ELSEVIER

Discrete Applied Mathematics 103 (2000) 55–87

DISCRETE APPLIED MATHEMATICS

Three-dimensional orthogonal graph drawing algorithms ☆

Peter Eades[a,1], Antonios Symvonis[b,*,2], Sue Whitesides[c,3]

[a] *Department of Computer Science and Software Engineering, University of Newcastle, Newcastle, Australia*

[b] *Basser Department of Computer Science, University of Sydney, Sydney, Australia*

[c] *School of Computer Science, McGill University, Montreal, Canada*

Received 29 January 1998; revised 10 November 1999; accepted 13 December 1999

Abstract

We use basic results from graph theory to design algorithms for constructing three-dimensional, intersection-free orthogonal grid drawings of n vertex graphs of maximum degree 6. The best previous result generated a drawing bounded by an $O(\sqrt{n}) \times O(\sqrt{n}) \times O(\sqrt{n})$ box, with each edge route containing up to 16 bends. Our algorithms initiate the study of bend/bounding box trade-off issues for three-dimensional grid drawings and produce drawings with the following characteristics:

1. at most 7 bends per edge route, bounded by an $O(\sqrt{n}) \times O(\sqrt{n}) \times O(\sqrt{n})$ box;
2. at most 6 bends per edge route, bounded by an $O(\sqrt{n}) \times O(\sqrt{n}) \times O(n)$ box;
3. at most 5 bends per edge route, bounded by an $O(\sqrt{n}) \times O(n) \times O(n)$ box;
4. at most 3 bends per edge route, bounded by an $O(n) \times O(n) \times O(n)$ box; and
5. for maximum degree 4 graphs, at most 3 bends per edge route, bounded by and $O(n) \times O(n) \times O(1)$ box. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Graph drawing; Graph theory; Layout; Graph colouring

1. Introduction

The three-dimensional orthogonal grid consists of *grid points* whose coordinates are all integers, together with the axis-parallel *grid lines* determined by these points. A *three-dimensional orthogonal grid drawing* of a graph G (possibly having loops and

☆ Partially written while the third author was visiting the University of Newcastle. Many of the results of this paper were announced in [11].

* Corresponding author.

E-mail address: symvonis@cs.su.oz.au (A. Symvonis).

[1] Supported in part by an Australian Research Council grant.

[2] Supported in part by an Australian Research Council grant.

[3] Supported in part by Quebec FCAR grant.

0166-218X/00/$ - see front matter © 2000 Elsevier Science B.V. All rights reserved.
PII: S0166-218X(00)00172-4

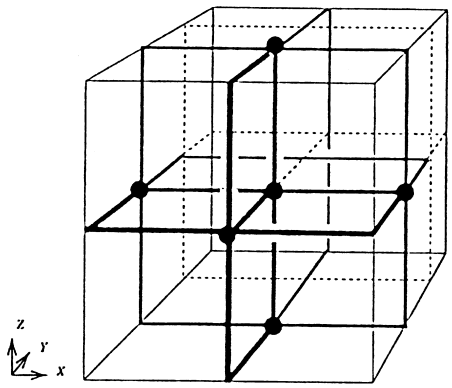

Fig. 1. An orthogonal grid drawing of a graph on 6 vertices.

multiple edges) places the vertices of G at grid points and routes the edges of G along sequences of contiguous segments contained in the grid lines. Edge routes are allowed to contain bends but are not allowed to cross or to overlap, that is, no internal point of one edge route may lie in any other edge route. Throughout this paper, *grid* refers to the three-dimensional orthogonal grid, and *grid drawing* refers to the type of three-dimensional orthogonal grid drawing just described. The $+X$ *port* of a grid point (x, y, z) is the closed undirected unit line segment joining (x, y, z) to $(x+1, y, z)$; ports for the other 5 signed grid directions are defined analogously. Note that because each grid point has exactly six ports, any graph that admits a grid drawing necessarily has maximum vertex degree at most 6.

Throughout this paper, we measure extent in each dimension in terms of the *number of grid points* rather than in terms of length. For example, Fig. 1 shows a grid drawing of K_6, the clique on 6 vertices. This particular drawing lies in a $3 \times 4 \times 3$ bounding box. The length of an edge route is counted in "units" of length. The edge route joining the two extremal vertices in the Z-direction lies along the top, back and bottom faces of the box, contains 2 bends and has length 6. The edge route joining the two extremal vertices in the X-direction also contains 2 bends, but passes through the interior of the box, and has length 4.

While the graph drawing literature has extensively investigated two-dimensional grid drawings of graphs (see [8]), three-dimensional grid drawing has been little studied. Our research is motivated in part by recent interest in exploring the utility of three-dimensional drawings of graphs for visualisation purposes. It should also be noted that since VLSI technology now permits the stacking of many layers, this work may be relevant to that application area as well. Furthermore, future microfabrication technologies other than VLSI may be potential areas of application for these results.

This paper offers several algorithms for obtaining grid drawings of graphs of maximum degree 6, possibly having loops and multiple edges. All algorithms use basic graph theory to preprocess the input graph by colouring its edges; then the algorithms

route edges according to their colour class. One of our algorithms produces drawings with a small number of bends per edge, while the other algorithms produce more compact drawings, but at the cost of an increased number of bends per edge. This raises the issue of a trade-off between bends and bounding box size. Our algorithmic results establish upper bounds for the number of bends per route on the one hand, and for various measures of the bounding box on the other hand.

Since there are many measures of bounding box compactness (for example, volume, maximum dimension, sum of dimensions, length of long diagonal) we simply give the dimensions, measured in terms of grid points, of the bounding box of the drawings produced by our algorithms, we do not define compactness precisely. Note that volume is generally not the most appropriate measure of bounding box suitability for three-dimensional drawings for visualisation purposes.

Our first algorithm takes as input any n vertex graph of maximum degree at most 6 and produces a grid drawing bounded by a box of dimensions $O(\sqrt{n}) \times O(\sqrt{n}) \times O(\sqrt{n})$. Each edge route has length $O(\sqrt{n})$ and contains at most 7 bends. This improves the best previously known result [10] of 16 bends maximum per edge route, described further below. Kolmogorov and Barzdin ([19]; see also [10]) showed that no algorithm can produce asymptotically more compact drawings; hence we refer to our first algorithm as the *compact-drawing* algorithm.

By successive refinements of the compact-drawing algorithm, we explore trade-offs between the number of bends of each edge route and the size of the bounding box of the drawing. More specifically, we provide drawings for n-vertex graphs of maximum degree at most 6 with the following characteristics:

1. at most 6 bends per edge route, using a bounding box of dimensions $O(\sqrt{n}) \times O(\sqrt{n}) \times O(n)$;
2. at most 5 bends per edge route, using a bounding box of dimensions $O(\sqrt{n}) \times O(n) \times O(n)$; and
3. at most 4 bends per edge route, using a bounding box of dimensions $O(n) \times O(n) \times O(n)$.

Further, using a different and very simple technique, we show that each graph of maximum degree 6 has a three-dimensional orthogonal drawing with at most 3 bends per edge, bounded by a box of dimensions $O(n) \times O(n) \times O(n)$. Recently, a complex constructions of Papakostas and Tollis [22] has been used to produce orthogonal drawings with at most 3 bends per edge, bounded by a box of $O(n^3)$ volume, but with an improved constant.

Also, for graphs of maximum degree at most 4 we produce an algorithm which draws the graph in a box of dimensions $O(n) \times O(n) \times O(1)$ with 3 bends per edge route.

Note that the results of this paper are limited to graphs of maximum degree 6. This is because we use a point to represent a vertex. Rectangles and other geometric objects may be used to draw graphs of maximum degree greater than six in three dimensions; see, for example, [6,22].

For two dimensions, there are many methods for producing compact orthogonal grid drawings with few bends for graphs of maximum degree 4. Several methods for

obtaining drawings of planar and nonplanar graphs in area $O(n) \times O(n)$ with $O(1)$ bends per edge are available; see, for example, [3–5,13,18,21,27,29,31–33].

Not surprisingly, problems that are seemingly computationally intractable arise in both two-dimensional and three-dimensional grid drawing. For example, in two dimensions, minimising the number of bends is NP-complete [15], but can be solved in polynomial time for any fixed planar embedding [30]. Eades et al. [10] have shown how to generalise various two-dimensional NP-completeness results such as minimising the number of bends, the volume of the drawing, and the maximum individual edge route length (see [2,9,14,15,20]) to three dimensions.

Reference [10] provides an algorithm, based on the technique of Kolmogorov and Barzdin [19], for obtaining three-dimensional orthogonal grid drawings. The algorithm takes as input an n vertex graph of maximum degree at most 6 and produces a drawing in which each edge route has atmost 16 bends and has $O(\sqrt{n})$ length. The drawing lies in an $O(\sqrt{n}) \times O(\sqrt{n}) \times O(\sqrt{n})$ bounding box. Thus the compact-drawing algorithm we present here reduces the number of bends per edge route to a maximum of 7 while still achieving the bounding box dimensions and maximum edge route length obtained by [10].

Biedl (private communication) has shown that drawings with similar bounds on the edge length and bounding box dimensions can be obtained by using the techniques of three-dimensional VLSI layout from the early 1980's [24–26].

Wood [34] studied orthogonal graph drawings in higher dimensions. He showed how to draw a graph of maximum degree $d > 5$ with at most 5 bends per edge route, bounded by a $\lceil d/2 \rceil$-hypercube of side length $O(n)$. Wood also showed that K_7, the clique on 7 vertices, has a grid drawing with at most 2 bends per edge route; we had originally conjectured that K_7 requires at least 3 bends on some edge route. The question of whether every graph of maximum degree 6 has an orthogonal grid drawing with at most 2 bends per edge remains open.

The rest of this paper is organised as follows. Section 2 gives the simple graph-theoretic methods that our algorithms use to preprocess the input graph. Section 3 presents and analyses the compact-drawing algorithm, while Section 4 presents successive refinements of the compact-drawing algorithm that illustrate bend/bounding box trade-offs. Section 5 presents and analyses the 3-bends algorithm. Section 6 describes the 3-bend drawing algorithm for graphs of maximum degree 4. We conclude in Section 7.

2. Preliminaries

This section gives the preprocessing step that all of our drawing algorithms employ. First we recall a definition from graph theory.

Definition 1. A *cycle cover* of a directed graph is a spanning subgraph that consists of directed cycles.

The following theorem combines a classical theorem of graph theory due to Petersen [23] stating that "a regular multigraph of degree $2k$ has k edge-disjoint factors" (see also, [1, p. 227]) with result of Schrijver [28]. We present the proof in algorithmic form for completeness, since we use this algorithm throughout the paper.

Theorem 1. *Suppose that $G = (V, E)$ is an undirected graph of maximum degree Δ and let $d = [\Delta/2]$. Then there is a directed graph $G' = (V, E')$ such that*
- *each vertex of G' has indegree d and outdegree d;*
- *G is a subgraph of the underlying undirected graph of G'; and*
- *the edges of G' can be partitioned into d edge disjoint cycle covers.*

Furthermore, for an n-vertex graph G, the directed graph G' and its d cycle covers can be computed in $O(\Delta^2 n)$ time.

Proof. Pair the odd degree vertices of G and add a new edge for each pair. This can be done since the number of vertices of odd degree in any graph is even. After the addition of these new edges, each vertex has even degree at most $2d$. Add k self-loops (v, v) to each $v \in V$ of degree $2(d - k)$ to create a regular multigraph (with self-loops and multiple edges allowed) of degree $2d$. This graph is Eulerian, since each of its vertices has even degree. Direct its edges by following an Eulerian circuit to obtain a directed graph G' with indegree d and outdegree d at each vertex. Clearly these operations can be performed in $O(\Delta n)$ time.

From G', construct an undirected bipartite graph $G' = (V_{\text{out}} \cup V_{\text{in}}, E'')$ with $V_{\text{out}} = \{v_{\text{out}} \mid v \in V\}, V_{\text{in}} = \{v_{\text{in}} \mid v \in V\}$, and $E'' = \{(u_{\text{out}}, v_{\text{in}}) \mid (u, v) \in E'\}$. Note that G'' is d-regular and bipartite. By Hall's Theorem [16,12], G'' contains a perfect matching; colour the edges of the matching with *colour*-1 and remove them. The remaining graph is bipartite and $(d-1)$-regular, so it again contains a perfect matching. Colour its edges with *colour*-2 and remove them. Continuing in a similar fashion, colour the remaining edges of G'' by using $d - 2$ additional colours.

Now colour each directed edge (u, v) of G' with the colour given to $(u_{\text{out}}, v_{\text{in}})$ of G''. This gives each vertex of G' exactly one incoming and one outgoing edge of each of the d colours. Hence the edges of G' are partitioned into d coloured subgraphs $C_1, C_2, \ldots, C_d$ each of which is a cycle cover for G'.

Since a maximum matching in an arbitrary bipartite graph with n vertices and maximum degree Δ can be computed in $O(\Delta^2 n)$ [28], the computation of the $d = [\Delta/2]$ cycle covers can be done in $O(\Delta^3 n)$ time. A simple observation can further reduce the time complexity to $O(\Delta^2 n)$. The procedure outlined above applies the maximum matching algorithm for arbitrary bipartite graphs and effectively results to a Δ-edge colouring of the bipartite graph. In fact, a Δ-edge colouring is all we need and it can be directly computed for a Δ-regular bipartite graph in $O(\Delta^2 n)$ time [28].

Fig. 2 shows the process of partitioning the edges of a 4-regular undirected graph into two cycle covers, C_{red} and C_{green}.

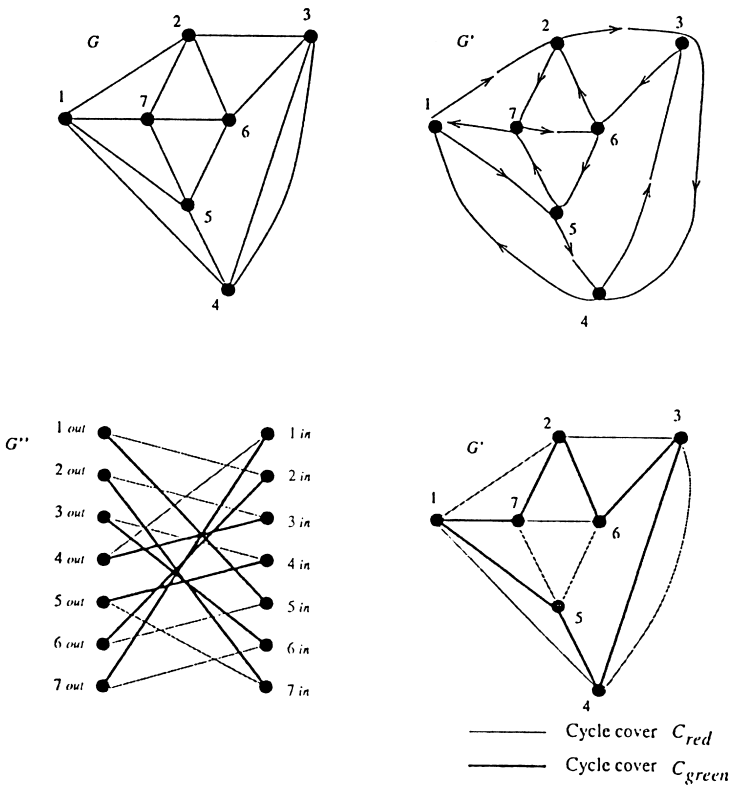

Fig. 2. A decomposition of a 4-regular undirected graph into 2 cycle covers.

Preprocessing algorithm. We call the algorithm described in the proof of Theorem 1 the *preprocessing algorithm*. We concentrate on graphs of maximum degree 6 since they are the only graphs that admit a three-dimensional orthogonal grid drawing. To summarize, the preprocessing algorithm takes as input an undirected graph G of maximum degree 6 and computes as output a directed graph G' whose underlying undirected graph contains G; the preprocessing algorithm also computes a partition of the edges of G' into three edge disjoint cycle covers, denoted $C_{\text{red}}, C_{\text{green}}$ and C_{blue}. For the case where the maximum degree of G is 4, the edges of G' are partitioned into two edge disjoint cycle covers, denoted C_{red} and C_{green}. Since $\Delta \leqslant 6$, the algorithm runs in $O(n)$ time.

All of our drawing algorithms specify edge routes for the cycle covers of G'. To obtain a drawing for G, the algorithms route the undirected edges of G according to the routes for the corresponding directed edges of G'. Loops and edges of G' that do not arise from loops and edges of G are simply not drawn.

Remark. In the following sections, it is helpful to keep in mind that each vertex of G' has exactly one incoming and exactly one outgoing edge (not distinct in the case

of loops) of each colour. This has an important consequence for multiple edges. Two vertices a and b in G' may have several edges between them. However, for any specific colour c, there are at most two edges of colour c between a and b; if there are two such edges, then they are directed oppositely.

3. The compact-drawing algorithm

This section describes our compact-drawing algorithm, which takes as input a graph $G=(V,E)$ of maximum degree at most 6 and produces as output a grid drawing for G having at most 7 bends per edge, maximum edge length $16\lceil\sqrt{n}\rceil - 10$, and bounding box dimensions $(3\lceil\sqrt{n}\rceil + 2) \times 5\lceil\sqrt{n}\rceil \times (8\lceil\sqrt{n}\rceil - 3)$, where $|V| = n$.

The compact-drawing algorithm

1. Run the preprocessing algorithm of Section 2 to construct the directed graph G' and to obtain a partition of its edges into three edge disjoint cycle covers, denoted $C_{\text{red}}, C_{\text{blue}}$, and C_{green}.
2. Use cycle cover C_{red} to place the vertices on the plane $Z=0$; design routes for the edges in C_{red} that do not leave this plane and that have at most 7 bends per route. Draw those routes arising from edges of G.
3. Design routes for the edges in cycle cover C_{blue} that lie on and above plane $Z=0$ and that have at most 7 bends per route. Draw those routes arising from edges of G.
4. Design routes for the edges in cycle cover C_{green} that lie on and below plane $Z=0$ and that have at most 7 bends per route. Draw those routes arising from edges of G.

The details of the first step are described in the previous section. Steps 2–4 are described in the next few subsections.

3.1. Routing the red cycle cover

Assume that cycle cover C_{red} consists of k directed cycles $c_1, c_2, \ldots, c_k$, and use these cycles to order the vertices of G' as follows. Arbitrarily choose a starting vertex from c_1, and order the remaining vertices of c_1 by following the cycle; then order the vertices of the remaining cycles in similar fashion, ordering the vertices of c_i before those of c_j if $i < j$.

Next, define a square array of *special* grid points $p_{i,j}$ in the plane $Z=0$ as follows. For $0 \leqslant i,\ j < \lceil\sqrt{n}\rceil$, $p_{i,j} = (5i+3, 5j+3, 0)$. The number of columns here is chosen for convenience of description, and an improvement is possible, as we will see later.

Rather than give explicit formulas for vertex placement and edge routing we illustrate this in the context of a specific example from which the reader can easily infer the general rules. Suppose that a graph G' has the following cycle cover C_{red}: $\langle v_1, v_2, v_3\rangle, \langle v_4, \ldots, v_9\rangle \langle v_{10}, \ldots, v_{21}\rangle, \langle v_{22}, v_{23}, v_{24}\rangle$, and $\langle v_{25}\rangle$. Using the order just obtained from cycle cover C_{red}, assign the vertices of G' to grid points $p_{i,j}$ in the

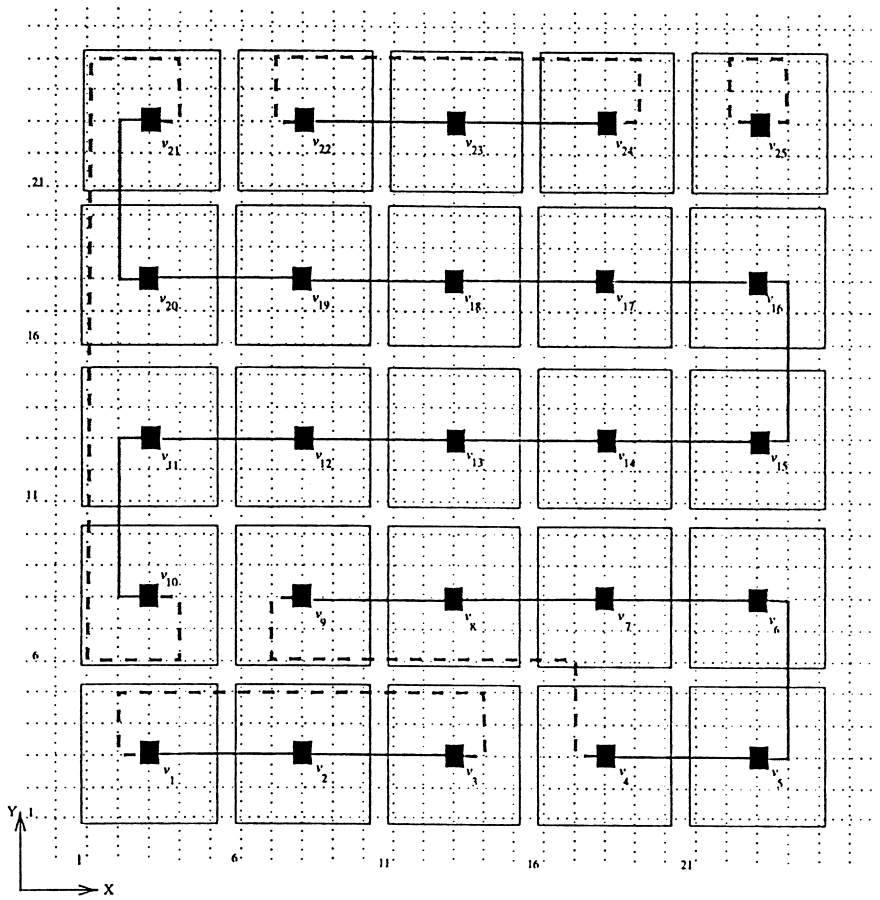

Fig. 3. The layout of the vertices of G and the routing of edges in cycle cover C_{red}.

snake-like fashion illustrated in Fig. 3, omitting any edges that did not arise from the original graph G. Then route the edges of the cycles as shown in Fig. 3. In particular, Fig. 3 shows how to handle cycles whose vertices lie within one row of special grid points, cycles whose vertices lie in parts of two neighbouring rows, and cycles whose vertices occupy more that one row.

Denote $\{(5i+k, 5j+l, 0) | 1 \leqslant k, l \leqslant 5\}$ by $Sq(i, j)$; this is a set of grid points within a square centred at $p_{i,j}$. The squares themselves form a square array, so we speak of the *squares of row i* (rows are parallel to the X-axis) and the *squares of column j* (columns are parallel to the Y-axis).

Now we make two observations for future reference. The first one will be used in proving that no two edge routes intersect while the second will be used in reducing the volume of the drawing (by a constant factor).

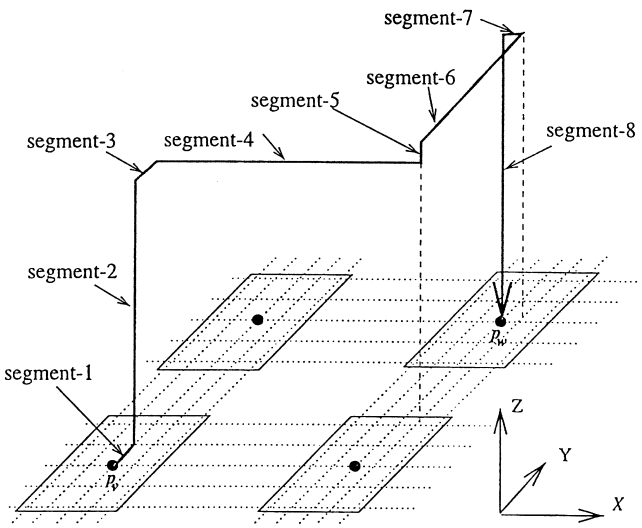

Fig. 4. The 7 bend route for edge (v, w) of C_{blue}.

Observation 1. The routes for red edges satisfy the following properties:
1. The edge routes for edges in C_{red} have at most 6 bends per route.
2. The red routes do not use the $-Y$ and $+Y$ ports of any special grid point $p_{i,j}$.
3. The grid points contained in routes connecting vertices in the same cycle of C_{red} are entirely contained in the squares to which those vertices are assigned.

Observation 2. Consider the line segments parallel to the Y-axis that are contained in routes for red edges. In the first (last) column of squares, these segments contain no grid points in the fifth (first) column of grid points of the column of squares. In all other columns of squares, these segments contain no grid points in the first and fifth column of grid points of each column of squares.

3.2. Routing the blue and green cycles covers

We describe how to route the edges in cycle cover C_{blue}. The edges of C_{green} are routed similarly, but on the other side of the plane $Z = 0$.

Suppose that vertices v and w are assigned to special grid points having coordinates $p_v = (x_v, y_v, 0)$ and $p_w = (x_w, y_w, 0)$, respectively. Then the route for edge (v, w) of C_{blue} is illustrated in Fig. 4 and consists of the 8 segments described in Table 1. Note that the horizontal line segments are routed in the planes $Z = z_{vw}$ and $Z = z_{vw} + 1$. Here we defer until later the specification of the value z_{vw}, noting for the moment that for all v, w, the value of z_{vw} will be a positive *odd* integer.

Table 1
The route coordinates for edge (v,w) of C_{blue}

Segment	Start point	→	Finish point
1	$(x_v, y_v, 0)$	→	$(x_v, y_v+1, 0)$
2	$(x_v, y_v+1, 0)$	→	(x_v, y_v+1, z_{vw})
3	(x_v, y_v+1, z_{vw})	→	(x_v, y_v+2, z_{vw})
4	(x_v, y_v+2, z_{vw})	→	(x_w+1, y_v+2, z_{vw})
5	(x_w+1, y_v+2, z_{vw})	→	$(x_w+1, y_v+2, z_{vw}+1)$
6	$(x_w+1, y_v+2, z_{vw}+1)$	→	$(x_w+1, y_w, z_{vw}+1)$
7	$(x_w+1, y_w, z_{vw}+1)$	→	$(x_w, y_w, z_{vw}+1)$
8	$(x_w, y_w, z_{vw}+1)$	→	$(x_w, y_w, 0)$

3.3. Proof of correctness

Using techniques from [10], we prove that the routes of any two edges do not intersect (except at mutually incident vertices). Note that if a unit length segment contains an intersection point not located at a special grid point $p_{i,j}$, then one of its adjacent segments (of length greater than 1) also contains this intersection point. Thus we need only consider the possibility of intersections of segments longer than one unit of length, i.e. intersections among even-numbered segments.

Observe that the routing of segment-4 for every edge route takes place in plane $Y = 5j + 5,\ 0 \leqslant j < \lceil \sqrt{n} \rceil$, and that the routing of segment-6 for every edge route takes place in plane $X = 5i + 4,\ 0 \leqslant i < \lceil \sqrt{n} \rceil$. This implies that these segments cannot intersect with any segment numbered 2 or 8 from any edge route. Also, note that segment-2 obviously cannot intersect segment-8.

Further observe that it is not possible for any segment-4 to intersect any segment-6. This is because the two segments are routed on parallel planes, one with even and the other with odd Z-coordinates. Hence, the only possible intersections are between segment-4 of one route and segment-4 of another, or between segment-6 of one route and segment-6 of another.

Finally we explain how to choose the values for z_{vw}. This is done by using the method of [10]. Consider edge (v,w). From the preceding paragraph, the route for this edge can intersect only with the edge routes with origin in the row of squares in which vertex v is placed and the edge routes with destination in the same column of squares in which vertex w is placed.

We construct a graph H whose vertex set is the edge set of cycle cover C_{blue}. An edge is inserted between two vertices in the graph H whenever the vertices correspond to edges with start vertices in the same row or end vertices in the same column. The graph H has maximum degree $2(\lceil \sqrt{n} \rceil - 1)$ and thus it has a vertex colouring with $2\lceil \sqrt{n} \rceil - 1$ colours, which can be obtained by a greedy algorithm [7, Brook's Theorem]. This algorithm takes time linear in the size of H, that is, $O(n^{3/2})$ time in terms of the number n of vertices of G. Suppose that colour $c, 1 \leqslant c \leqslant 2\lceil \sqrt{n} \rceil - 1$, has been assigned to the vertex of H that corresponds to blue edge (v,w). Then we set $z_{vw} = 2c - 1$.

Now we are ready to state the main result of this section.

Theorem 2. *Every n vertex maximum degree* 6 *graph G has a three-dimensional orthogonal grid drawing with the following characteristics*:

(i) *at most* 7 *bends per edge route*,
(ii) $16\lceil\sqrt{n}\rceil - 10$ *maximum edge length*, and
(iii) *a bounding box of dimensions* $(3\lceil\sqrt{n}\rceil + 2) \times 5\lceil\sqrt{n}\rceil \times (8\lceil\sqrt{n}\rceil - 3)$.

Moreover, the drawing can be obtained in $O(n^{3/2})$ *time*.

Proof. Green edge routes begin with a $-Y$ port. Except for such ports, green edge routes lie below the plane $Z=0$ and obviously do not intersect blue edge routes, which begin with a $+Y$ port and otherwise lie above the plane $Z=0$. Based on Observation 1 and the fact that the red edge routes lie on the plane $Z=0$ (using different ports from either the green or the blue edge routes), we conclude that red edge routes do not intersect blue or green edge routes. As discussed previously, no two edges of the same colour intersect. Thus the drawing obtained by routing edges of a graph G according to the routes for their corresponding directed edges in G' gives a proper grid drawing.

The fact that there are at most 7 bends per edge route can be seen by inspection. We now determine the dimensions of the axis-aligned bounding box. Its base is a square of dimensions $5\lceil\sqrt{n}\rceil \times 5\lceil\sqrt{n}\rceil$ due to the placement of the vertices and the routing of the edges of C_{red}. The height of the bounding box is at most $8\lceil\sqrt{n}\rceil - 3$. To see this, recall that there are at most $2\lceil\sqrt{n}\rceil - 1$ different z_{vw} values assigned to the blue edges, and that two adjacent planes determined by each z_{vw} values are used for the routing of each edge. Thus for the routing of the blue edges, we use at most $2(2\lceil\sqrt{n}\rceil-1)=4\lceil\sqrt{n}\rceil-2$ planes parallel to the XY-plane. The same number of planes is required for the routing of the green edges, leading to a bounding box of height $8\lceil\sqrt{n}\rceil - 3$ (remembering to count plane $Z=0$).

The maximum possible edge route length corresponds either to a blue edge route starting from $p_{0,0}=(3,3,0)$ and leading to $p_{\lceil\sqrt{n}\rceil-1,\lceil\sqrt{n}\rceil-1}=(5(\lceil\sqrt{n}\rceil-1)+3, 5(\lceil\sqrt{n}\rceil - 1) + 3, 0)$ through the top of the box, or to its green counterpart directed in the opposite direction. The length of such a route is at most $(18\lceil\sqrt{n}\rceil - 12)$ units. To see this, observe that it takes $8\lceil\sqrt{n}\rceil-4$ units of length of climb to the top of the bounding box and then return to the plane $Z=0$, and that the length of the projection of the path from $p_{0,0}$ to $p_{\lceil\sqrt{n}\rceil-1,\lceil\sqrt{n}\rceil-1}$ on the plane $Z=0$ is at most $2.5(\lceil\sqrt{n}\rceil - 1) + 2$, adding up to an edge length of at most $(18\lceil\sqrt{n}\rceil - 12)$ units.[4]

The maximum edge route length and the volume of the drawing can be improved by a constant factor. Based on Observation 2, each 5×5 square except those in the first and last column of squares can be replaced by a 3×5 rectangle, while the squares in the first and last column of squares can be replaced by a 4×5 rectangle. This change to the X-coordinates of the special grid points does not affect the general formulas for

[4] The term "+2" in the expression $2.5(\lceil\sqrt{n}\rceil - 1) + 2$ corresponds to the overshoot by one unit of length of point $p_{\lceil\sqrt{n}\rceil-1,\lceil\sqrt{n}\rceil-1}$ along the X-dimension, and its correction.

the blue and green edge routes. The modified drawing remains correct, and fits in a box of dimensions $3(\lceil\sqrt{n}\rceil+2)\times 5\lceil\sqrt{n}\rceil\times(8\lceil\sqrt{n}\rceil-3)$.

To compute the maximum edge length after the above modification, observe that in the condensed drawing the vertices previously positioned in squares $Sq(1,1)$ and $Sq(\lceil\sqrt{n}\rceil,\lceil\sqrt{n}\rceil)$ differ by $3(\lceil\sqrt{n}\rceil-1)$ in their X-coordinates and by $5(\lceil\sqrt{n}\rceil-1)$ in their Y-coordinates. Now add 2 units of length in order to cover the overshoot of the one end of the edge in the X-dimension, as well as the $8\lceil\sqrt{n}\rceil-4$ units of length required to climb to the top of the bounding box and then return to the plane $Z=0$; this yields a maximum edge length of $(16\lceil\sqrt{n}\rceil-10)$ units.

The time consuming part of the compact-drawing algorithm is the vertex colouring used in the routing of the blue and green edges. This can be implemented in linear time in the size of the conflict graph H, which is $O(n^{3/2})$ in terms of the number n of vertices of G. □

4. Bend/bounding box trade-offs

In this section we explore trade-offs between the number of bends per edge and the dimensions of the bounding box of the drawing. By successive refinements of the compact-drawing algorithm, we provide drawings with the following characteristics:

- 6 bends per edge route, using a bounding box of dimensions $O(\sqrt{n})\times O(\sqrt{n})\times O(n)$;
- 5 bends per edge route, using a bounding box of dimensions $O(\sqrt{n})\times O(n)$ $O(n)\times$; and
- 4 bends per edge route, using a bounding box of dimensions $O(n)\times O(n)\times O(n)$.

We choose to focus solely on asymptotic measures in these trade-offs, and we make no effort to minimize the dimensions of the bounding box with respect to the constants hidden in the big-Oh notation. Condensed drawing that improve the volume of the layout by a constant factor can be easily obtained based on an observation similar to Observation 2.

We refine the compact-drawing algorithm by eliminating the three unit-length segments, that is, segment-3, segment-5, segment-7, from the routes of the blue and green edges. For each segment that we eliminate, the number of bends reduces by one while the length of the bounding box side parallel to the eliminated segment increases by a factor of $O(\sqrt{n})$.

Recall from Observation 1 that in the drawing produced by the compact-drawing algorithm, each red edge has at most 6 bends. We next revise the routing of C_{red} so that each edge has a most 4 bends.

4.1. Routing the red cycle cover with at most 4 bends per edge route

The vertices of the graph are again placed on grid points at the centres of 5×5 squares. However, this time the layout consists of at most $\lceil 2\sqrt{n}\rceil$ rows of squares, with each row consisting of $\lceil\sqrt{n}\rceil$ squares.

Assume as before that cycle cover C_{red} consists of k directed cycles $c_1, c_2, \ldots, c_k$ but now also assume that the cycles are listed in decreasing order with respect to their size, that is, $|c_1| \geqslant |c_2| \geqslant \cdots \geqslant |c_k|$, where $|c_i|$ denotes the number of vertices of cycle $c_i, 1 \leqslant i \leqslant k$. Let $c_l, 1 \leqslant l \leqslant k-1$, be the cycle with the largest index that consists of more than $\lceil \sqrt{n} \rceil$ vertices, that is, $|c_l| > \lceil \sqrt{n} \rceil$ and $|c_{l+1}| \leqslant \lceil \sqrt{n} \rceil$. In the event that all cycles have more than $\lceil \sqrt{n} \rceil$ vertices, set $l = k$. If no cycle with more than $\lceil \sqrt{n} \rceil$ vertices exists, by definition set $l = 0$.

Assign the vertices of cycles $c_1, \ldots c_l$ to the grid points at the centres of the squares in a snake-like fashion, ensuring that the first vertex of each cycle is assigned to a grid point in the leftmost column of squares, with the cycle initially extending to the right. A different set of rows of squares is devoted to the layout of the vertices of the remaining cycles. The vertices of cycles $c_{l+1}, \ldots, c_k$ are assigned to rows of squares, extending from left to right, in a way similar to the famous *first-fit-decreasing bin-packing* algorithm [17]. We process the cycle in decreasing order of their lengths, and when we assign the vertices of a cycle to grid points at the centres of squares, we try to allocate them to squares to the right of a cycle whose vertices have already been assigned to grid points, provided that there are enough squares in the row to accommodate the whole cycle. If we are not able to find such a row of squares, then we simply use a new one and we place the cycle starting at the leftmost square and extending to the right. We refer to the assignment of the vertices to grid points just described as the *4-bend vertex placement*.

Fig. 5 shows the 4-bend vertex placement based on the cycle cover $c_1 = \langle v_{10}, \ldots, v_{21} \rangle$, $c_2 = \langle v_4, \ldots, v_9 \rangle, c_3 = \langle v_1, v_2, v_3 \rangle, c_4 = \langle v_{22}, v_{23}, v_{24} \rangle$, and $c_5 = \langle v_{25} \rangle$. This is the same cycle cover as in the example of Fig. 3, but the order of the cycle placement is different.

Fig. 5 also shows how to route the edges of each cycles. Cycles of size at most $\lceil \sqrt{n} \rceil$ lie entirely within one row of squares, and the routing of their edges is done in a way identical to the routing of the edges of cycles $\langle v_1, v_2, v_3 \rangle$, and $\langle v_{22}, v_{23}, v_{24} \rangle$, in Fig. 3, with at most 4 bends per edge route. For cycles of size greater than $\lceil \sqrt{n} \rceil$ we consider two cases. In the first case, vertices of the cycle occupy the leftmost squares of the last row of squares devoted to the cycle. The routing of the edges is performed as is shown for c_1 in Fig. 5 with at most 4 bends per edge. The second case corresponds to the situation where vertices of the cycle occupy the leftmost squares of the last row of squares devoted to the cycle. In this case, the routing of the edges is done as is shown for c_2 in Fig. 5; here there are less than 4 bends.

Lemma 1. *The* 4*-bend vertex placement of an n vertex degree-*6 *graph G requires at most* $\lfloor 2\sqrt{n} \rfloor$ *rows of squares.*

Proof. Let $c_1, c_2, \ldots, c_k$ denote the cycles of cycle over C_{red}, listed in decreasing order of their sizes. Let $c_l, 0 \leqslant l \leqslant k$, be the cycle with the largest index and size greater than $\lceil \sqrt{n} \rceil$, that is, $|c_l| > \lceil \sqrt{n} \rceil$ and $|c_{l+1}| \leqslant \lceil \sqrt{n} \rceil$; if no such cycle exists, by definition we set $l = 0$.

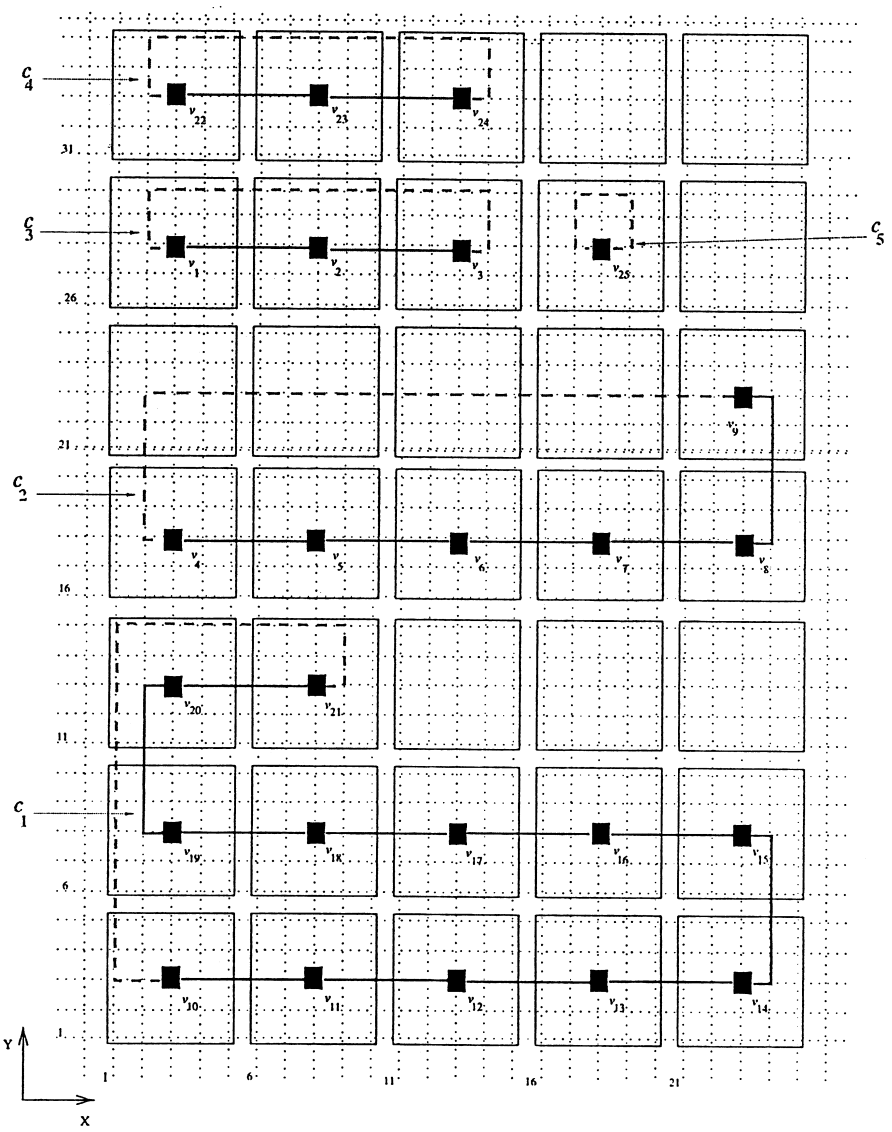

Fig. 5. The layout of the vertices of G and the routing of edges in cycle cover C_{red}.

Let N_1 denote the number of rows required for the placement of the vertices in cycles $c_1,\ldots,c_l$, and let N_2 denote the number of rows required for the placement of the vertices in cycles $c_{l+1},\ldots,c_k$.

Since $|c_i| \geqslant \lceil\sqrt{n}\rceil$, $1 \leqslant i \leqslant l$, we have that

$$\sum_{i=l+1}^{k} |c_i| \leqslant n - l\lceil\sqrt{n}\rceil. \tag{1}$$

Cycle c_i, $1 \leqslant i \leqslant l$, requires $\lceil |c_i| / \lceil \sqrt{n} \rceil \rceil$ rows of squares for the placement of its vertices. Thus

$$N_1 = \sum_{i=1}^{l} \left\lceil \frac{|c_i|}{\lceil \sqrt{n} \rceil} \right\rceil \leqslant \frac{\sum_{i=1}^{l} |c_i|}{\lceil \sqrt{n} \rceil} + l. \tag{2}$$

During the placement of the vertices of cycles $c_{l+1}, \ldots, c_k$ we introduce a new row of squares only if we cannot fit the cycle under consideration into one of the existing rows of squares. This implies that, at the end of the vertex placement, out of all rows devoted to cycles $c_{l+1}, \ldots, c_k$ only one row can have as many as $\lceil \lceil \sqrt{n} \rceil / 2 \rceil$ squares without any vertex assigned to them. Thus there are $N_2 - 1$ rows r such that

$$\lceil \sqrt{n} \rceil < 2 \sum_{c_i \text{ in row } r} |c_i|.$$

Hence it follows that

$$(N_2 - 1)\lceil \sqrt{n} \rceil < 2 \sum_{i=l+1}^{k} |c_i|,$$

and that

$$N_2 < \frac{2 \sum_{i=l+1}^{k} |c_i|}{\lceil \sqrt{n} \rceil} + 1. \tag{3}$$

Combining inequalities (2) and (3) (and by using (1)) we get

$$\begin{aligned} N_1 + N_2 &< \frac{\sum_{i=1}^{l} |c_i|}{\lceil \sqrt{n} \rceil} + l + \frac{2 \sum_{i=l+1}^{k} |c_i|}{\lceil \sqrt{n} \rceil} + 1 \\ &= \frac{n + \sum_{i=l+1}^{k} |c_i|}{\lceil \sqrt{n} \rceil} + l + 1 \\ &\leqslant \frac{n + (n - l\lceil \sqrt{n} \rceil)}{\lceil \sqrt{n} \rceil} + l + 1 \\ &= \frac{2n}{\lceil \sqrt{n} \rceil} + 1 \\ &\leqslant \lfloor 2\sqrt{n} \rfloor + 1. \end{aligned}$$

Thus, the total number of rows of squares is bounded by $\lfloor 2\sqrt{n} \rfloor$. Note that the bound in the lemma is tight. For example, if $n = (\lceil \sqrt{n} \rceil + 1)(\lceil \sqrt{n} \rceil - 1)$ and if each cycle consists of exactly $\lceil \sqrt{n} \rceil + 1$ vertices and there are $\lceil \sqrt{n} \rceil - 1$ cycles in C_{red}, then the placement requires 2 rows of squares for each cycle, for a total of $2\lceil \sqrt{n} \rceil - 2 = 2(\lfloor \sqrt{n} \rfloor + 1) - 2 = 2\lfloor \sqrt{n} \rfloor$ rows of squares. □

An inspection of the 4-bend vertex placement reveals that the properties stated in Observation 1 still hold; the only difference is that we now have at most 4 bends per edge route. Thus if we employ the 4-bend vertex placement of this section together with the routing of C_{blue} and C_{green} described in Section 3.2, then we obtain a drawing with at most 7 bends per edge route and a bounding box of dimensions $O(\sqrt{n}) \times O(\sqrt{n}) \times O(\sqrt{n})$.

The squares we used in the 4-bend vertex placement have dimensions 5×5. The reader interested in reducing the volume of the bounding box by a constant factor should note that an observation similar to Observation 2 holds, allowing the use of a 3×5 rectangle (instead of a 5×5 rectangle) in all columns of squares but the leftmost one, where a rectangle of dimensions 4×5 is required. The additional observation that the 4-bend vertex placement never uses the first row of each square (see Fig. 5) results in some additional savings in volume, placing (in the worst case) each vertex in a 4×4 square. Also note that the 4-bend vertex placement can be refined so that, whenever possible, the number of rows of squares it occupies is reduced by a constant. This is achieved by either (i) having two cycles, each of size greater than $\lceil \sqrt{n} \rceil$, share their non-full row of squares (if possible; this might require that one of the two cycles starts from the rightmost column and extends to the left), and/or (ii) allowing cycles of size smaller than or equal to $\lceil \sqrt{n} \rceil$ to be placed in the non-full row of squares partly occupied by a cycle of size greater than $\lceil \sqrt{n} \rceil$.

4.2. Refined routing for the blue and green cycle covers

In this section we eliminate, in turn, segment-3, segment-5 and segment-7 from the routes of the blue and green edges.

Theorem 3. *Every n vertex maximum degree 6 graph G has a three-dimensional orthogonal grid drawing with the following characteristics*:
(i) *at most 6 bends per edge route, and*
(ii) *a bounding box of dimensions $O(\sqrt{n}) \times O(\sqrt{n}) \times O(n)$.*
Moreover, the drawing can be obtained in $O(n)$ time.

Proof. We achieve a drawing with the characteristics stated in the theorem by eliminating segment-5 from the edge route of the blue (and green) edges in the compact-drawing algorithm (as shown in Fig. 4). The elimination of segment-5 results in segment-4 and segment-6 becoming adjacent to each other and routed on the same XY-plane. Recall from the compact-drawing algorithm that we assigned to each edge a z-value (determining the XY-planes that are used by its edge route) based on a colouring algorithm that use at most $2\lceil \sqrt{n} \rceil - 1$ colours. The small number of colours was possible due to the fact that we only had to avoid having an intersection between two edge routes along their segment-4 (or segment-6) parts. Now we also have to prevent segment-4 of an edge intersecting with segment-6 of another edge. Preventing these intersections is simple. Just devote an individual XY-plane to the routing of each edge. Thus n XY-planes are used for the routing of C_{blue}, n for the routing of C_{green} and 1 for the routing of C_{red}, resulting in a bounding box of height $2n + 1$ grid points and an $O(\sqrt{n}) \times O(\sqrt{n})$ base.

Devoting a distinct XY-plane to the routing of each edge, eliminates the need for building and vertex coloring the conflict graph used in the compact-drawing algorithm. Thus, the dominant part of the running time is due to the computation of the cycle covers, resulting in $O(n)$ time complexity. □

Theorem 4. *Every n vertex maximum degree* 6 *graph G has a three-dimensional orthogonal grid drawing with the following characteristics*:
(i) *at most* 5 *bends per edge route, and*
(ii) *a bounding box of dimensions* $O(\sqrt{n}) \times O(n) \times O(n)$.
Moreover, the drawing can be obtained in $O(n)$ *time.*

Proof. We achieve the stated result by a second refinement of the compact drawing algorithm. This time we route the red edges according to one of the ways described in Section 4.1, since the routing originally employed had 6 bends per edge, one more than the number of bends stated in the theorem. Again we route the blue (and green) edges as described in the proof of Theorem 3 (with 6 bends per edge route), and now we show how to eliminate segment-3, resulting in a drawing with 5 bends per edge route. The purpose of segment-3 of the edge route of a blue edge was to guarantee that it is impossible for a segment-2 of one edge to intersect with a segment-4 of another, since segment-2 and segment-4 were routed on XZ-planes with even and odd Y-coordinates, respectively. Now, with the elimination of segment-3, we have to consider intersections between segment-2 and segment-4 of two different edges. However, observe that segment-2 of one edge e can only intersect with segment-4 of another edge e' originating in the same row of squares as e. Thus, if we devote a different XZ-plane to each of the $\lceil\sqrt{n}\rceil$ edges starting in a given row of squares, then we still get an intersection-free layout. This can be easily achieved by extending the square devoted to each vertex to a rectangle, and routing edges starting from vertices in the ith column of rectangles on the ith XZ-plane within the rectangle, relative to the XZ-plane of the origin vertex ($(-i)$th for C_{green}). In total, for any given row of squares, we use:
- $\lceil\sqrt{n}\rceil$ XZ-planes for the blue edges,
- $\lceil\sqrt{n}\rceil$ XZ-planes for the green edges,
- On XZ-plane through the vertex itself, and
- two XZ-planes devoted to the routing of the red edges.

This gives a layout where each constant size square has been replaced by an $O(1) \times O(\sqrt{n})$ rectangle. This results in a bounding box with a base of dimensions $O(\sqrt{n}) \times O(n)$.

The running time remains $O(n)$ since constant time is spent in determining the new route of each edge. □

Theorem 5. *Every n vertex maximum degree* 6 *graph G has a three-dimensional orthogonal grid drawing with the following characteristics*:
(i) *at most* 4 *bends per edge route, and*
(ii) *a bounding box of dimensions* $O(n) \times O(n) \times O(n)$.
Moreover, the drawing can be obtained in $O(n)$ *time.*

Proof. We achieve the stated result by a third successive refinement of the compact-drawing algorithm. This time we eliminate the only unit-length segment left, that is, segment-7. This segment was used for two reasons: Firstly to guarantee that segment-6

of one edge route does not intersect with segment-8 of another, and secondly, in order to guarantee that segment-6 of an incoming edge to an arbitrary vertex v does not intersect with segment-2 of an outgoing edge (belonging to the same cycle cover) from vertex v. We describe the part of the refinement devoted to avoiding each type of possible intersection separately.

Avoiding intersections between segment-6 *and segment*-8: Observe that only edge routes that terminate at vertices in a given column of rectangle can intersect. Thus, if for each edge route that terminates in the same column of rectangles we devote a distinct YZ-plane, then no intersection of a segment-6 with a segment-8 belonging to a different edge route is possible. We devote a distinct YZ-plane to each segment-8 of each column of rectangles. This implies that the vertex placement has to be refined so that in each column of rectangles each vertex is placed in its own YZ-plane. Given that each column of rectangles consists of at most $\lfloor 2\sqrt{n} \rfloor$ rectangles (due to Lemma 1), it is sufficient to extend each $4 \times (2\lceil \sqrt{n} \rceil + 3)$ rectangle (used in the 5-bend drawing) to a $(3 + \lfloor 2\sqrt{n} \rfloor) \times (2\lceil \sqrt{n} \rceil + 3)$ rectangle and to assign the vertices at the ith row of rectangles, $1 \leqslant i \leqslant \lfloor 2\sqrt{n} \rfloor$, to the $(i+2)$nd column of grid points within the given row of rectangles. This results in a bounding box in the XY-plane with base of dimensions $O(n) \times O(n)$.

Avoiding intersections between segment-6 *and segment*-2: Notice that this type of intersection is only possible between adjacent edges that belong in the same cycle of the blue (or the green) cycle cover.

Without loss of generality, assume that the edges belong in C_{blue} (the case where they belong in C_{green} is treated symmetrically). Let (u, v) and (v, w) be the two edges in question where edges (u, v) immediately precedes edge (v, w) in some cycle of C_{blue}. We describe how to draw these edges so that segment-6 of (u, v) does not intersect with segment-2 of (v, w). Our routing does not require any increase in volume. We avoid intersections by simply routing each edge in an appropriate XY-plane. Consider the vertices of the cycle c of C_{blue} that contains (u, v) and (v, w). We examine two cases:

Case 1: *Not all vertices of cycle c are placed in the same row of rectangles.* Consider edges (u, v) and (v, w) and let vertices u and v be placed at points $(x_u, y_u, 0)$ and $(x_v, y_v, 0)$, respectively. Observe that if $y_u < y_v$, then no overlap between segment-6 of (u, v) and segment-2 of (v, w) is possible since the points of the former segment have smaller Y-coordinate, than the points of the latter segment (see Fig. 6).

If follows that an intersection is only possible if $y_u \geqslant y_v$. However, in this case we can avoid the intersection by routing edge (u, v) on an XY-plane which is above the XY-plane in which edge (v, w) is routed. This is because the points of segment-6 of (u, v) have larger Z-coordinate than the points of segment-2 of (v, w) (see Fig. 7).

Based on the above observations, it is easy now to assign XY-planes to the routing of the edges of cycle c so that there is no intersection between two adjacent edges of the cycle. Identify 2 adjacent (in the direction of cycle c) edges (u, v) and (v, w) such that $y_u < y_v$. The fact that not all vertices are placed in the same row of rectangles guarantees that these two edges exist. Starting from edge (v, w), traverse cycle c and assign decreasing Z-coordinates (i.e., lower XY-planes) to its edges.

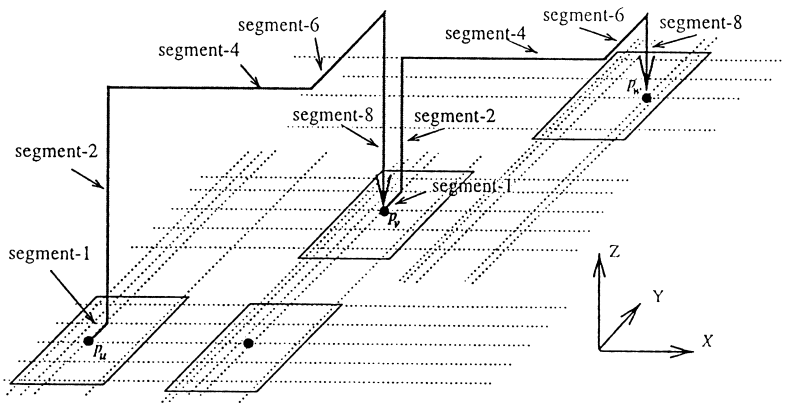

Fig. 6. No overlap is possible in the refinement of Theorem 5 for the case where not all vertices are placed in the same row of rectangles and $y_v < y_u$.

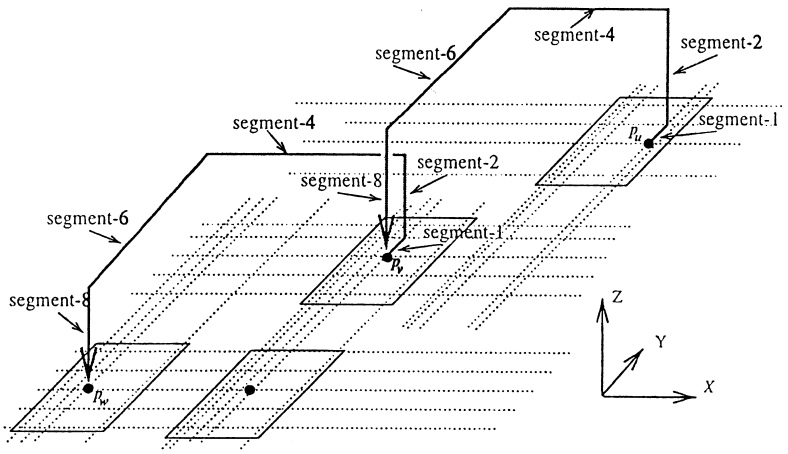

Fig. 7. Avoiding overlaps in the refinement of Theorem 5 for the case where not all vertices are placed in the same row of rectangles and $y_v \geqslant y_u$.

Edge (v,w) is assigned the largest Z-coordinate while edge (u,v) is assigned the smallest. Edges (u,v) and (v,w) cannot intersect because $y_u < y_v$ and no other pair of adjacent edges can intersect with each other due to the decreasing Z-coordinates which were assigned to their routes.

Case 2: *All vertices of cycle c are placed in the same row of rectangles.* Recall that after the refinement that eliminated segment-3 (see proof of Theorem 4) a different XZ-plane is devoted to the routing of each edge starting at the same row of rectangles. Consider again two adjacent edges (u,v) and (v,w) of cycle c and assume that they are routed on the ith and the jth XZ-planes of their row of rectangles, respectively. The XZ-planes in a given row of rectangles are numbered relative to

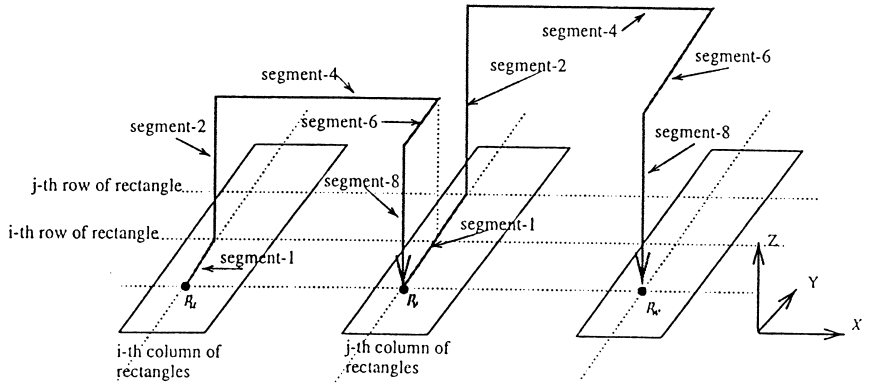

Fig. 8. Avoiding overlaps in the refinement of Theorem 5 for the case where all vertices are placed in the same row of rectangles and $x_u < x_v$. Row numbers within a rectangle are relative to the position of the vertex on the rectangle.

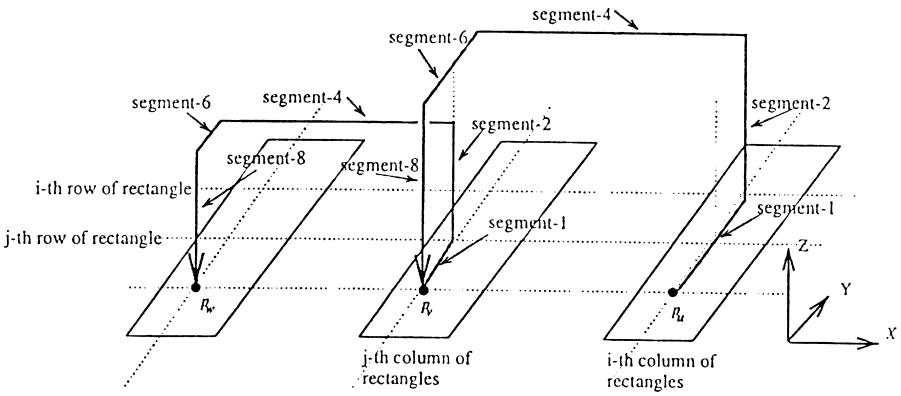

Fig. 9. Avoiding overlaps in the refinement of Theorem 5 for the case where all vertices are placed in the same row of rectangles and $x_u > x_x$. Row numbers within a rectangle are relative to the position of the vertex on the rectangle.

the XZ-plane on which the vertices of the row of rectangles are positioned (see Figs. 8 and 9). According to the drawing of Theorem 4, vertex u and v are positioned in the ith and the jth column of rectangles, respectively. Let x_u and x_v be the X-coordinates of vertices u and v, respectively. Then we have that $i < j$ if and only if $x_u < x_v$. We guarantee that no intersection between the routes of (u, v) and (v, w) takes place by routing (u, v) on a XY-plane below (i.e., with smaller Z-coordinate) the one in which (v, w) is routed if and only if $x_u < x_v$ (that is, $i < j$). To see that no overlap is possible, consider first the case where $x_u < x_v$, that is, $i < j$ (see Fig. 8). In this case, the points of segment-6 of (u, v) have smaller Y-coordinate than that of the points of segment-2 of (v, w) and thus no intersection is possible. In the case where $x_u > x_v$, that is, $i > j$ (see Fig. 9), the points of segment-6 of (u, v)

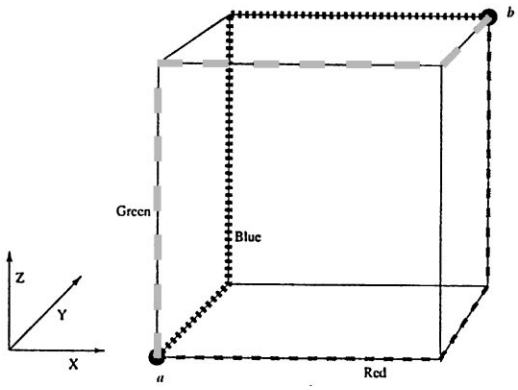

Fig. 10. Three disjoint paths along the edges of an isothetic cube.

have larger Z-coordinate than that of the points of segment-2 of (v,w) and again no intersection is possible.

The running time remains $O(n)$ since the assignment of Z-coordinates to the edge routes can be implemented by traversing the edges of each cycle at most twice, one time to determine whether all vertices are positioned in the same row of rectangles (and if not the edge which will be allocated the largest Z-coordinate) and one time to actually allocate the Z-coordinates. □

5. The 3-bends algorithm

In this section we present an algorithm, referred to as the 3-*bends* algorithm, which draws a graph of maximum degree at most 6 with 3 bends per edge route in a box of dimensions $O(n) \times O(n) \times O(n)$. The drawing produced by the 3-bends algorithm is quite different from the 4-bend drawing with identical (in the sense of asymptotic notation) bounding box dimensions produced in Section 4.2.

The 3-bends algorithm uses the preprocessing algorithm of Section 2 to obtain a 6-regular directed graph G' together with edge disjoint cycle covers $C_{\text{red}}, C_{\text{green}}$ and C_{blue} for G'. However, it places the vertices of G (that is, the vertices of G') on the diagonal of a cube. More precisely, it arbitrarily assigns numbers $1,2,\ldots,n$ to the vertices and places vertex $a \in \{1,2,\ldots,n\}$ at location $(3a,3a,3a)$. For simplicity, we use the same symbol to denote both a vertex and its location.

Each pair a,b of vertices in G' determines an isothetic cube $C(a,b)$ with a and b at opposite corners. For the purpose of defining routes for possible coloured edges of the form (a,b), we first define red, green and blue paths between a and b along the edges of the cube $C(a,b)$ as illustrated in Fig. 10. Each path of cube edges has only 2 bends.

Later, we are going to route a coloured edge (a,b) of G' close to the coloured path of cube edges of the same colour on $C(a,b)$, so that no point on the actual route for

(a,b) is more than one unit away from some point on the corresponding coloured path of cube edges on $C(a,b)$. The following easy lemma shows that by doing this, we guarantee that routes for edges that are not incident to a common vertex of G' do not intersect.

Lemma 2. *Suppose that $a,b,c,d \in V$ are distinct vertices of G'. Suppose that p is a point on a cube edge of $C(a,b)$ and that q is a point on a cube edge of $C(c,d)$. Then the Euclidean distance between p and q is at least* 3.

The above lemma, together with the fact that coloured paths of cube edges on the same cube get close to one another only in the vicinity of the ends of the paths, suggests that the main difficulty will be to ensure that routes do not intersect in the vicinity of their endpoints.

Given this intuition about the routing strategy, we first give an overview of the 3-bends algorithm and then give the routing details.

The 3-bends algorithm

1. Use the preprocessing algorithm of Section 2 to compute the 6-regular directed graph G' and its three cycle covers $C_{\text{red}}, C_{\text{green}}$ and C_{blue}.
2. Arbitrarily number the vertices of G' 1 to n; for $1 \leqslant a \leqslant n$, place vertex a at $(3a, 3a, 3a)$.
3. Compute and draw the routes for each coloured edge of G' that arises from an edge of G, as described in detail below.

To specify the edge routes in detail, it is helpful first to introduce the concept of a local minimum or maximum of a coloured cycle of length at least 2. Suppose, for example, that $u_1 \rightarrow u_2 \rightarrow \cdots \rightarrow u_k \rightarrow u_1$ is the red cycle through some vertex b of G'. Hence $b = u_i$ for some $1 \leqslant i \leqslant k$, and each u_j on the cycle is a number in the range 1 to n. The successor u_{i+1} of $b = u_i$ may be a larger or a smaller number than u_i. Hence as one moves along the red cycle, the coordinate values associated with the vertices on the cycle are sometimes increasing, sometimes decreasing. This motivates the following definition.

Definition 2. A vertex u_i is a *local maximum* with respect to a coloured cycle $u_1, \ldots, u_k$ of length $k > 1$ if its value is greater than that of both its predecessor and its successor, i.e., if $u_{i-1} < u_i$ and $u_{i+1} < u_i$, where subscript arithmetic is modulo k.

A *local minimum* is defined analogously. A vertex is *normal* with respect to a colour if it is neither a local maximum nor a local minimum for that colour. (A vertex of a loop is normal.) An edge (u_i, u_{i+1}), $u_i \neq u_{i+1}$, is said to be *increasing* or *decreasing* if $u_i < u_{i+1}$ or $u_i > u_{i+1}$, respectively. The route for a coloured edge (u_i, u_{i+1}) will depend on whether the edge is increasing or decreasing and also, on whether u_{i+1} is a local minimum, a local maximum, or normal for that colour. Note, for example, that

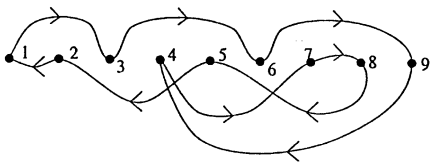

Fig. 11. Categories of edges in the 3-bends algorithm.

a vertex may be a local maximum with respect to one colour and a local minimum with respect to another colour.

We define five categories of edges:

- *normal increasing edges*: edges (u_i, u_{i+1}) with $u_i < u_{i+1}$ and $u_{i+1} < u_{i+2}$,
- *normal decreasing edges*: edges (u_i, u_{i+1}) with $u_i > u_{i+1}$ and $u_{i+1} > u_{i+2}$,
- *loops*,
- *edges entering a local minimum*: edges (u_i, u_{i+1}) with $u_i > u_{i+1}$ and $u_{i+1} < u_{i+2}$,
- *edges entering a local maximum*: edges (u_i, u_{i+1}) with $u_i < u_{i+1}$ and $u_{i+1} > u_{i+2}$.

The categories (except for the case of loops) are illustrated in Fig. 11 for a cycle that passes through vertices numbered 1 to 9 with no omissions. In the cycle $1 \rightarrow 3 \rightarrow 6 \rightarrow 9 \rightarrow 4 \rightarrow 7 \rightarrow 8 \rightarrow 5 \rightarrow 2 \rightarrow 1$, edges $(1,3)$, $(3,6)$ and $(4,7)$ are normal increasing, edges $(8,5)$ and $(5,2)$ are normal decreasing, edges $(9,4)$ and $(2,1)$ enter a local minimum, and edges $(6,9)$ and $(7,8)$ enter a local maximum.

A red normal edge (u_i, u_{i+1}) (increasing or decreasing) is routed along the red path of cube edges of the cube $C(u_i, u_{i+1})$, that is, the routes for red normal edges are as shown in Table 2. Note that each route for a normal red edge has two bends. The table also describes the routes for loops. A red loop uses the $+X$ and $-Z$ ports of the grid point at which its vertex u_i is placed, and consists of the unit square determined by these two ports. The route contains 3 bends. Finally the table describes the routes for red edges entering a local minimum or maximum with exactly 3 bends per edge route. These red edges are routed near the red path of cube edges, but slightly offset, as illustrated in Fig. 12. The blue and green edges are routed similarly.

The routing scheme leads to the following theorem.

Theorem 6. *Every n vertex graph G of maximum degree at most 6 has a three-dimensional orthogonal grid drawing with the following characteristics*:

(i) *at most 3 bends per edge*,
(ii) *maximum edge route length $9(n-1)+2$, and*
(iii) *a bounding box of dimensions $3n \times 3n \times 3n$.*

Moreover, the drawing can be obtained in $O(n)$ time.

Proof. A red *conduit* between distinct vertex locations a and b consists of the line segments for all possible red routes between a and b. Blue and green conduits are

Table 2
The coordinates of the routes for red edges in the 3-bends algorithm

Edge category	Segment	Start point	→	Finish point
Normal increasing red edge	1	$(3u_i, 3u_i, 3u_i)$	→	$(3u_{i+1}, 3u_i, 3u_i)$
	2	$(3u_{i+1}, 3u_i, 3u_i)$	→	$(3u_{i+1}, 3u_{i+1}, 3u_i)$
	3	$(3u_{i+1}, 3u_{i+1}, 3u_i)$	→	$(3u_{i+1}, 3u_{i+1}, 3u_{i+1})$
Normal decreasing red edge	1	$(3u_i, 3u_i, 3u_i)$	→	$(3u_i, 3u_i, 3u_{i+1})$
	2	$(3u_i, 3u_i, 3u_{i+1})$	→	$(3u_i, 3u_{i+1}, 3u_{i+1})$
	3	$(3u_i, 3u_{i+1}, 3u_{i+1})$	→	$(3u_{i+1}, 3u_{i+1}, 3u_{i+1})$
Red loop	1	$(3u_i, 3u_i, 3u_i)$	→	$(3u_i + 1, 3u_i, 3u_i)$
	2	$(3u_i + 1, 3u_i, 3u_i)$	→	$(3u_i + 1, 3u_i, 3u_i - 1)$
	3	$(3u_i + 1, 3u_i, 3u_i - 1)$	→	$(3u_i, 3u_i, 3u_i - 1)$
	4	$(3u_i, 3u_i, 3u_i - 1)$	→	$(3u_i, 3u_i, 3u_i)$
Red edge entering a local minimum	1	$(3u_i, 3u_i, 3u_i)$	→	$(3u_i, 3u_i, 3u_{i+1} - 1)$
	2	$(3u_i, 3u_i, 3u_{i+1} - 1)$	→	$(3u_i, 3u_{i+1}, 3u_{i+1} - 1)$
	3	$(3u_i, 3u_{i+1}, 3u_{i+1} - 1)$	→	$(3u_{i+1}, 3u_{i+1}, 3u_{i+1} - 1)$
	4	$(3u_{i+1}, 3u_{i+1}, 3u_{i+1} - 1)$	→	$(3u_{i+1}, 3u_{i+1}, 3u_{i+1})$
Red edge entering a local maximum	1	$(3u_i, 3u_i, 3u_i)$	→	$(3u_{i+1} + 1, 3u_i, 3u_i)$
	2	$(3u_{i+1} + 1, 3u_i, 3u_i)$	→	$(3u_{i+1} + 1, 3u_{i+1}, 3u_i)$
	3	$(3u_{i+1} + 1, 3u_{i+1}, 3u_i)$	→	$(3u_{i+1} + 1, 3u_{i+1}, 3u_{i+1})$
	4	$(3u_{i+1} + 1, 3u_{i+1}, 3u_{i+1})$	→	$(3u_{i+1}, 3u_{i+1}, 3u_{i+1})$

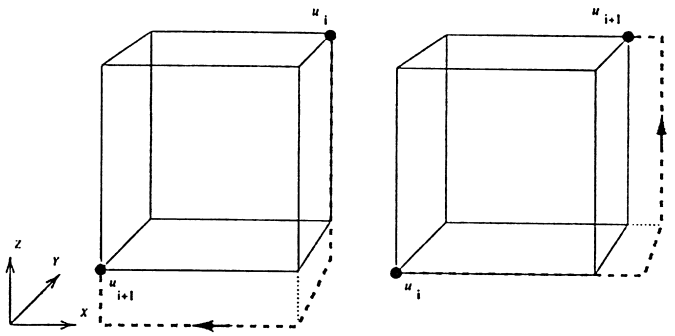

Fig. 12. Red routes in the 3-bends algorithm entering a local minimum and maximum, respectively.

defined similarly. See the dark segments in Fig. 13. Note that red conduits and red loops use the $+X$ and $-Z$ ports of endpoints a and b; blue conduits and blue loops use the $+Y$ and $-X$ ports of their endpoints; and green conduits and green loops use the $-Y$ and $+Z$ ports of their endpoints.

We first show that no two edge routes produced by the 3-bends algorithm intersect each other, that is, no two edge routes share a common point other than a route endpoint. By Lemma 2, two edges routes that intersect must have one or both endpoints

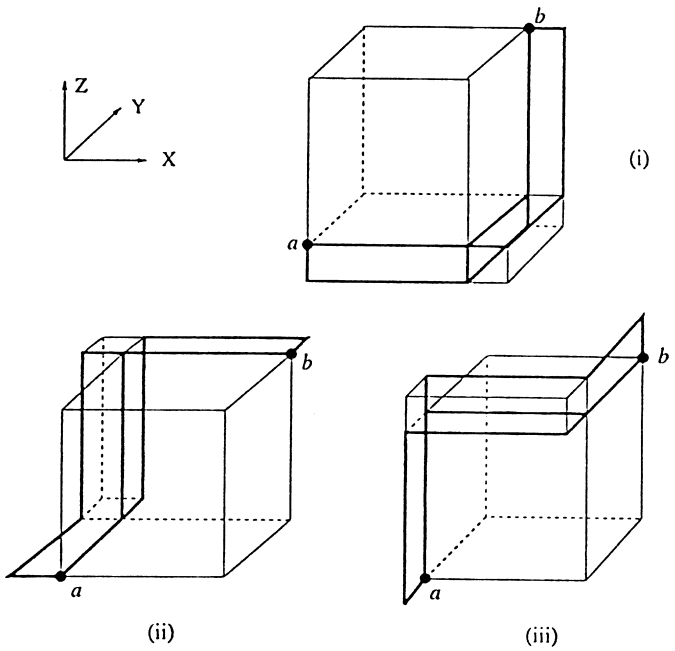

Fig. 13. Conduits: (i) red, (ii) blue, and (iii) green.

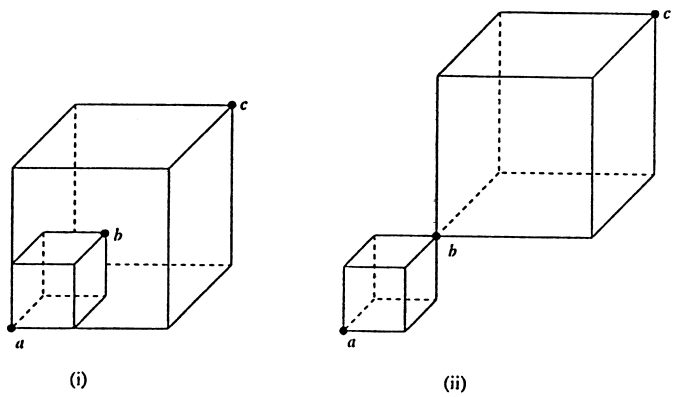

Fig. 14. Cubes for routes with a common endpoint.

in common. In what follows, consider Fig. 13, which illustrates coloured conduits, and Fig. 14, which illustrates the underlying cubes for routes of nonloop edges with a common endpoint.

Clearly a route for a loop intersects no other route of the same or different colour; from now on, we consider the routes for nonloop edges.

Suppose that two routes of the same colour share both endpoints, say a and b where $b > a$; note that b is a local maximum and a is a local minimum (see Fig. 14(i)).

Although the routes use the same conduit, they use different line segments within this conduit and hence do not intersect.

Next, consider two edge routes of the same colour that share exactly one endpoint. Suppose that the three endpoints determined by the two edge routes are a, b and c, where $a < b < c$. If a is the common endpoint, then a is a local minimum. Hence the arc entering a is offset while the arc leaving a is not; thus the two routes do not intersect. If c is the common endpoint, then c is a local maximum and an analogous argument applies. If b is the common endpoint then b must be normal (see Fig. 14(ii)); hence routes of the edges incident to b lie along cubes that intersect only at b, and the routes do not intersect.

Now consider two routes of different colours i and j. If they share both endpoints, their conduits do not intersect. For $a < b < c$, the colour i conduit between a and b does not intersect the colour j conduit between b and c, and likewise for conduits of different colours between a and c and between b and c. Hence routes of different colours do not intersect.

To establish the size of the bounding box, observe that vertex 1 and vertex n are assigned to grid points $(3,3,3)$ and $(3n,3n,3n)$, respectively. In the worst case, vertex 1 is a local minimum and vertex n is a local maximum in all of the three cycle covers. In this case the grid points $(2,2,2)$ and $(3n+1,3n+1,3n+1)$ are the diametrically opposite corners of the bounding box, leading to a drawing of dimensions $3n \times 3n \times 3n$ (recall that we measure the extent on the bounding box in each dimension in terms of the number of grid points rather than in terms of length).

The length of the longest edge route corresponds to the case where edge $(1,n)$ enters a local maximum at vertex n, or the symmetric case where edge $(n,1)$ enters a local minimum at vertex 1. In both cases, the edge route overshoots the end-vertex by one unit of length in one direction (depending on the colour of the edge) and then corrects it. Thus, the longest edge route is two units longer than the distance between vertex 1 and vertex n i.e., $9(n-1)+2$ units.

The part of the algorithm that dominates the runtime is the computation of the cycle covers; this takes time $O(n)$ by Theorem 1, since the maximum degree Δ of G is at most 6. □

6. Drawings for graphs of maximum degree 4

In this section, we present a drawing algorithm for graphs of maximum degree 4. The drawing algorithm again uses the partition of a regular graph into disjoint cycle covers. The drawing produced has 3 bends per edge and is contained in a $2n \times (n+2) \times 3$ bounding box, where n is the number of vertices of the graph. Compared to the 3-bends algorithm of the previous section for graphs of maximum degree 6, which required a bounding box of $O(n^3)$ volume, the algorithm of this section demonstrates that graphs of maximum degree 4 require a bounding box of asymptotically smaller volume for their drawing. An overview of the algorithm follows.

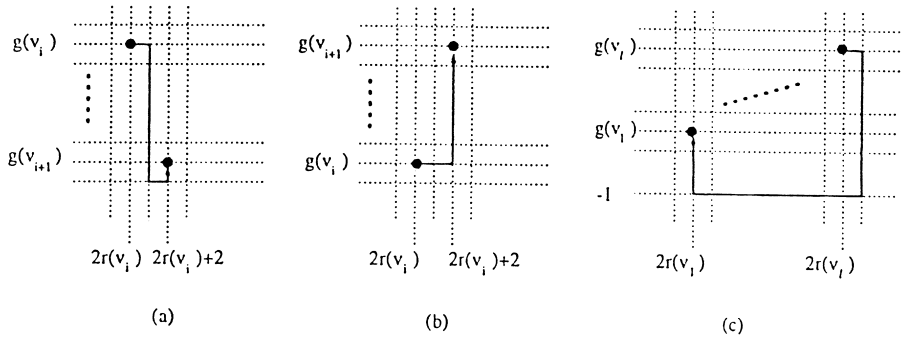

Fig. 15. The edge routes of red edges in the 3-bend algorithm for graphs of maximum degree 4.

Algorithm for graphs of maximum degree 4.

1. Use the preprocessing algorithm of Section 2 to compute the 4-regular directed graph G' and its two cycle covers, C_{red} and C_{green}.
2. Suppose that C_{red} consists of k directed cycles $c_1, c_2, \ldots, c_k$. Use these cycles to assign for each vertex $v \in V$ an order number $r(v)$ with respect to cycle cover C_{red}, $1 \leqslant r(v) \leqslant n$, as follows. Arbitrarily choose a starting vertex for c_1 and assign numbers in increasing order to the remaining vertices of c_1 by following the directed cycle. Then order the vertices of the remaining cycles in a similar fashion, ordering the vertices of cycle c_i before those of cycle c_j if and only if $i < j$.
3. In similar fashion, assign for each vertex $v \in V$ an order number $g(v)$ with respect to cycle cover C_{green}, $1 \leqslant g(v) \leqslant n$.
4. Place vertex $v \in V$ at location $(2r(v), g(v), 0)$.
5. Design routes for each edge of G' as described in detail below. Draw those routes arising from edges of G.

We describe how to route the edges of a single cycle in each of the cycle covers C_{red} and C_{green}. In a similar way we route all the remaining cycles in each cycle cover.

6.1. Routing the red cycle cover

The route for a red loop uses the $+X$ and $-Y$ ports of the grid point at which its vertex is placed, and consists of the unit square determined by these two ports.

Now consider an arbitrary cycle $c = v_1, v_2, \ldots, v_l$ of C_{red}, $l \geqslant 2$.

We first consider the route edge (v_i, v_{i+1}), $1 \leqslant i \leqslant l-1$. Vertex v_i is at location $(2r(v_i), g(v_i), 0)$ while vertex v_{i+1} is at location $(2r(v_i)+2, g(v_{i+1}), 0)$. We consider two cases.

Case 1: $g(v_i) > g(v_{i+1})$. The route of edge (v_i, v_{i+1}) consists of the following four segments, illustrated in Fig. 15(a).

Segment	Start point	→	Finish point
1	$(2r(v_i), g(v_i), 0)$	→	$(2r(v_i)+1, g(v_i), 0)$
2	$(2r(v_i)+1, g(v_i), 0)$	→	$(2r(v_i)+1, g(v_{i+1})-1, 0)$
3	$(2r(v_i)+1, g(v_{i+1})-1, 0)$	→	$(2r(v_i)+2, g(v_{i+1})-1, 0)$
4	$(2r(v_i)+2, g(v_{i+1})-1, 0)$	→	$(2r(v_i)+2, g(v_{i+1}), 0)$

Case 2: $g(v_i) < g(v_{i+1})$. The route of edge (v_i, v_{i+1}) consists of the following two segments, illustrated in Fig. 15(b).

Segment	Start point	→	Finish point
1	$(2r(v_i), g(v_i), 0)$	→	$(2r(v_i)+2, g(v_i), 0)$
2	$(2r(v_i)+2, g(v_i), 0)$	→	$(2r(v_i)+2, g(v_{i+1}), 0)$

The route of the last edge (v_l, v_1) of the cycle consists of the following four segments, illustrated in Fig. 15(c) (the figure assumes that $g(v_1) < g(v_l)$; the same route is valid when $g(v_1) > g(v_l)$).

Segment	Start point	→	Finish point
1	$(2r(v_l), g(v_l), 0)$	→	$(2r(v_l)+1, g(v_l), 0)$
2	$(2r(v_l)+1, g(v_l), 0)$	→	$(2r(v_l)+1, -1, 0)$
3	$(2r(v_l)+1, -1, 0)$	→	$(2r(v_l), -1, 0)$
4	$(2r(v_l), -1, 0)$	→	$(2r(v_l), g(v_1), 0)$

Observe that, in all cases, the edge routes for the red edges do not intersect each other. This is because an edge always leaves a vertex with a $+X$ port and enters a vertex with a $-Y$ port.

Further, it is easy to see that there is no overlap between the routes of edges that belong to the same cycle of C_{red}. In order to prove that the routes of edges that belong to different cycles of C_{red} do not intersect, simply observe that the edges of each cycle are routed entirely within a strip of columns the width and position of which depend on the number of vertices in the cycle and the ordering of the vertices with respect to cycle cover C_{red}, respectively. The ordering of the vertices with respect to C_{red} guarantees that the vertical strips devoted to the routing of different cycles of C_{red} do not overlap.

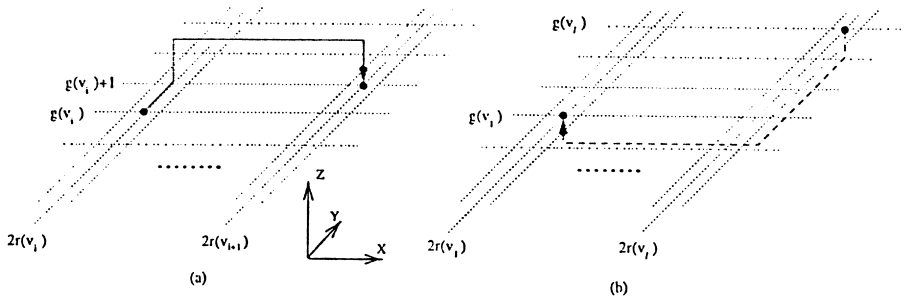

Fig. 16. The edge routes of green edges in the 3-bend algorithm for graphs of maximum degree 4.

6.2. Routing the green cycle cover

The route for a green loop uses the $+Y$ and $+Z$ ports of the grid point at which its vertex is placed, and consists of the unit square determined by these two ports.

Consider an arbitrary cycle $c = v_1, v_2, \ldots, v_l$ of C_{green}, $l \geqslant 2$.

We first consider the route of edges (v_i, v_{i+1}), $1 \leqslant i \leqslant l-1$. Vertex v_i is at location $(2r(v_i), g(v_i), 0)$ while vertex v_{i+1} is at location $(2r(v_{i+1}), g(v_i)+1, 0)$ since $g(v_{i+1}) = g(v_i)+1$. The route of edge (v_i, v_{i+1}) consists of the four segments illustrated in Fig. 16(a) for the case $r(v_{i+1}) > r(v_i)$.

Segment	Start point	→	Finish point
1	$(2r(v_i), g(v_i), 0)$	→	$(2r(v_i), g(v_i)+1, 0)$
2	$(2r(v_i), g(v_i)+1, 0)$	→	$(2r(v_i), g(v_i)+1, 1)$
3	$(2r(v_i), g(v_i)+1, 1)$	→	$(2r(v_{i+1}), g(v_i)+1, 1)$
4	$(2r(v_{i+1}), g(v_i)+1, 1)$	→	$(2r(v_{i+1}), g(v_i)+1, 0)$

The route of the last edge (v_l, v_1) of the cycle consists of the four segments illustrated in Fig. 16(b).

Segment	Start point	→	Finish point
1	$(2r(v_l), g(v_l), 0)$	→	$(2r(v_l), g(v_l), -1)$
2	$(2r(v_l), g(v_l), -1)$	→	$(2r(v_l), g(v_1), -1)$
3	$(2r(v_l), g(v_1), -1)$	→	$(2r(v_l), g(v_1), -1)$
4	$(2r(v_1), g(v_1), -1)$	→	$(2r(v_1), g(v_1), 0)$

Note that in both Figs. 16(a) and (b) it is assumed that $r(v_i) < r(v_{i+1}), 1 \leqslant i \leqslant l-1$. The same routes are valid when $r(v_i) > r(v_{i+1})$.

It is easy to check that the routes of edges of C_{green} do not intersect, as the routing of each cycle of C_{green} lies within a strip of adjacent rows (parallel to the X-axis), and the strips of distinct cycles are disjoint. The described drawing allows us to state the following theorem.

Theorem 7. *Every n vertex maximum degree 4 graph G has a three-dimensional orthogonal grid drawing with*

(i) *at most 3 bends per edge route,*
(ii) *$4n+1$ maximum edge length, and*
(iii) *a bounding box of dimensions $2n \times (n+2) \times 3$.*

Moreover, the drawing can be obtained in $O(n)$ time, where n is the number of vertices.

Proof. The drawing described in this section satisfies all the stated characteristics of the theorem.

By inspection we can verify (i), that is, each edge route has at most 3 bends.

For (ii), note that the maximum edge length clearly occurs when the last edge of a red cycle is as long as possible; this happens when edge (v_n, v_1) is the last edge of a red cycle and moreover $g(v_1) = n-1$ and $g(v_n) = n$. In this case (see Fig. 15(c)), the red edge route moves 1 unit in the $+X$ direction, $n+1$ units in the $-Y$ direction, $2n-1$ units in the $-X$ direction, and n units in the $+Y$ dimension to v_1, for a total of $4n+1$ units of length.

Now consider (iii). Inspection of the routing tables (for red and green) reveals that the smallest X-coordinate is 2, and the largest X-coordinate is $2n+1$; thus the bounding box has X-dimension $(2n+1)-2+1 = 2n$ (counting by grid points). Similarly, we can use extreme values in the routing tables to show that the bounding box has Y and Z-dimensions $n+2$ and 3, respectively.

It has already been established that edges of the same cycle cover do not intersect with each other. Now we check that the routes of edges of C_{red} and C_{green} do not intersect. The edges of C_{red} are routed on plane $Z=0$ and use only the $+X$ and $-Y$ ports, while the edge routes of C_{green} intersect the plane $Z=0$ only in $+Y$ ports. Hence all that remains is to check two cases, namely that a $+Y$ port of a green edge does not intersect either a Y-segment or an X-segment of a red route. For the first case, note that a green $+Y$ port does not intersect a red Y-segment because Y-segments of red edges use either use the $-Y$ port or occupy the grid lines immediately to the right of the vertices. Also, note that any two vertices have X-coordinates which differ by at least 2. For the second case, an intersection is impossible because the X-segments of concern in Figs. 15(a) and (b) are at most 2 units long, and in that case (Fig. 15(b)) the vertex at the grid line containing the bend has the same X-coordinate as the bend. Again, recall that X-coordinates of vertices differ by at least 2.

Thus, the drawing is intersection free.

Table 3
Trade-offs between the number of bends and the drawing size

Maximum degree	Maximum bends per edge	Dimensions
6	7	$O(\sqrt{n}) \times O(\sqrt{n}) \times O(\sqrt{n})$
6	6	$O(\sqrt{n}) \times O(\sqrt{n}) \times O(\sqrt{n})$
6	5	$O(\sqrt{n}) \times O(n) \times O(n)$
6	3	$O(n) \times O(n) \times O(n)$
4	3	$O(1) \times O(n) \times O(n)$

The dominant part of the running time is again due to the computation of the cycle covers, resulting in $O(n)$ time complexity. □

7. Conclusion

In this paper, we studied three-dimensional orthogonal intersection-free grid drawings for n vertex graphs. We presented several drawing algorithms, focusing on the number of bends per edge route, on the dimension of the bounding box of the drawing and on trade-offs between these quantities.

Table 3 summarises these trade-offs.

The work presented in this paper raises new issues. These include the following.

1. It would be interesting to know whether a maximum of 3 bends per edge route is best possible. Note, however, that K_7, the 6-regular complete graph on 7 points, *does* have a grid drawing with at most 2 bends per edge [34]. The techniques presented in this paper do not appear to be sufficient on their own to produce a drawing with two bends per edge, if one is possible. The fact that our algorithms were based on the decomposition of a regular graph into cycle covers suggests that the use of different decomposition techniques might lead to layouts of two bends per edge. If this proves feasible, then the trade-off issues should be revisited.
2. The drawing produced by the 3-bends algorithm is quite different from the 4-bend drawing with identical (in the sense of asymptotic notation) bounding box dimensions that was produced by the refinement of the compact-drawing algorithm. This suggests that it might still be possible to get drawings with fewer bends per edge route than the drawings produced by the refinements of the compact-drawing algorithm, for bounding boxes of similar dimensions.
3. The drawing with 3 bends per edge route for graphs of maximum degree 6 requires $O(n^3)$ volume while the drawing with the same number of bends per graphs of maximum degree 4 requires $O(n^2)$. Further, one can achieve $O(n^{3/2})$ volume using Theorem 2. This suggests that it is worthwhile studying the trade-off between bends and volume for graphs of maximum degree 4.
4. The drawing produced for graphs of maximum degree 4 raises the issue of drawings of constant height. Such drawings might be useful for VLSI applications. The study

of trade-offs between the number of bends and the base of the bounding box of the drawing, for a fixed constant height, is worth investigation.

Acknowledgements

We would like to thank Therese Beidl for pointing out Schrijver's algorithm [28], David Wood for his comments on an earlier version of the paper, and the anonymous referees for many helpful suggestions.

References

[1] C. Berge, Graphs, 2nd revised Edition, North-Holland, Amsterdam, 1985.
[2] S. Bhatt, S. Cosmadakis, The Complexity of Minimizing Wire Lengths in VLSI Layouts, Inform. Process. Lett. 25 (1987) 263–267.
[3] T. Biedl, Embedding Nonplanar Graphs in the Rectangular Grid, Rutcor Research Report 27-93, 1993.
[4] T. Biedl, G. Kant, A better heuristic for orthogonal graph drawings, Comput. Geom. Theory Appl. 9 (3) (1998) 159–180.
[5] T. Biedl, New lower bounds for orthogonal graph drawings, Graph Drawing, Lecture Notes in Computer Science, Vol. 1027, Springer, Berlin, 1995, pp. 28–39.
[6] T. Biedl, T. Shermer, S. Whitesides, S. Wismath, Orthogonal 3D graph drawing, in: GD97, Lecture Notes in Computer Science, Vol. 1353, Springer, Berlin, 1998, 76–86.
[7] J.A. Bondy, U.S.R. Murty, Graph Theory with Applications, North Holland, Amsterdam, 1976.
[8] G. DiBattista, P. Eades, R. Tamassia, I. Tollis, Algorithms for drawing graphs: an annotated bibliography, Comput. Geom. Theory Appl. 4 (1994) 235–282.
[9] D. Dolev, F.T. Leighton, H. Trickey, Planar Embeddings of Planar Graphs, in: F.P. Preparata (Ed.), Advances in Computing Research 2, JAI Press Inc., Greenwich CT, USA, 1984, pp. 147–161.
[10] P. Eades, C. Stirk, S. Whitesides, The techniques of Komolgorov and Barzdin for three dimensional orthogonal graph drawings, Inform. Process. Lett. 60 (2) (1996) 97–103.
[11] P. Eades, A. Symvonis, S. Whitesides, Two algorithms for three dimensional orthogonal graph drawing, in: S. North (Ed.), GD96, Springer Lecture Notes in Computer Science, Vol. 1190, Springer, Berlin, 1996, pp. 139–154.
[12] S. Even, Graph Algorithms, Computer Science Press, Rockville, MD, 1979.
[13] S. Even, G. Granot, Rectilinear Planar Drawings with Few Bends in Each Edge, Technical Report 797, Computer Science Department, Technion, 1994.
[14] M. Formann, F. Wagner, The VLSI layout problem in various embedding models, in: WG91, Lecture Notes in Computer Science, Vol. 484, Springer, Berlin, 1991, 130–139.
[15] A. Garg, R. Tamassia, On the computational complexity of upward and rectilinear planarity testing, Proceedings of the Graph Drawing: DIMACS International Workshop (GD'94), Lecture Notes in Computer Science, Vol. 894, Springer, Berlin, 1995, pp. 286–297.
[16] P. Hall, On representation of subsets, J. London Math. Soc. 10 (1935) 26–30.
[17] D.S. Johnson, A. Demers, J.D. Ullman, M.R. Garey, R.L. Graham, Worst-case Performance Bounds for Simple One-dimensional Packing Algorithms, SIAM J. Comput. 3 (1974) 299–325.
[18] G. Kant, Drawing planar graphs using the canonical ordering, Algorithmica 16 (1) (1996) 4–32.
[19] A.N. Kolmogorov, Ya.M. Barzdin, About realisation of sets in 3-dimensional space, Problems Cybernet. (1967) 261–268.
[20] M. Kramer, J. van Leeuwen, The complexity of wire routing and finding minimum area layouts for arbitrary VLSI circuits, in: F. Preparata (Ed.), Advances in Computing Research, Vol. 2 (VLSI Theory), 1984, 129–146.
[21] A. Papakostas, I. Tollis, Algorithms for area-efficient orthogonal drawings, Comput. Geom. Theory Appl. 9 (1-2) (1998) 83–110.

[22] A. Papakostas, I. Tollis, Incremental orthogonal graph drawing in three dimensions, in: GD97, Lecture Notes in Computer Science, Vol. 1353, Springer, Berlin, 1998, pp. 52–63.
[23] J. Petersen, Die theorie der regulären graphen, Acta Math. 15 (1891) 193–220.
[24] F.P. Preparata, Optimal three-dimensional VLSI layouts, Math. Systems Theory 16 (1983) 1–8.
[25] A.L. Rosenberg, Three-dimensional integrated circuitry, in: Kung, Sproule, Steele (Eds.), VLSI Systems and Computations, Computer Science Press, 1981, pp. 69–80.
[26] A. L. Rosenberg, Three-dimensional VLSI: a case study, J. ACM 30 (3) (1983) 397–416.
[27] M. Schäffter, Drawing graphs on rectangular grids, Discrete Appl. Math. 63 (1995) 75–89.
[28] A. Schrijver, Bipartite edge-coloring in $O(\Delta m)$ time, SIAM J. Comput. 28 (3) (1998) 841–846.
[29] J. Storer, On minimal node-cost planar embeddings, Networks 14 (1984) 181–212.
[30] R. Tamassia, On embedding a graph in the grid with a minimum number of bends, SIAM J. Comput. 16 (3) (1987) 421–443.
[31] R. Tamassia, I. Tollis, A unified approach to visibility representations of planar graphs, Discrete Comput. Geom. 1 (1986) 321–341.
[32] R. Tamassia, I. Tollis, Efficient embeddings of planar graphs in linear time, IEEE Symposium on Circuits and Systems (1987) 495–498.
[33] R. Tamassia, I. Tollis, Planar grid embedding in linear time, IEEE Trans. Circuits and Systems Vol. 36 (9) (1989) 1230–1234.
[34] D. Wood, On higher-dimensional orthogonal graph drawing, Proceedings of CATS'97, Computing: The Australasian Theory Symposium, February 1997, Sydney, Australia, pp. 3–8.