, p_2, p_3, and edge (u_a, v) is routed with two bends as described in the previous case.

If there are adjacent vertices with front or back connectors, the edges that come from them are routed using the First Fundamental Route. Any edge coming from an origin of B (different from u_c) is routed in the way u_B was routed (see Fig. 4a), only if the front and/or back free directions of v are available. If neither of these two free directions of v is available, then the edge coming from B's origin (different from u_c) is routed to the right free direction of v using the First Fundamental Route. Edges coming from the origins of B_{max} are routed using the First, Second, and Third Fundamental Routes.

Two new planes p_c and p'_c are inserted so that p_c is to the right of the rightmost plane of the current drawing and p'_c is parallel to p_3 and one unit to the top of it. Edge (u_c, v) is routed with three bends (see Fig. 4b) as follows: $u_c \rightarrow p_c \rightarrow p'_c \rightarrow p_1 \rightarrow v$. The top free direction of u_c is not available, so there can be no future edge incident to u_c crossing edge (u_c, v). Finally, note that since placing v and routing edge (u_a, v) requires only two new planes (i.e., p_1, p_3), the total number of new planes that we open up in order to route all six of v's incident edges is at most eight.

3.3 Routing Properties - Analysis

Placing a new vertex v and routing its incident edges has several cases which were described in detail in the previous subsection. We believe that this description of all the involved cases was necessary especially if this algorithm is to be implemented. We saw that vertex v is usually placed at the intersection point of three planes, at least two of which are new.

Placing a vertex v directly to the top of vertex u (when routing edge (u, v)) happens only when u had current degree 5 before this edge insertion. This way, edges that enter u and v from their same plane free direction cannot cross. In any other case that two vertices have the same either x or y-coordinate, the one with the lower z-coordinate does not have an available top free direction (see Routing Case 3). Note that, although it is possible for two vertices to have the same x and/or y-coordinate, there are never two vertices with the same z-coordinate.

An important feature of our edge routing technique is that no two edges cross. The route of each edge is naturally decomposed in three different stages. Let us consider an edge e from u to v, where v is the vertex most recently placed in the drawing. The first stage contains the portion of the route that lies entirely in plane $p_{z,u}$. This portion consists of a straight line segment that goes all the way until it hits a new plane positioned outside the current drawing. This line segment does not cross any other edges in its way, because no available free direction is ever blocked.

When it hits the new plane, the line segment of the first stage may bend staying always in plane $p_{z,u}$. This new segment does not cross any other edge either, since it runs entirely outside the current drawing. In the second stage, we have the portion of the route consisting of a straight line segment running vertically from plane $p_{z,u}$ to plane $p_{z,v}$. This segment always lies entirely outside the current drawing, so it does not cross any other edge.

Finally, the third stage consists of the portion of the route that runs entirely in plane $p_{z,v}$. This portion is typically a straight line segment entering v. If this portion needs to have one or two bends in plane $p_{z,v}$, then the routing is done without (a) crossing portions of other edges lying in $p_{z,v}$, and (b) blocking v's free directions that may be used in the future (see Same-Plane Route and Routing Case 3).

If there is a portion of the third stage lying in $p_{x,v}$ or $p_{y,v}$, then it does not cross line segments of other edges that are perpendicular to $p_{z,v}$ since neither $p_{x,v}$ nor $p_{y,v}$ can be used to route edges that are not incident to v. The only exception to this would be the case where there was a vertex w so that v was placed directly to the top of w. But in that case, w would have degree 6 that is no future incoming vertices will be adjacent to it.

If there is a portion s of the third stage lying in $p_{x,u}$ or $p_{y,u}$, then we distinguish two cases: In the first case, edge e leaves u from its top free direction. The portion of any future edge (u, v') which is perpendicular to $p_{z,v}$ will lie outside the current drawing, so it cannot cross s. In the second case, edge e leaves u from one of its plane free directions (in other words, we have Routing Case 3). In this case, u's top free direction was used in the past to route some other edge e'. The whole route of e' lies entirely below v, so no portion of this route can cross s. Hence we have:

Lemma 1 *Our routing technique for 3-D orthogonal graph drawing guarantees that each edge has at most three bends and no two edges cross (in a current drawing).*

In order to measure the volume of the current 3-D drawing at time t, we count the total number of planes that were inserted in the drawing up to time t. Placing v and routing edge (u_a, v) requires two or three new planes. Routing any other edge (u_i, v) (where u_i is not the cover) adds at most one new plane. Routing edge (u_c, v) requires one or two new planes. However, whenever two new planes are required to route edge (u_c, v), placing v and routing edge (u_a, v) requires only two new planes (see Routing Case 3).

From this it follows that if v's local degree is $l \geq 1$, then at most $l + 2$ new planes need to be inserted when v is placed in the current drawing. Vertex v_2 (i.e., the second vertex to be inserted in the drawing) is the only exception and requires four new planes, although it has local degree 1 (see Fig. 2a). Let $n_1(t), n_2(t), n_3(t), n_4(t), n_5(t), n_6(t)$ be the total number of vertices with local degree 1, 2, 3, 4, 5, 6, respectively, and $n(t)$ be the total number of vertices of the current drawing at time t. Also recall that v_1 (i.e., the first vertex to be inserted) introduces three planes. If $P(t)$ is the total number of planes inserted in the drawing at time t, then we have:

$$P(t) \quad \leq \quad 3 + 3n_1(t) + 1 + 4n_2(t) + 5n_3(t) + 6n_4(t) + 7n_5(t) + 8n_6(t)$$

$$\begin{aligned}
&\leq (n_1(t) + 2n_2(t) + 3n_3(t) + 4n_4(t) + 5n_5(t) + 6n_6(t)) \\
&\quad +(2n_1(t) + 2n_2(t) + 2n_3(t) + 2n_4(t) + 2n_5(t) + 2n_6(t)) + 4 \\
&\leq 3n(t) + 2(n(t) - 1) + 4 \\
&\leq 5n(t) + 2.
\end{aligned}$$

Let $P_x(t), P_y(t), P_z(t)$ be the total number of planes perpendicular to the x, y, z-axes, respectively, inserted in the current drawing up to time t, so that $P_x(t) + P_y(t) + P_z(t) = P(t)$. The volume $V(t)$ of the current drawing at time t is: $V(t) = (P_x(t) - 1) \times (P_y(t) - 1) \times (P_z(t) - 1)$, and it is maximized when: $P_x(t) = P_y(t) = P_z(t) = \frac{P(t)}{3} = \frac{5n(t)+2}{3}$. From this it follows that $V(t) \leq (\frac{5n(t)+2}{3} - 1)^3 \leq (\frac{5n(t)}{3})^3 \approx 4.63n^3(t)$.

Clearly, each edge route has a constant number of bends and each vertex insertion requires a constant number of planes to be added to the current drawing. However, in order to guarantee constant time when a plane is inserted at a particular position in the middle of the current drawing, we have to use the data structure by Dietz and Sleator [9]. A vertex insertion can be completed in constant time as long as the system does not have to produce the new drawing. If a drawing is required, then the time is linear per vertex insertion operation. From the above discussion and Lemma 1 we have:

Theorem 1 *There is a 3-D orthogonal graph drawing algorithm for graphs of maximum degree 6 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 $4.63n^3(t)$, 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, according to its number. A numbering with the following property will guarantee that each placed vertex has local degree at least 1: If j ($j > 1$) is the number assigned to vertex v_j, then there is at least one edge (v_i, v_j) in the graph, where $i < j$. Consider the directed acyclic graph that results from the given graph when each edge is directed from the lower to the higher

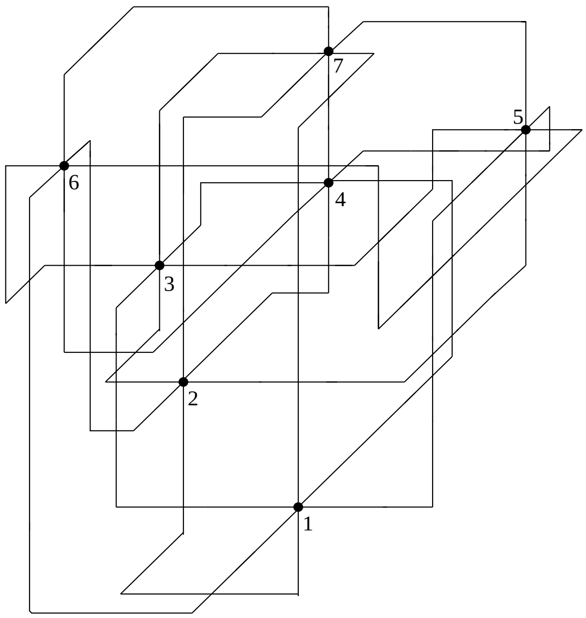


Figure 5: 3-D orthogonal drawing of K_7 produced by our algorithm.

numbered vertex. What the above condition means is that the resulting directed graph has one source and at least one sink. A simple Depth-First-Search (DFS [14]) or Breadth-First-Search (BFS [14]) can provide such a numbering for any connected graph. Hence we have the following:

Theorem 2 *Consider an n-vertex connected graph G of maximum degree 6. 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* $4.63n^3$.

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. The algorithm has been implemented within 3DCube [29] and we present preliminary experimental results in the Conclusions.

4 A Model for Vertices of High Degree

In this section we describe a model to support high degree interactive 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. At any time there is a change in the drawing, our target is to maintain the general shape of the current drawing. Our model and the drawing algorithm which is based on it, apply also to non-interactive settings. In these settings, the whole graph is known in advance and the user provides the order in which the vertices are considered for placement.

The first issue that has to be addressed is the way that vertices are represented. In the previous section, it sufficed to map vertices to points in space since the degree of any vertex was at most 6 at any time during the drawing process. Clearly, we now need a different approach to accommodate arbitrary vertex degrees. We choose to represent vertices using 3-D boxes of volume (initially) at least one cubic unit, regardless of the degree of the vertices.

When a vertex is inserted into the drawing, it is represented by a cubic box with 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. Each box has six *sides*, and each side is a rectangle parallel to one of the base planes. If $d \times d'$ is the size of one side of a box, then both d and d' are integers and there are $(d+1)(d'+1)$ connectors on this side.

Let us consider a vertex v and the box that represents it. The two sides of the box that are parallel to the xy-plane are called *top* and *bottom* sides, with top being the side located at a higher z-coordinate between the two. The two sides parallel to the yz-plane are called *left* and *right* sides, with right being the side located at a higher x-coordinate between the two. Similarly, the remaining two sides are the *front* and *back* sides, with back being the side located at a higher y-coordinate between the two. The six sides of box v are shown in Fig. 6a, and the connectors of the front side are shown in Fig. 6b, where v is a $4 \times 4 \times 4$ cube.

Note that all connectors located along the line where two sides meet are shared by both sides. Also, the single connector located at the point where three sides meet is shared by all three sides. The distance between the left and right sides is called *length* of the box. Similarly, the distance between the front and back sides is called *width*, and the distance between the top and bottom sides is called *height* of the box. In a $d_1 \times d_2 \times d_3$ box the length is d_1, the width is d_2 and the height is d_3.

Edge routing follows the Relative-Coordinates scenario, and tries to keep both volume and number of bends as low as possible. However, as a result of this 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 3-D space

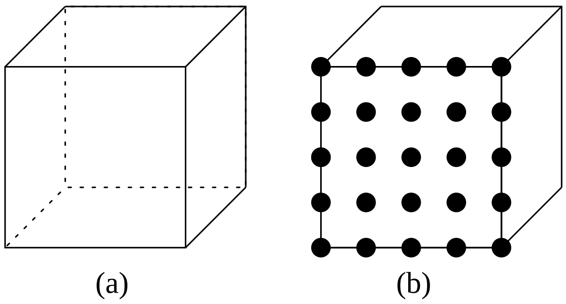


Figure 6: (a) Every box has six rectangular sides, (b) front side connectors of a box.

supports box growing.

Let us assume that we have a $d_1 \times d_2 \times d_3$ box representing vertex v. There are two ways to grow a box by increasing its length: one way is to increase the length towards the right direction (see Fig. 7a), and the other is to increase the length towards the left direction (see Fig. 7b). In either case, the result is a $(d_1 + 1) \times d_2 \times d_3$ box v. Also notice that the resulting box has:

- $d_2 - 1$ new connectors on its top side,
- $d_2 - 1$ new connectors on its bottom side,
- $d_3 - 1$ new connectors on its front side,
- $d_3 - 1$ new connectors on its back side,
- one new connector shared by the top (bottom) and back (front) sides,
- one new connector shared by the top (bottom) and front (back) sides.

Similarly, we can grow box v by increasing its width by one unit (towards the front or the back direction), or its height by one unit (towards the top or the bottom direction). Regardless of the way we choose to grow a box, we always insert a new plane in the middle of the current 3-D drawing. This insertion affects the coordinates of some connectors and bends which shift by one unit along the x, y or z-axes. The general shape of the drawing, though, remains the same. Finally, note that although the box of every vertex starts out having a cubic configuration (when it is inserted for the first time), this is not necessarily the case later. This happens because a box may grow its size in several different ways in the course of the drawing process.

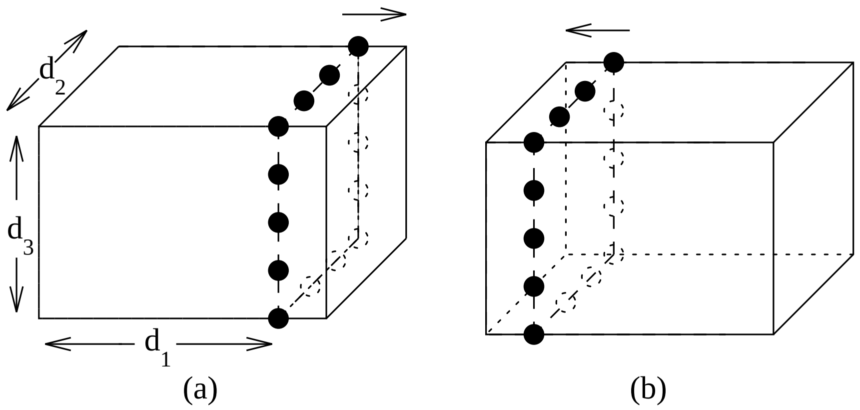


Figure 7: (a) Increasing the length of a box towards the (a) right or (b) left direction creates new connectors.

5 Drawing Graphs of High Degree

In this section we present a drawing algorithm based on the Relative-Coordinates scenario for producing orthogonal drawings of graphs in the three dimensional space. Since this algorithm allows vertices to arrive on-line, the order of vertex insertion is decided by the user. In order to simplify our presentation, we assume that the local degree of any vertex that is about to be inserted cannot be greater than 16. We will discuss how to easily generalize our technique at the end of this section. We compute bounds on the volume and the total number of bends that each (current) drawing requires at time t, when we start the drawing from scratch.

A typical user request consists of the name of the vertex to be inserted, say v, and a list of its adjacent vertices in the current drawing. Our algorithm produces a 3-D drawing considering and placing one vertex at a time. The placement of vertex v follows the Relative-Coordinates paradigm. In other words, it tries to maintain the general shape of the current drawing after v is inserted. For this reason, the selection of the position where v will be placed depends on the following factors:

- Vertex v's local degree.
- The connector availability situation on the sides of the boxes of v's adjacent vertices.
- The relative position of v's adjacent vertices in the current drawing.

5.1 Placing a New Vertex

Again, assume that v is the next vertex to be inserted in the current 3-D drawing. Let k ($k \leq 16$) 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. Recall that some connectors are shared by two or even three sides. 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.

For example, assume that $k = 10$, there are three adjacent boxes that have available connectors on their back side, and for any other side (i.e., top, bottom, right, left, front) there are no more than two adjacent boxes having available connectors on that side. Then, for each one of the ten adjacent boxes, the edge that connects this box with v will attach to a back side connector of that box. If there is any adjacent box which does not have available connectors on its back side, then we grow the box. This can be done by increasing either the length or the height of the box by one unit, as described in the previous section. Such an increase will create more than one connectors which lie not only on the back side but on other sides as well. These connectors may be used at a later stage.

The next step is to create the box representing v and place it in the current 3-D drawing. Box v is a $1 \times 1 \times 1$ cube. Each newly inserted box is placed in such a way so that none of its connectors have the same x, y or z-coordinate as any other connector of any box of the current drawing. This means that our 3-D drawing has the property that there are no two connectors of two different boxes lying on a plane parallel to a base plane. We call this the *no-common-plane* property of the 3-D drawing. Next we show how v's exact position in the drawing is calculated so that this property is maintained.

For the sake of description, let us consider the case where all k edges connecting v with its adjacent vertices attach to the back sides of these vertices. Also, let $c_1, c_2, \cdots c_k$ be the connectors of the adjacent boxes where these edges will be attached. We insert a new plane p' to the back of the backmost plane of the current drawing (clearly, p' is parallel to the xz-plane).

We compute the projections of connectors $c_1, c_2, \cdots c_k$ on the xz-plane. Because of the no-common-plane property, there are no two projection points sharing the same row or column in the xz-plane. We find the projection point c'_i whose x-coordinate is the median of the x-coordinates of all projection points; if there are two medians, we take the one with the higher x-coordinate. Also, we find the projection point c'_j whose z-coordinate is the median of the z-coordinates of all projection points; in case there are two medians, we take the one with the higher z-coordinate. Note that it is possible for c'_i and c'_j to be the same point. In Fig. 8a we show points c'_i and c'_j chosen from a set of 11 points placed in the xz-plane.

If c'_i is the projection of connector c_i located on the back side of box u_i, we insert a new plane p_i in the 3-D drawing parallel to the left side of u_i at a

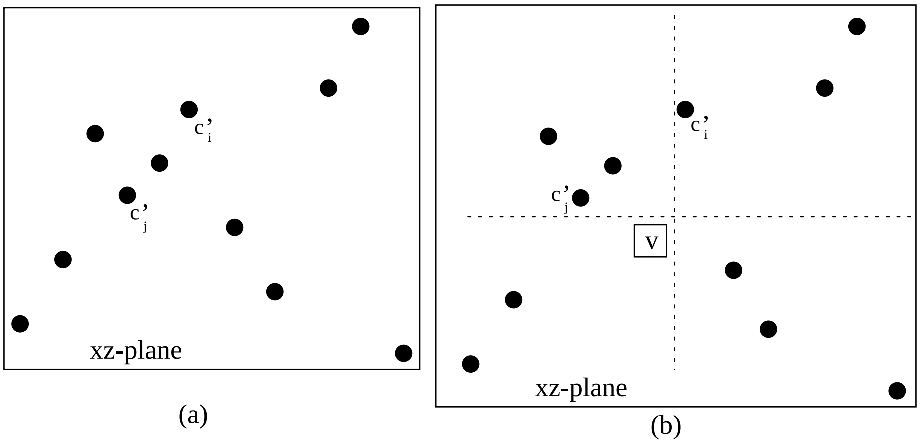


Figure 8: (a) Choosing the median points along x and z-axes, (b) projection of v's position and connectors on xz-plane.

distance of one unit from that. If c_j' is the projection of connector c_j located on the back side of box u_j, we insert a new plane p_j in the 3-D drawing parallel to the bottom side of u_j at a distance of one unit from that.

The three planes p', p_i and p_j intersect at a single point. This is the point where the connector shared by the top, right and front sides of box v is going to be placed. Notice that the exact coordinates of a specific connector of box v determine the location of v. We open up new planes (if required) in the middle of the drawing to accommodate the size of v. Figure 8b shows the relative position of v and the 11 projection points of the example of Fig. 8a. Note that Fig. 8b is the projection of that portion of the 3-D drawing on the xz-plane. Notice that there are no two points (with at most one of them belonging to box v) sharing the same row or column in the xz-plane (i.e., no-common-plane property).

Similarly, we compute the coordinates of a newly inserted box v in the cases where all connectors $c_1, c_2, \cdots c_k$ lie on the top, bottom, front, right or left sides of v's adjacent vertices. From the above discussion regarding the position where a new vertex is inserted, we have the following three propositions:

Proposition 5.1 *At any time t, the 3-D drawing has the no-common-plane property, that is there is no plane parallel to one of the three base planes containing two connectors of two different boxes.*

Proposition 5.2 *If $c_1, c_2, \cdots c_k$ are the connectors of the adjacent boxes of a newly inserted vertex v where v's incident edges will be attached, then all these* *connectors lie on the 'same' side of their boxes (i.e., they are all either on the top, bottom, right, left, front, or back sides of their boxes).*

Proposition 5.3 *Consider the case of placing a new vertex v with local degree k, so that the edges connecting v with its adjacent vertices attach to the back sides of the boxes of the adjacent vertices. Compute the projections of v and the connectors of the adjacent boxes where the edges attach, on the xz-plane. The following hold:*

- *there are $\lceil \frac{k}{2} \rceil$ ($\lfloor \frac{k}{2} \rfloor$) projection points lying strictly above (below) v's top (bottom) side,*
- *there are $\lceil \frac{k}{2} \rceil$ ($\lfloor \frac{k}{2} \rfloor$) projection points lying strictly to the right (left) of v's right (left) side.*

This generalizes accordingly depending on the way the new vertex v is placed in the 3-D drawing.

5.2 Routing Edges with Two Bends

The step that concludes the insertion of v is routing the edges that connect v with its adjacent vertices. For each edge e_i we have that one endpoint of the edge is connector c_i of adjacent box u_i, and the other endpoint is a connector of some side of v. The portion of edge e_i between its two endpoints is routed in an orthogonal fashion in three dimensional space. In the description that follows, we assume that v has been placed as described above, and we continue with the routing of v's incident edges.

Let us first route the edges that come from connectors having z-coordinate greater than the z-coordinate of any point on v's top side. Because of the no-common-plane property, some of these connectors have x-coordinate smaller than the x-coordinate of any point on v's left side, and the remaining connectors have x-coordinate greater than the x-coordinate of any point on v's right side. We call the connectors of the first group *outleft*, and the connectors of the second group *outright*.

Assume that the outleft connectors are fewer than the outright ones; this means that the number of the outleft connectors is less than $\lceil \frac{\lceil \frac{k}{2} \rceil}{2} \rceil$. Since k can be at most 16, we have that the outleft connectors are at most four, which is the maximum number of connectors lying on a single side of box v. The edges that start from outleft connectors are routed to connectors lying on the top side of box v.

More specifically, let us route edge e_i from outleft connector c_i to connector c_v of v (see Fig. 9a). From connector c_i, we draw a straight line parallel to the y-axis that intersects plane p_{y,c_v} at point b_i forming a right angle. Then we draw another straight line from b_i along plane p_{y,c_v} and parallel to the x-axis, that intersects plane p_{x,c_v} at point b'_i forming a right angle. Then, we simply

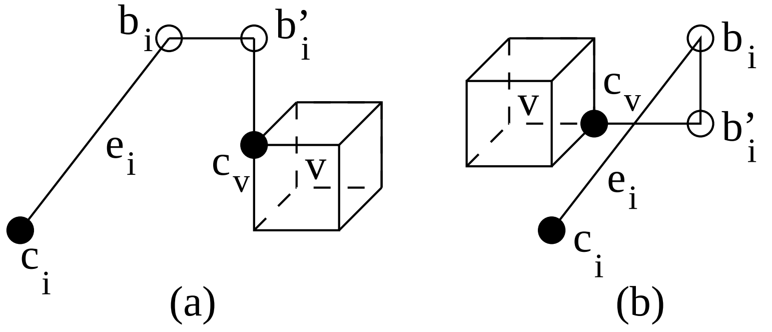


Figure 9: Routing edges to the top side of box v when c_i is to the (a) left and (b) right of v.

draw a straight line between b'_i and connector c_v. The full route is depicted in Fig. 9a. Observe that it follows the orthogonal paradigm. Also, edge e_i has exactly two bends, one at point b_i and one at point b'_i.

Connector c_v of v is picked so that line segment $b_i b'_i$ does not block the top free direction of any other connector of v (see Fig. 9a). Also, if it turns out that there is a connector c'_i of box u_i whose back free direction is blocked by line segment $b_i b'_i$, then we use c'_i as the other endpoint of edge e_i instead of c_i.

The edges that start from the remaining outleft connectors are routed in a similar fashion, that is with exactly two bends per route. The edges that start from the outright connectors are routed to connectors of the top side of box v that are still available, and the routing is done in the way described above. If there are still outright connectors whose edges have not been routed, then these edges are routed to connectors of the right side of v. In Fig. 9b we show an example of the routing of such an edge. Notice that the routing can still be done with two bends, and the line between the two bends is parallel to the z-axis.

If there are fewer outright than outleft connectors, then we first route the edges of the outright connectors to the top side of v, and then the edges of the outleft connectors to the top and (if necessary) left side of v. In either case, from Proposition 5.3 and the fact that v's local degree is at most 16, it follows that routing the edges of all the outleft and outright connectors requires at most two sides of v.

So far in this subsection we have discussed how to route edges that come from outleft and outright connectors with z-coordinates greater than the z-coordinate of any point on v's top side. We follow a similar procedure to route the edges coming from connectors whose z-coordinates are lower than the z-coordinate of any point on v's bottom side. This means that we first split these connectors into outleft and outright ones, find which one of the two sets has the smallest cardinality, and then route the individual edges using the connectors lying on the other two sides of v.

After the end of the routing of all incident edges of v, the insertion operation of v is complete. We used a specific situation for the position of v in order to describe our edge routing technique. It is easy to generalize this technique to any other case of box positioning in the 3-D drawing, as long as the principles and properties described in the following **routing lemma** are maintained:

Lemma 2 *Let v be the next vertex to be inserted in a 3-D drawing. Let e be the edge connecting v with its already placed adjacent vertex u. Let connector c_u of u be one endpoint, and connector c_v of v be the other endpoint of edge e. Also, let fd_{c_u} and fd_{c_v} be c_u's and c_v's free directions, respectively, used by e. The following hold:*

1. *If connector c_u belongs to side s_u which is perpendicular to fd_{c_u} and connector c_v belongs to side s_v which is perpendicular to fd_{c_v}, then the planes containing sides s_u and s_v are perpendicular to each other.*

2. *Edge e connecting c_u and c_v is routed orthogonally with exactly two bends.*

3. *If b_1 is the bend adjacent to connector c_u and b_2 is the bend adjacent to connector c_v then:*

    - *if ϵ is the straight line perpendicular to side s_u at point c_u and p is the plane parallel to side s_u containing c_v, then bend b_1 lies at the intersection of line ϵ and plane p,*

    - *if ϵ' is the straight line of plane p parallel to side s_v containing bend b_1 and ϵ'' is the straight line of plane p perpendicular to side s_v containing point c_v, then bend b_2 lies at the intersection of lines ϵ' and ϵ'',*

    - *line segment b_1b_2 does not block the free direction of any other connector of u and v.*

4. *The connectors of v used for routing all v's incident edges lie on at most four sides of v.*

5.3 No-Crossing Routing

Our routing scheme guarantees that each edge of every newly inserted vertex v can be routed so that there are no edge crossings in the resulting 3-D drawing. In order to show this, we first give some definitions.

Each edge consists of three straight line segments since it has two bends (see above lemma). We call these line segments *legs*. More specifically, a leg with a connector as one of its two endpoints is called an *end leg*, and a leg whose both endpoints are bend-points is called a *middle leg*. Let v be the next vertex to be inserted and e be the edge between v and an adjacent vertex u (u is already placed). We insert v and route e, according to the algorithm. We show that no leg of edge e crosses a leg of any edge of the current drawing.

Recall that if two line segments cross each other, then the four endpoints of the two segments lie in the same plane. Because of the no-common-plane property (see Proposition 5.1), it is never the case that an end leg of e crosses an end leg of any other edge of the current drawing. Otherwise we would have two connectors of two different boxes lying in a plane which is parallel to one of the base planes.

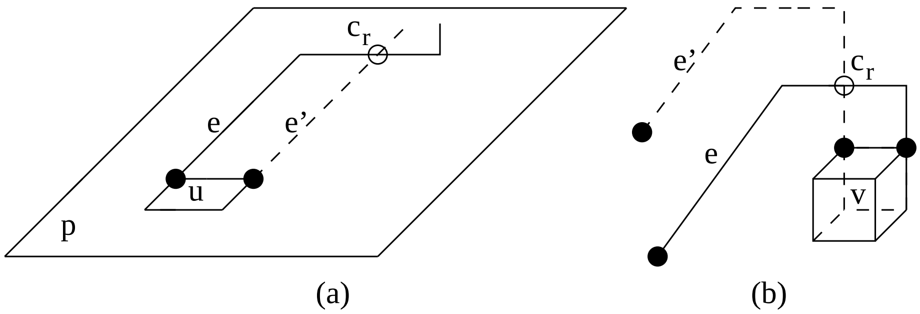


Figure 10: Crossings occurring when e blocks the free direction of other connectors.

The middle leg of edge e lies completely outside the current drawing. Because of this, it does not cross a leg of any other edge of the current drawing. Also, since the middle leg of edge e does not block the free direction of any connector of boxes u and v, crossing situations similar to the ones shown in Fig. 10 cannot happen. Figure 10a shows a situation where e blocks the back free direction of some connector of u, and Fig. 10b shows a situation where e blocks the top free direction of some connector of v. If a future edge e' uses these connectors, then e and e' will cross at point c_r. Therefore we have:

Lemma 3 *There are no edge crossings in any 3-D orthogonal drawing produced using the techniques for vertex placement and edge routing described in the previous sections.*

5.4 Volume and Bend Analysis

From the discussion in previous subsections we have that if $m(t)$ is the total number of edges in the drawing at time t, then the drawing has $2m(t)$ bends. In what follows we express both the bends and the volume of the drawing in terms of the number of vertices $n(t)$ and the average vertex degree $d_a(t)$ at time t.

We have assumed that every time a new vertex is inserted, its local degree is at most 16. Since this is the case, every vertex is represented by a $1 \times 1 \times 1$ box the moment it is placed in the drawing. Clearly, when a new box is placed, we open up two new planes parallel to each one of the three base planes. In other words, each of the length, width, and height of the drawing increases by two units.

When a new box is placed, we may have to grow the boxes of some adjacent vertices. We insert a new plane in the drawing when we grow a box. In the *worst case*, we have to grow the boxes of all adjacent vertices of each vertex inserted in the drawing up to time t. Assume that d_{loc} is the local degree of some newly inserted vertex, so that $d_{loc} \leq 16$. Let l, w and h be the numbers of new planes we open up as a result of vertex growth, parallel to the yz-plane, xz-plane, and xy-plane, respectively. In the worst case we have that: $l + w + h = d_{loc}$.

Lemma 4 *Let r be a positive number; let d be a positive constant so that integer variables x, y and z satisfy: $x+y+z = d$, where $x, y, z \geq 0$. Then it holds that: $(r+x)(r+y)(r+z) \leq (r+\frac{d}{3})^3$.*

Proof. It is well known that if we have three positive numbers a, b, c and a positive constant s, then it always holds that: $abc \leq (\frac{s}{3})^3$ as long as $a+b+c = s$. If we set: $a = r+x$, $b = r+y$, $c = r+z$ and $s = 3r+d$, we have the above lemma. □

The first vertex inserted to an empty drawing is a $1\times1\times1$ cube, so the volume of the drawing after this insertion is $1^3 = 1$. Let the volume of the current drawing right before the insertion of new vertex v be at most r^3. Then the volume of the resulting drawing after the insertion of v is at most $(r+\frac{d_{loc}}{3}+2)^3$ (see Lemma 4). The number 2 in the expression giving the volume comes from the size of the box of inserted vertex v. Routing v's incident edges does not affect the volume since all these edges are routed along existing planes. Also note that the above expression for the volume holds even if inserted vertex v has local degree 0.

Theorem 3 *There is an algorithm to produce 3-D orthogonal drawings of graphs (not necessarily connected) which allows vertices to arrive on-line. Each inserted vertex is adjacent to at most 16 other vertices at the time of insertion. 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 $(\frac{m(t)}{3}+2n(t))^3$, where $m(t)$ and $n(t)$ are the number of edges and vertices in the drawing at time t, respectively, and*

- *vertex insertion takes constant time.*

Proof. From the description of edge routing in Section 5.2 and Lemma 2 we have that every edge in the drawing has two bends. From Lemma 3 it follows that no two edges cross at any time t during the drawing process.

For every box v in the drawing, v's incident edges attach to connectors located on the sides of the box. Box v grows only when there are no available connectors on the side of v where one of v's incident edges needs to attach. The worst case happens when v's incident edges always attach to the same side of v; let s_v be this side. Also, let t be the current time and $deg_t(v)$ be the current degree of v. The surface of side s_v is at most $deg_t(v)$, and the total surface of all sides of v is at most $6 \times deg_t(v)$.

We saw above that the volume of the 3-D orthogonal drawing after the insertion of a vertex v is at most $(r + \frac{d_{loc}(v)}{3} + 2)^3$, where r^3 is the volume before the insertion and $d_{loc}(v)$ is v's local degree. Let $m(t)$ be the number of edges in the drawing at time t and $G(t)$ be the underlying graph. It holds that: $\sum_{v \in G(t)} \frac{d_{loc}(v)}{3} = \frac{m(t)}{3}$. Hence, we obtain the upper bound shown in the theorem for the volume of the drawing.

Let v be the next vertex to be inserted into the current drawing. Creating a box for v and placing v takes constant time. At most $d_{loc}(v)$ other boxes of the current drawing need to grow and exactly $d_{loc}(v)$ edges are routed as a result of v's insertion. Growing a box and routing an edge takes constant time, therefore v's insertion takes $O(d_{loc}(v))$ time. Since $d_{loc}(v) \leq 16$, this time is constant. □

In practice, we expect the volume to be smaller than the upper bound given in the above theorem. This is because our analysis assumed that for each vertex insertion the boxes of all the adjacent vertices had to grow. We expect a box to grow very infrequently, since each box has several connectors on its sides. If $d_a(t)$ is the average vertex degree of the graph represented by the drawing at time t, we have that $d_a(t)$ is given by: $d_a(t) = \frac{\sum_v deg_t(v)}{n(t)} = \frac{2m(t)}{n(t)}$, where $deg_t(v)$ is the degree of vertex v at time t. Another expression for the volume of the drawing is given by the following corollary:

Corollary 5.1 *If the average vertex degree at time t is $d_a(t)$, then a 3-D orthogonal drawing produced by our algorithm has volume $(\frac{d_a(t)+12}{6} n(t))^3$, in addition to the properties discussed in Theorem 3.*

It is worth noting that, in terms of volume, the performance of this algorithm for graphs of average degree 6, is the same or better than the one of the algorithm in [13]. For example, if the average vertex degree is 6, the volume of the drawing is at most $(3n)^3$, which is the same as the exact volume required by the algorithm in [13] for 3-D orthogonal graph drawing. However, the drawing approach of [13] represents vertices with points, handles only vertices of degree at most 6, allows three bends per edge, and requires that the whole graph be known in advance.

The algorithm for 3-D orthogonal graph drawing and the analysis we presented in this section apply to the case of a non-interactive setting as well. In this case, the whole graph is known ahead of time. Our algorithm produces a 3-D orthogonal drawing of the graph in $O(m)$ time (m is the number of edges in the graph), as long as it is supplied with any vertex numbering. This numbering determines the order of vertex placement. Figure 11 shows the 3-D orthogonal drawing of K_5 as produced by our algorithm. The box numbers denote the vertex insertion order.

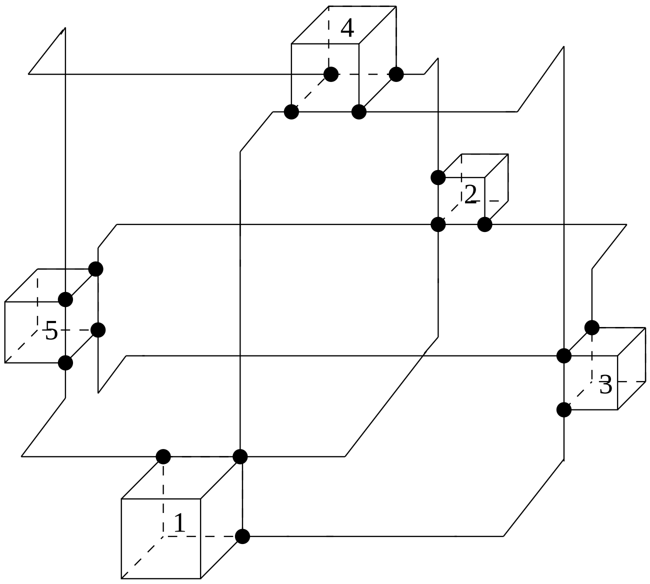


Figure 11: 3-D orthogonal drawing of K_5 representing vertices with boxes.

If we allow the insertion of a vertex v having local degree higher than 16, then the only thing that changes in the above algorithm is the size of the box representing v. Recall that a new box placed in the current drawing is always of cubic configuration. In addition to that, its incident edges attach to at most four of its six sides. A $2 \times 2 \times 2$ cube has nine connectors on each side and can represent a vertex with local degree at most 36. Similarly, a $3 \times 3 \times 3$ cube has 16 connectors on each side and can represent a vertex with local degree at most 64. In this way, we can accommodate the insertion of a vertex of any local degree.

The insertion of a $2 \times 2 \times 2$ box requires opening up three new planes parallel to each one of the base planes (nine new planes total). Similarly, the insertion of an $r \times r \times r$ box requires opening up $r + 1$ new planes parallel to each one of the base planes ($3(r+1)$ new planes total). From the volume analysis we presented in this subsection, we have that the general formula for the volume is: $V \leq (\frac{m}{3} + 2n_{1..16} + 3n_{17..36} + \cdots + rn_{4(r-1)^2+1..4r^2} + \cdots)^3$, where $n_{i..j}$ is the number of vertices in the current drawing at time t with local degrees in the range from i to j.

6 Conclusions and Open Problems

We presented two incremental algorithms for producing orthogonal graph drawings in three dimensional space with no edge crossings.

The first algorithm deals with simple graphs of maximum degree 6. For any n-vertex graph, the produced drawing has volume at most $4.63n^3$. No edge has more than three bends. This improves the best known [13] volume requirement of exactly $27n^3$ significantly, while maintaining the same upper bound for the number of bends per edge. It is also important to underline that our algorithm requires linear time to produce a 3-D orthogonal drawing of an n-vertex graph, whereas the algorithm in [13] runs in $O(n^{\frac{3}{2}})$ time.

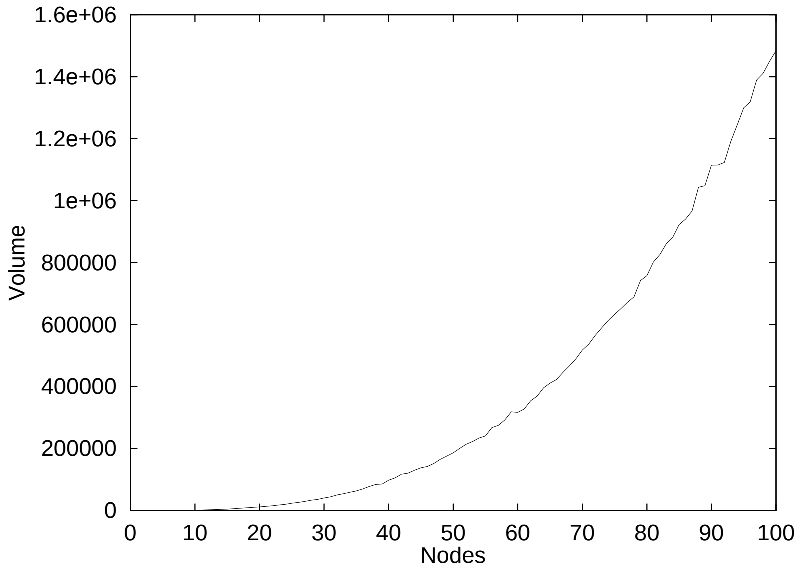
The algorithm has been implemented within 3DCube [29] along with the algorithms of [12, 13, 21] and a new algorithm introduced in [29]. The discussion here is based on the preliminary experimental results of the implementation of our algorithm sent to us by Maurizio Patrignani [30]. He applied our algorithm on 1920 random, simple, connected graphs of maximum degree 6, such that the number of edges is always twice the number of nodes. More extensive results on several 3D orthogonal algorithms will be presented in a forthcoming paper [30].

The graphs of the experimental results are presented in Figs. 12 and 13. These preliminary results indicate that:

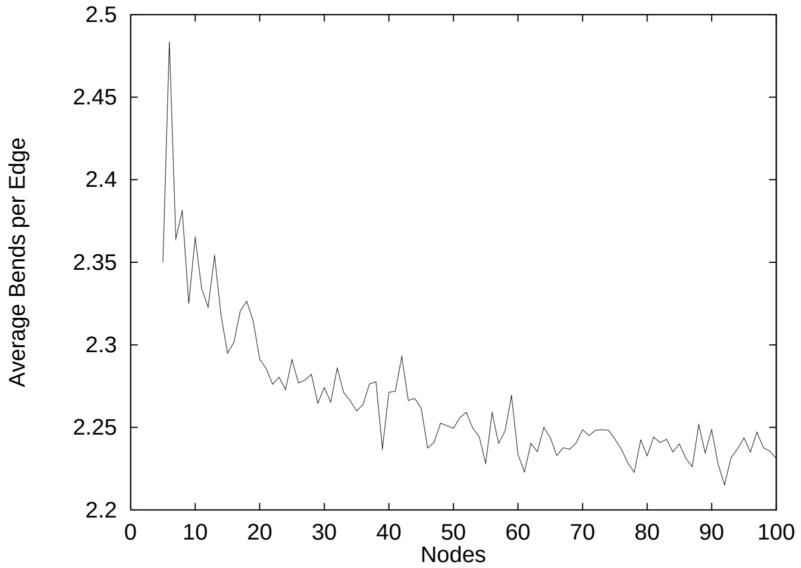
- the volume is on the average $1.5n^3 + O(n^2)$,
- for medium and large size graphs the average number of bends per edge is about 2.25, and
- as expected, the average edge length grows linearly with respect to the number of nodes.

The second algorithm introduces 3-D orthogonal drawing for simple graphs of degree higher than 6. Vertices are represented by solid 3-D boxes whose surface is proportional to the degree of the vertex. The number of bends per edge is only two.

Improving our volume upper bounds is a natural open problem. Although there have been better volume upper bounds [13] at the expense of allowing more bends per edge (seven bends per edge are allowed in [13]), cubic upper bounds are the best known so far when we allow at most three bends per edge. It is clear that there is a trade-off between volume and number of bends per edge, or between volume and edge crossings. Determining the trade-offs is a



(a)



(b)

Figure 12: Graphs of the experimental findings: (a) volume; (b) bends.

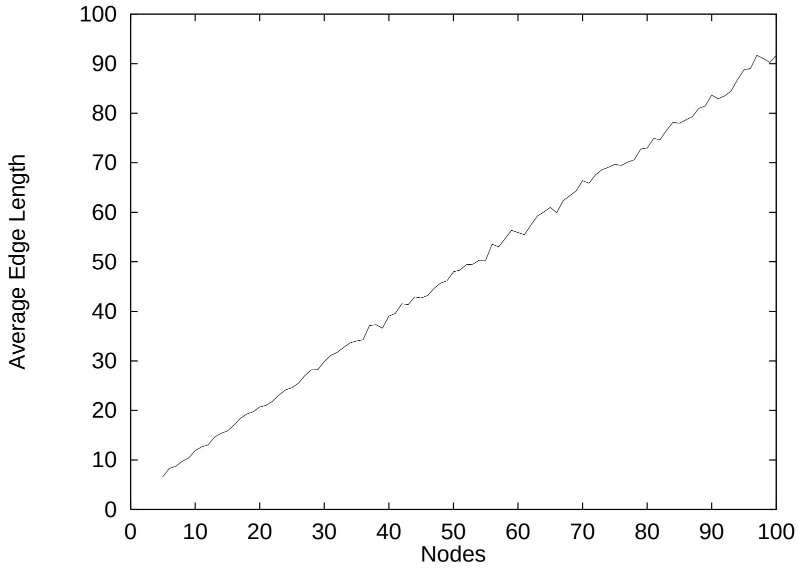


Figure 13: Graphs of the experimental findings: edge length.

very interesting problem. Finally, if one restricts his/her attention to drawings of graphs that are known ahead of time, is there a numbering of the vertices that guarantees better performance with respect to volume and/or other aesthetic measures?

Acknowledgements

We would like to thank Maurizio "Titto" Patrignani for sharing with us the valuable experimental results of his implementation of our algorithm; also, we would like to thank Janet Six for useful discussions, and the referees for their thorough reading of the paper and their useful comments that improved the presentation of the algorithms.

References

[1] H. Alt, M. Godau, S. Whitesides, *Universal 3-Dimensional Visibility Representations for Graphs*, Comput. Geom. Theory Appl., 9:111-125, 1998.

[2] 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.

[3] P. Bose, H. Everett, S. Fekete, M. Houle, A. Lubiw, H. Meijer, K. Romanik, G. Rote, T. Shermer, S. Whitesides, and C. Zelle. *A visibility representation for graphs in three dimensions*, J. Graph Algorithms Appl., 2:1-16, 1998.

[4] M. Brown and M. Najork, *Algorithm animation using 3D interactive graphics*, Proc. ACM Symp. on User Interface Software and Technology, 1993, pp. 93-100.

[5] 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.

[6] M. Chrobak, M. Goodrich, and R. Tamassia, *Convex drawings of graphs in two and three dimensions*, Proc. 12th Annual ACM Symposium on Computational Geometry, 1996, pp. 319-328.

[7] R. Cohen, P. Eades, T. Lin, F. Ruskey, *Three Dimensional Graph Drawing*, Algorithmica, 17:199-208, 1997.

[8] 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.

[9] P. F. Dietz and D. D. Sleator, *Two algorithms for maintaining order in a list*, Proc. 19th Annual ACM Symp. Theory of Computing, 1987, pp. 365-372.

[10] G. Di Battista, P. Eades, R. Tamassia and I. G. Tollis, *Graph Drawing*, Prentice Hall, 1999.

[11] D. Dodson, *A Tool for Information Visualization using Co-operative 3D Diagram Layout*, Proc. of Workshop GD '95, Lecture Notes in Comp. Sci. 1027, Springer-Verlag, 1995, pp. 190-201.

[12] P. Eades, C. Stirk, S. Whitesides, *The Techniques of Kolmogorov and Bardzin for Three Dimensional Orthogonal Graph Drawings*, Information Processing Letters, 60:97-103, 1996.

[13] P. Eades, A. Symvonis, S. Whitesides, *Two Algorithms for Three Dimensional Orthogonal Graph Drawing*, Proc. of GD '96, Lecture Notes in Comp. Sci. 1190, Springer-Verlag, 1996, pp. 139-154.

[14] S. Even, *Graph Algorithms*, Computer Science Press, 1979.

[15] S. Even and G. Granot, *Grid Layouts of Block Diagrams — Bounding the Number of Bends in Each Connection* Proc. DIMACS Workshop GD '94, Lecture Notes in Comp. Sci. 894, Springer-Verlag, 1994, pp. 64-75.

[16] S. Fekete, M. Houle, S. Whitesides, *New Results on a Visibility Representation of Graphs in 3D*, Proc. of Workshop GD '95, Lecture Notes in Comp. Sci. 1027, Springer-Verlag, 1995, pp. 234-241.

[17] A. Garg and R. Tamassia, *GIOTTO3D: A System for Visualizing Hierarchical Structures in 3D*, Proc. of GD '96, Lecture Notes in Comp. Sci. 1190, Springer-Verlag, 1996, pp. 193-200.

[18] A. Garg, R. Tamassia, and P. Vocca, *Drawing with colors*, Proc. 4th Annual European Symposium on Algorithms (ESA '96), Lecture Notes Comp. Sci. 1136, Springer-Verlag, 1996, pp. 12-26.

[19] T. Kamada and S. Kawai, *An algorithm for drawing general undirected graphs*, Inf. Proc. Letters 31, (1989), 7-15.

[20] G. Kant, *Drawing Planar Graphs Using the Canonical Ordering*, Algorithmica, vol. 16, no. 1, 1996, pp. 4-32.

[21] A. N. Kolmogorov and Y. M. Bardzin, *About Realization of Sets in 3-dimensional Space*, Problems in Cybernetics, 1967, pp. 261-268.

[22] 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.

[23] A. Papakostas and I. G. Tollis, *Improved Algorithms and Bounds for Orthogonal Drawings*, Proc. DIMACS Workshop GD '94, Lecture Notes in Comp. Sci. 894, Springer-Verlag, 1994, pp. 40-51.

[24] A. Papakostas and I. G. Tollis, *Algorithms for Area-Efficient Orthogonal Drawings*, Computational Geometry Theory and Applications, 9 (1998), 83-110.

[25] 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.

[26] A. Papakostas and I. G. Tollis, *Interactive Orthogonal Graph Drawing*, IEEE Transactions on Computers, vol. 47, no. 11 November 1998, pp. 1297-1309.

[27] A. Papakostas and I. G. Tollis, *A Pairing Technique for Area-Efficient Orthogonal Drawings*, Proc. of GD '96, Lecture Notes in Comp. Sci. 1190, Springer-Verlag, 1996, pp. 355-370.

[28] A. Papakostas, J. Six, and I. G. Tollis, *Experimental and Theoretical Results in Interactive Graph Drawing*, Proc. of GD '96, Lecture Notes in Comp. Sci. 1190, Springer-Verlag, 1996, pp. 371-386.

[29] M. Patrignani and F. Vargiu, *3DCube: A Tool for Three Dimensional Graph Drawing*, Proc. of GD '97, Lecture Notes in Comp. Sci. 1353, Springer-Verlag, 1997, pp. 284-290.

[30] M. Patrignani, Personal communication, March 1998.

[31] P. Reid, *Dynamic interactive display of complex data structures*, in Graphic Tools for Software Engineers, Cambridge 1989.

[32] S. Reiss, *An engine for the 3D visualization of program information*, J. Visual Languages and Computing, vol. 6, no. 3, 1995.

[33] M. Schäffter, *Drawing Graphs on Rectangular Grids*, Discr. Appl. Math. 63 (1995) pp. 75-89.

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

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

[36] R. Tamassia and I. G. Tollis, *Planar Grid Embeddings in Linear Time*, IEEE Transactions on Circuits and Systems CAS-36 (1989), pp. 1230-1234.

[37] O. Tversky, S. Snibbe, R. Zeleznik, *Cone Trees in the UGA graphics system: Suggestions of a more robust visualization tool*, Technical Report CS-93-07, Brown University.

[38] C. Ware, D. Hui, G. Frank, *Visualizing object oriented software in 3 dimensions*, in Proc. CASCON, 1993.