Two Algorithms for Three Dimensional Orthogonal Graph Drawing

Peter Eades[1] and Antonios Symvonis[2] and Sue Whitesides[3]

[1] Dept. of Computer Science, University of Newcastle, N.S.W. 2308, Australia; eades@cs.newcastle.edu.au; supported by ARC
[2] Basser Dept. of Computer Science, University of Sydney, N.S.W. 2006, Australia symvonis@cs.su.oz.au; supported by an ARC Institutional grant
[3] School of Computer Science, McGill University, Montreal, Canada H3A 2A7; sue@cs.mcgill.ca; supported by FCAR and NSERC

Abstract. We use basic results from graph theory to design two algorithms for constructing 3-dimensional, intersection-free orthogonal grid drawings of n vertex graphs of maximum degree 6. Our first algorithm gives drawings bounded by an $O(\sqrt{n}) \times O(\sqrt{n}) \times O(\sqrt{n})$ box; each edge route contains at most 7 bends. The best previous result generated edge routes containing up to 16 bends per route. Our second algorithm gives drawings having at most 3 bends per edge route. The drawings lie in an $O(n) \times O(n) \times O(n)$ bounding box. Together, the two algorithms initiate the study of bend/bounding box trade-off issues for 3-dimensional grid drawings.

1 Introduction

The 3-dimensional orthogonal grid consists of *grid points* whose coordinates are all integers, together with the axis-parallel *grid lines* determined by these points. A *3-dimensional orthogonal grid drawing* of a graph G 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, i.e., no internal point, not necessarily a grid point, of one edge route may lie in any other edge route. Throughout this paper, *grid* refers to the 3-dimensional orthogonal grid, and *grid drawing* refers to the type of 3-dimensional orthogonal grid drawing just described. Note that because each grid point lies at the intersection of three grid lines, any graph that admits a grid drawing necessarily has maximum vertex degree at most 6.

Figure 1 shows a grid drawing of a graph. This particular drawing lies in a $2 \times 3 \times 2$ bounding box. The edge route joining the two extremal vertices in the Z-direction lies along the top, back and bottom faces of the box and contains 2 bends. The edge route joining the two extremal vertices in the X-direction also contains 2 bends, but passes through the interior of the box.

While the graph drawing literature has extensively investigated 2-dimensional grid drawings of graphs (see [7]), 3-dimensional grid drawing has been little studied. Our research is motivated in part by recent interest in exploring the utility of 3-dimensional drawings of graphs for visualization purposes. It should also be

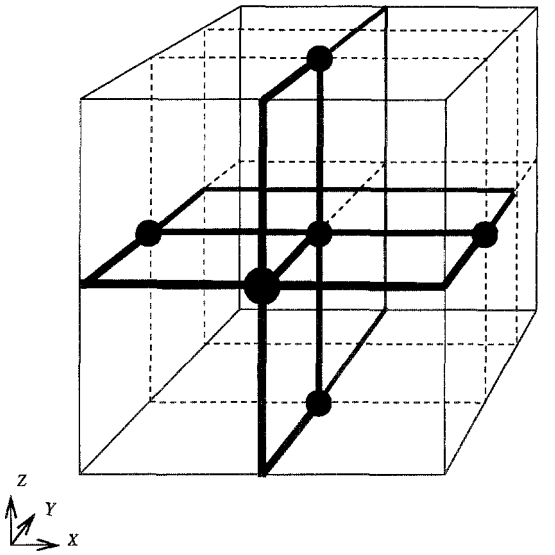

Fig. 1. *An orthogonal grid drawing*

noted that since VLSI technology now permits the stacking of many layers, this work may be relevant to that application area as well.

This paper offers two algorithms for obtaining grid drawings. Both algorithms use basic graph theory methods to preprocess the input graph by colouring its edges; then the algorithms route edges according to their colour class. One algorithm produces drawings with a small number of bends per edge, while the other algorithm produces more compact drawings, but at the cost of an increased number of bends per edge. This raises for 3-dimensional grid drawing the same kind of bend versus bounding box trade-off issues that have been studied in the 2-dimensional case. 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 (e.g., volume, maximum dimension, sum of dimensions, length of long diagonal) we simply give the dimensions of the bounding box of the drawings produced by our algorithms, and we do not define compactness precisely. Note that volume is generally not the most appropriate measure of bounding box suitability for 3-dimensional drawings to be displayed in projection on screen or paper for visualization 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 [9] of 16 bends maximum per edge route, described further below. Komolgorov and Bardzin ([16]; see also [9]) showed that no algorithm can produce asymptotically more compact drawings; hence we refer to our first algorithm as the *compact drawing* algorithm.

Our second algorithm produces drawings having only 3 bends maximum per edge route. However, the bounding box of the drawing increases to one of dimensions $O(n) \times O(n) \times O(n)$. We refer to this algorithm as the *3-bends* algorithm. The maximum of 3 bends per route exhibited by the 3-bends algorithm may be best possible; we conjecture the existence of a graph of maximum degree at most 6 that cannot be embedded in the grid with a maximum of 2 bends per edge route. However, the compact drawing algorithm suggests that the bounding box can be reduced in size, at least if more bends per route are admitted.

For two dimensions, the problem of producing compact orthogonal grid drawings with few bends has received much attention. Several methods for obtaining crossing-free drawings in area $O(n) \times O(n)$ with $O(1)$ bends per edge are available, e.g. [3, 4, 11, 15, 21, 23, 24, 25, 17]. Some recent research aims at reducing the number of bends while allowing crossings between edges, e.g. [5].

Not surprisingly, problems that are seemingly computationally intractable arise in both 2-dimensional and 3-dimensional grid drawing. For example, in two dimensions, minimizing the number of bends is NP-complete [12], but can be solved in polynomial time for any fixed planar embedding [22]. Eades, Stirk and Whitesides [9] have shown how to generalize various 2-dimensional NP-completeness results such as minimizing the number of bends, the volume of the drawing, and the maximum individual edge route length [12, 8, 2] to 3 dimensions.

As cited above, [9] provided an algorithm, based on the technique of Komolgorov and Bardzin [16], for obtaining 3-dimensional orthogonal grid drawings. The algorithm produces drawings for n-vertex, maximum degree at most 6 graphs in which each edge route has at most 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 [9].

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 3-dimensional VLSI layout from the early 1980's [18, 19, 20].

The rest of this paper is organized as follows. Section 2 gives the simple graph theoretic methods that our two algorithms use to preprocess the input graph. Sections 3 and 4 present and analyse the compact drawing algorithm and the 3-bends algorithm, respectively. Section 5 concludes.

2 Preliminaries

This section gives the preprocessing step, based on elementary graph theoretic methods, that both the compact drawing algorithm and the 3-bends algorithm 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.

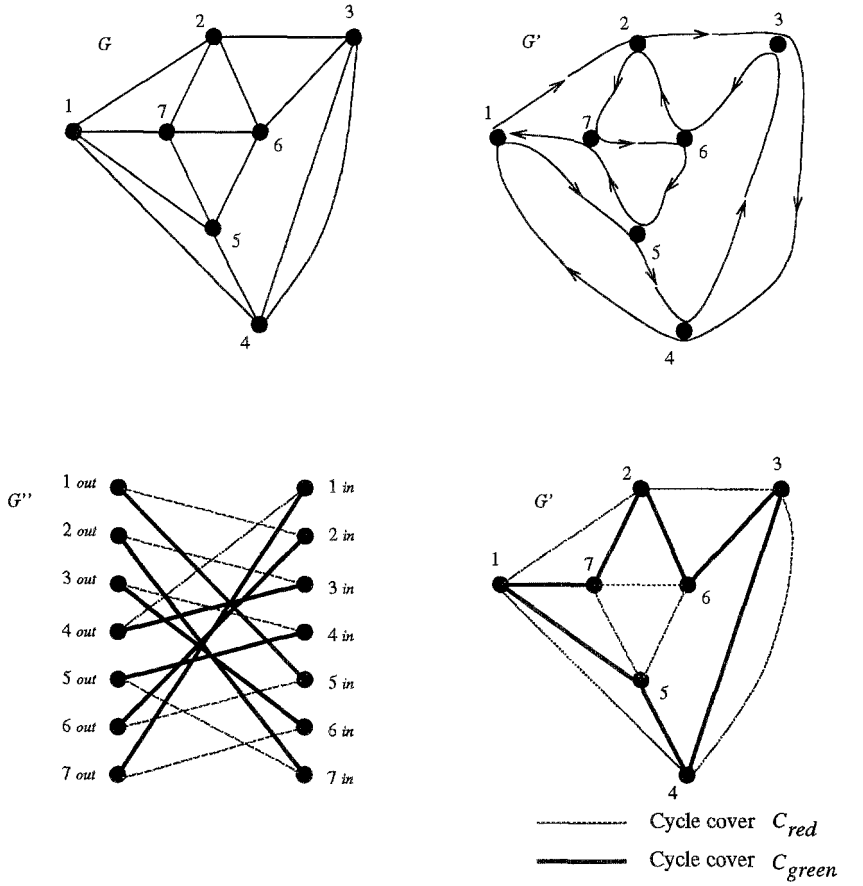

Fig. 2. *A decomposition of a 4-regular undirected graph into 2 cycle covers.*

Theorem 2. *Suppose that $G = (V, E)$ is an undirected graph of maximum degree at most 6. Then there is a directed graph $G' = (V', E')$ on the same set of vertices $V = V'$ such that*

- *each vertex of G' has indegree at most 3 and outdegree at most 3;*
- *G is a subgraph of the underlying undirected graph of G'; and*
- *the arcs of G' can be partitioned into three edge disjoint cycle covers.*

Furthermore, given $G = (V, E)$, the directed graph G' and its three cycle covers can be computed in $O(n^{3/2})$ time, where n is the number of vertices.

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 6. Add one self-loop (v, v) to each $v \in V$ of degree 4, and add two self-loops to each vertex of degree 2 to create a regular graph (with self-loops and multiple

edges allowed) of degree 6. 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 3 and outdegree 3 at each node. Clearly these operations can be performed in $O(|V|)$ time.

Now from G', construct an undirected bipartite graph $G'' = (V_{out} \cup V_{in}, E'')$ by taking $V_{out} = \{v_{out} \mid v \in V\}$, $V_{in} = \{v_{in} \mid v \in V\}$, and $E'' = \{(u_{out}, v_{in}) \mid (u, v) \in E'\}$. Note that G'' is 3-regular and bipartite. By Hall's Theorem [13][10, p. 138] G'' contains a perfect matching; colour its edges *red* and remove them. The remaining graph is bipartite and 2-regular, so it again contains a perfect matching. Colour its edges *green* and remove them. The remaining edges form a perfect matching; colour them *blue*.

Now colour each directed arc (u, v) of G' with the colour given to (u_{out}, v_{in}) of G''. This gives each node of G' exactly one incoming and one outgoing arc of each of the three colours. Hence the arcs of G' are partitioned into three coloured subgraphs $C_{red}, C_{green}, C_{blue}$, each of which is a cycle cover for G'. Since a maximum matching in an arbitrary bipartite graph with n vertices and m edges can be computed in $O(m\sqrt{n})$ [14], and since for G'' we have $n = 2|V|$ and $m = 3|V|$, the computation of the three cycle covers can be done in $O(|V|^{3/2})$ time, including the time used to compute G' from G.

Figure 2 shows the analogous process of partitioning the edges of a 4-regular undirected graph into two cycle covers.

Preprocessing Algorithm: We call the algorithm contained in the proof of the previous theorem the *preprocessing algorithm*. 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 arcs of G' into three edge disjoint cycle covers, denoted $C_{red}, C_{green}, C_{blue}$. The algorithm runs in $O(|V|^{3/2})$ time.

Both the compact drawing algorithm and the 3-bends algorithm specify arc routes for the three 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 arcs of G'. Self-loops and arcs of G' that do not arise from edges of G are simply not drawn.

In the following sections, it is helpful to keep in mind that each node of G' has exactly one incoming and exactly one outgoing arc of each colour.

3 Compact Drawing

This section gives our compact drawing algorithm, which takes an 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 $12\sqrt{n}$, and bounding box dimensions $(3\lceil\sqrt{n}\,\rceil + 2) \times 5\lceil\sqrt{n}\,\rceil \times 4\lceil\sqrt{n}\,\rceil$, where $|V| = n$.

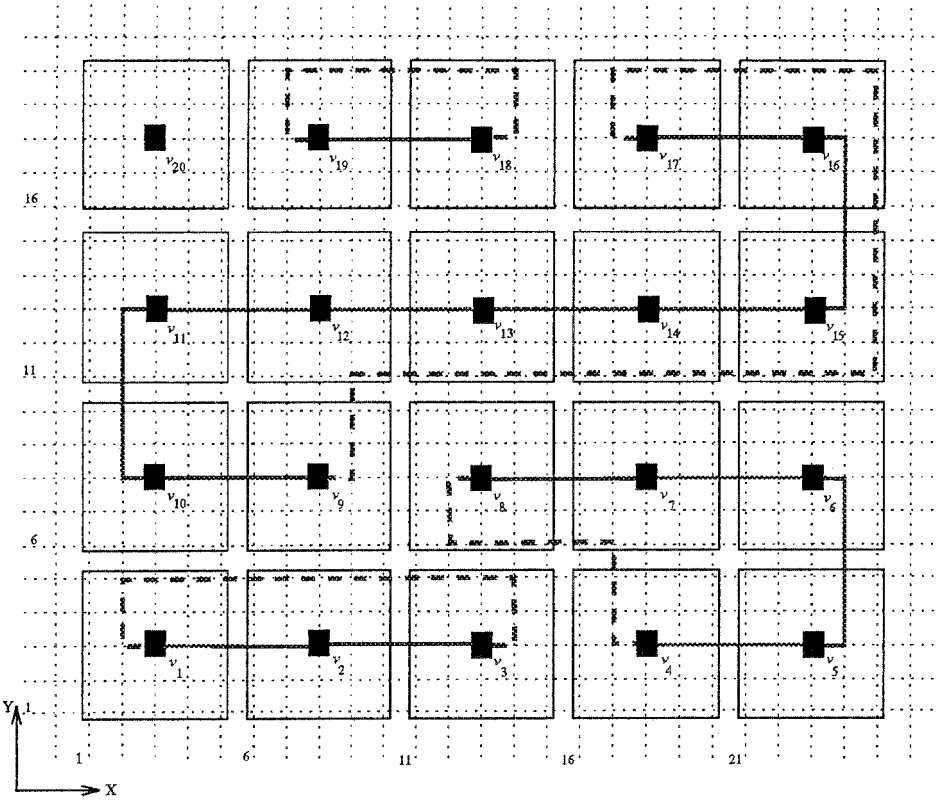

Fig. 3. *The layout of the vertices of G and the routing of edges in cycle cover C_{red}.*

Overview of the compact drawing algorithm:
The compact drawing algorithm has the following basic steps, the first of which has already been described and the rest of which are described in detail in subsequent subsections.

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

3.1 Processing C_{red}

Suppose cycle cover C_{red} consists of k directed cycles $c_1, c_2, \ldots, c_k$, and use these cycles to order the nodes of G' as follows. Arbitrarily choose a starting node from c_1, and order the remaining nodes of c_1 by following the cycle; then order the nodes of the remaining cycles in similar fashion, ordering the the nodes 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 $Z = 0$ plane as follows. For $0 \leq i, j < \lceil \sqrt{n} \rceil$, $p_{i,j} = (5i + 3, 5j + 3, 0)$.

Rather than give explicit formulas for node placement and arc routing, we illustrate this in the context of a specific example from which the reader can easily infer the general rules.

Consider the following cycle cover C_{red}: $c_1 = \langle v_1, v_2, v_3 \rangle$, $c_2 = \langle v_4, \ldots, v_8 \rangle$, $c_3 = \langle v_9, \ldots, v_{17} \rangle$, $c_4 = \langle v_{18}, v_{19} \rangle$, and $c_5 = \langle v_{20} \rangle$.

Using the order just obtained from cycle cover C_{red}, assign the nodes of G' to grid points $p_{i,j}$ in the snake-like fashion illustrated in Fig. 3. Then route the arcs of the cycles as shown in the figure. In particular, the figure shows how to handle cycles whose nodes lie within one row of special grid points, cycles whose nodes lie in parts of two neighboring rows, and cycles whose nodes occupy more than one row.

By $Sq(i, j)$ we refer to the set of grid points within the square of side length 4 centered at special grid point $p_{i,j}$. That is, $Sq(i, j) = \{5i + k, 5j + l, 0) \mid 1 \leq k, l \leq 5\}$.

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. These will be used in proving that no two arc routes intersect illegally.

Observation 1 *The routes for red arcs satisfy the following properties:*
i) The edge routes for arcs in C_{red} have at most 6 bends per route.
ii) The grid points having Y-coordinate one greater or less than the Y-coordinate of a special grid point $p_{i,j}$ lie in no red arc routes.
iii) The grid points contained in routes connecting nodes in the same cycle of C_{red} are entirely contained in the squares to which those nodes are assigned.

Observation 2 *Consider the positive length segments parallel to the Y-axis that are contained in routes for red arcs. Except for segments intersecting the first or last column of squares, these segments contain no grid points that differ by exactly 2 in X-coordinate from the X-coordinate of a special grid point $p_{i,j}$.*

3.2 Routing C_{blue} and C_{green}

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

The route of an arbitrary arc (v, w) of C_{blue} is illustrated in Fig. 4, where $v_{i,j}$ denotes the vertex placed at $p_{i,j}$.

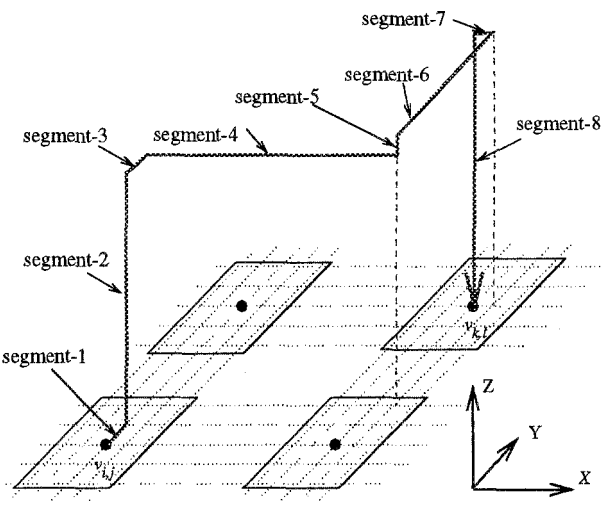

Fig. 4. *The route of arc (v, w) of π_2 with 7 bends.*

Let vertices v and w be assigned to special grid points having coordinates $(x_v, y_v, 0)$ and $(x_w, y_w, 0)$, respectively. Then the route for arc (v, w) consists of the 7 segments described in the table and shown in the figure. Here we defer until later the specification of the value z_{vw} in the table, noting for the moment that for all v, w, the value of z_{vw} will be an *odd* integer.

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

3.3 Proof of Correctness

Now we prove that the routes of any two arcs do not intersect illegally.

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 > 1) also contains this intersection point. Thus, we need only consider the possibility of

intersections of segments longer than 1, i.e, intersections among even-numbered segments.

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

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 Z-coordinate, one with odd Z-coordinate. Hence the only possible intersections are between the segment-4's of two different routes or between the segment-6's of two different routes.

Finally we explain how to choose the values for z_{vw}. This is done by using the method of [9]. Consider arc (v, w). The route for this arc can intersect only with the arc routes with origin in the row of squares in which vertex v is placed and the arc 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 arc set of cycle cover C_{blue}. An edge is inserted between two vertices in the graph H whenever the vertices correspond to arcs with start nodes in the same row or end nodes in the same column. H has maximum degree $2(\lceil\sqrt{n}\,\rceil - 1)$ and thus it has a vertex colouring by $2\lceil\sqrt{n}\,\rceil - 1$ colours, which can be obtained in $O(n\sqrt{n})$ time by a greedy algorithm [6, Brook's Theorem]. Say that colour $c, 1 \leq c \leq 2\lceil\sqrt{n}\,\rceil - 1$, has been assigned to the vertex of H that corresponds to blue arc (v, w). Then set $z_{vw} = 2c - 1$.

Now we give the main result of Section 2.

Theorem 3. *Every n-vertex maximum degree 6 graph G has a 3-dimensional, intersection-free orthogonal drawing with the following characteristics: i) at most 7 bends per edge route, ii) $(16\lceil\sqrt{n}\,\rceil - 7)$ maximum edge length, and iii) a bounding box of dimensions $(3\lceil\sqrt{n}\,\rceil + 2) \times 5\lceil\sqrt{n}\,\rceil \times (8\lceil\sqrt{n}\,\rceil - 6)$. Moreover, the drawing can be obtained in $O(n^{3/2})$ time.*

Proof. Except for their endpoints, green arc routes lie below the $Z = 0$ plane and obviously do not intersect blue arc routes illegally. Based on Observation 1 and the fact that the red arc routes lie in the $Z = 0$ plane, we conclude that they do not intersect blue or green arc routes illegally. Thus, the drawing obtained by routing edges of a graph G according to the routes for their corresponding directed arcs in G' gives a proper grid drawing.

The fact that there are at most 7 bends per edge route can be seen by inspection. The drawing fits in an axis-aligned box of dimensions $5\lceil\sqrt{n}\,\rceil \times 5\lceil\sqrt{n}\,\rceil \times (8\lceil\sqrt{n}\,\rceil - 6)$. The maximum possible edge route length corresponds either to a blue arc route starting from $Sq(1, \lceil\sqrt{n}\,\rceil)$ and going to $Sq(\lceil\sqrt{n}\,\rceil, 1)$ 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 $(18\lceil\sqrt{n}\,\rceil - 9)$ units.

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 the ones in the first and last column of squares can be replaced by a 3×5 rectangle. As a result of this change, the modified drawing fits in a box of dimensions $(3\lceil\sqrt{n}\,\rceil + 2) \times 5\lceil\sqrt{n}\,\rceil \times (8\lceil\sqrt{n}\,\rceil - 6)$ and the maximum length of a route is $(16\lceil\sqrt{n}\,\rceil - 7)$ units.

The time consuming parts of the compact drawing algorithm are the partition of the arcs of the 6-regular directed graph G' graph into red, green and blue cycle covers and the vertex colouring used in the routing of the blue and green arcs. Both of these operations can be completed in $O(n^{3/2})$ time.

4 The 3-Bends Algorithm

In this section we present an algorithm for constructing an orthogonal drawing of a graph $G = (V, E)$ of maximum degree at most 6 with at most 3 bends per edge. This is a substantial reduction in the number of bends from Section 3. The cost of the decrease in the number of bends is that the bounding box dimensions increase to $O(n) \times O(n) \times O(n)$, where $n = |V|$.

The 3-bends algorithm uses the preprocessing algorithm of Section 2 to obtain a 6-regular directed graph G' together with arc disjoint cycle covers C_{red}, C_{green} and C_{blue} for G'. However, it places the vertices of G (nodes of G') on the diagonal of a $3n \times 3n \times 3n$ 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 $p_a = (3a, 3a, 3a)$.

Each pair a, b of nodes in G' determines an isothetic cube $C(a, b)$ with p_a and p_b at opposite corners. For the purpose of defining routes for possible coloured arcs of the form (a, b), we first define red, green and blue paths between p_a and p_b along the edges of the cube $C(a, b)$ as illustrated in Fig. 5. Each path of cube edges has only 2 bends.

Later, we are going to route a coloured arc (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 arcs that are not incident to a common node of G' do not intersect.

Lemma 4. *Suppose that $a, b, c, d \in V$ are distinct nodes 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.

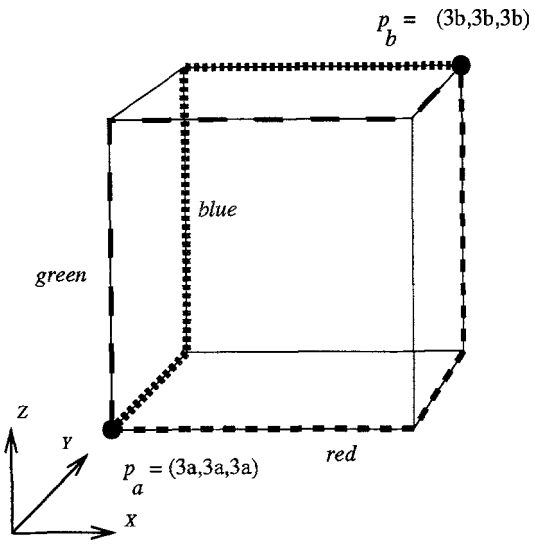

Fig. 5. *Disjoint paths*

Overview of the 3-bends algorithm:
The 3-bends algorithm has the following basic steps.

1. Use the preprocessing algorithm of Section 2 to compute the 6-regular directed graph G' and its three cycle covers C_{red}, C_{green} and C_{blue}.
2. Arbitrarily number the nodes $V' = V$ of G' 1 to $n = |V|$; for $1 \leq a \leq n$, place node a at p_a.
3. Design the routes for each coloured arc (a, b) of G' as described in detail below.
4. For each undirected edge of G, draw the route of the corresponding coloured, directed arc of G'.

To specify the arc routes in detail, it is helpful first to introduce the concept of a local minimum or maximum of a coloured cycle. Suppose, for example, that $u_1 \rightarrow u_2 \rightarrow \ldots \rightarrow u_k \rightarrow u_1$ is the red cycle through some node b of G'. Hence $b = u_i$ for some $1 \leq i \leq k$, 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 nodes on the cycle are sometimes increasing, sometimes decreasing. This motivates the following definition.

Definition 5. A node u_i is a *local maximum* with respect to a coloured cycle $u_1, \ldots, u_k$ 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. An arc (u_i, 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 arc (u_i, u_{i+1}) will depend on whether the arc is increasing or decreasing and also, on whether u_{i+1} is a local minimum, a local maximum, or neither for that colour. Note, for example, that a node may be a local maximum with respect to one colour and a local minimum with respect to another colour.

We define four categories of arcs:

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

The categories are illustrated in Fig. 6 for a cycle that passes through nodes numbered 1 to 9 with no omissions.

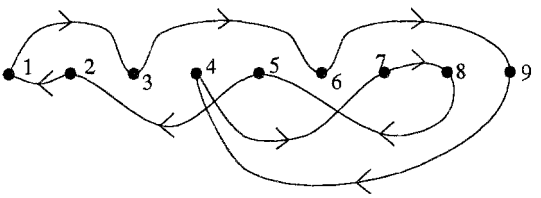

Fig. 6. *Categories*

In the cycle $1 \to 3 \to 6 \to 9 \to 4 \to 7 \to 8 \to 5 \to 2 \to 1$, arcs $(1,3)$, $(3,6)$ and $(4,7)$ are normal increasing, arcs $(8,5)$ and $(5,2)$ are normal decreasing, arcs $(9,4)$ and $(2,1)$ enter a local minimum, and arcs $(6,9)$ and $(7,8)$ enter a local maximum.

A red normal arc (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 arcs are:

- *normal increasing red arc*: $(3u_i, 3u_i, 3u_i) \to (3u_{i+1}, 3u_i, 3u_i) \to (3u_{i+1}, 3u_{i+1}, 3u_i) \to (3u_{i+1}, 3u_{i+1}, 3u_{i+1})$ and
- *normal decreasing red arc*: $(3u_i, 3u_i, 3u_i) \to (3u_i, 3u_i, 3u_{i+1}) \to (3u_i, 3u_{i+1}, 3u_{i+1}) \to (3u_{i+1}, 3u_{i+1}, 3u_{i+1})$.

Note that each route for a normal red arc has two bends.

The other red arcs are routed near the red path of cube edges, but slightly offset, as described below:

- *red arcs entering a local minimum:*
  $(3u_i, 3u_i, 3u_i) \rightarrow (3u_i, 3u_i, 3u_{i+1}-1) \rightarrow (3u_i, 3u_{i+1}, 3u_{i+1}-1) \rightarrow (3u_{i+1}, 3u_{i+1}, 3u_{i+1}-1) \rightarrow (3u_{i+1}, 3u_{i+1}, 3u_{i+1})$.
  This route is illustrated in Fig. 7.
- *red arcs entering a local maximum:*
  $(3u_i, 3u_i, 3u_i) \rightarrow (3u_{i+1}+1, 3u_i, 3u_i) \rightarrow (3u_{i+1}+1, 3u_{i+1}, 3u_i) \rightarrow (3u_{i+1}+1, 3u_{i+1}, 3u_{i+1}) \rightarrow (3u_{i+1}, 3u_{i+1}, 3u_{i+1})$.
  This route is illustrated in Fig. 8.

Note that each of these red arc routes has 3 bends.

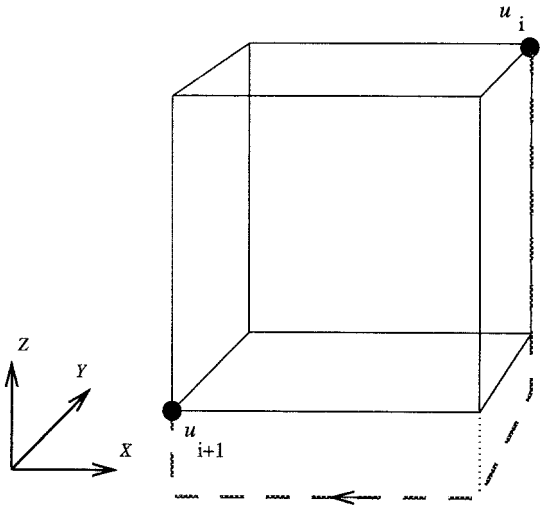

Fig. 7. *Red route into a local minimum*

The blue and green arcs are routed similarly.

Theorem 6. *Every n-vertex graph G of maximum degree at most 6 has an orthogonal grid drawing with the following characteristics: i) at most 3 bends per edge, ii) maximum edge length* $9(n-1)+2$*, and iii) a bounding box of dimensions* $(3n-2) \times (3n-3) \times (3n-2)$*. Moreover, the drawing can be obtained in* $O(n^{3/2})$ *time.*

Proof. Observe that each segment of each arc route lies in at least one plane that is parallel to one of the coordinate planes and that contains some vertex position $p_a = (3a, 3a, 3a)$. Thus if a pair of routes intersect, they have an intersection point in such a plane. For a give generic vertex position $p_a = (3a, 3a, 3a)$, it is easy to determine the places on each of the planes $X = 3a$, $Y = 3a$ and $Z = 3a$ where route segments could possibly lie or pass through. One can exploit the similarity of the construction for routes of different colours to further simplify the task. It is straightforward to check that no illegal intersection of routes can occur.

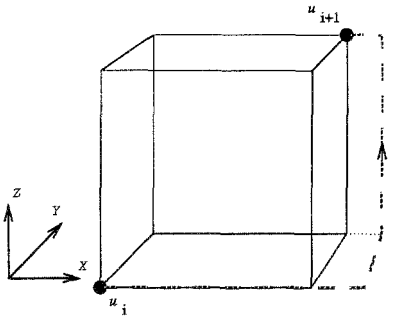

Fig. 8. *Red route into a local maximum*

The slow part of the running time is the time required to compute the cycle covers, which is $O(n^{3/2}$. The longest possible edge route would be the one connecting the extreme vertices on the diagonal. The length of this edge and the bounding box dimensions are evident.

Finally, we observe that the technique of the previous theorem can be extended in a simple manner to give a result for orthogonal grid drawing in arbitrary dimension $d \geq 3$.

Theorem 7. *Suppose G is a graph of maximum degree at most $2d$. Then there is an orthogonal grid drawing of G in dimension d with at most d bends per edge.*

5 Conclusions

We have presented two algorithms for producing grid drawings of n-vertex graphs of maximum degree 6. Both of algorithms compute an associated 6-regular directed graph G' from the input graph G together with three edge disjoint cycle covers for G'.

We note that applying this cycle cover decomposition to the algorithm of [9] eliminates the need for the dummy vertices that algorithm introduces and immediately reduces the number of bends produced from a maximum of 16 per edge route to a maximum of 8 per route. Our first algorithm further reduces this number to 7, and our second algorithm reduces it to 3, at the expense in the latter case of increased bounding box dimensions.

Our results suggest the study of trade-offs between the number of bends in routes and the dimensions of the drawing, and they contribute upper bounds for such a study.

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 [26].

Acknowledgment We thank David Wood for showing us that K_7 has a grid embedding with at most 2 bends per edge route; we had originally conjectured that K_7 requires at least 3 bends on some edge route.

References

1. C. Berge, *Graphs and Hypergraphs*, North Holland, Amsterdam, 1973.
2. S. Bhatt, S. Cosmadakis, "The Complexity of Minimizing Wire Lengths in VLSI Layouts", *Information Processing Letters*, Vol. 25, 1987, pp. 263–267.
3. T. Biedl, Embedding Nonplanar Graphs in the Rectangular Grid, *Rutcor Research Report 27-93*, 1993.
4. T. Biedl and G. Kant, "A Better Heuristic for Orthogonal Graph Drawings", *Proc. 2^{nd} European Symposium on Algorithms (ESA '94)*, Lecture Notes in Computer Science, Vol. 855, Springer Verlag, 1994, pp. 24-35.
5. T. Biedl, "New Lower Bounds for Orthogonal Graph Drawings", *Graph Drawing*, Lecture Notes in Computer Science, Vol. 1027, Springer Verlag, 1995, pp. 28–39.
6. J. A. Bondy, U. S. R. Murty, *Graph Theory with Applications*, North Holland, Amsterdam, 1976.
7. G. DiBattista, P. Eades, R. Tamassia, I. Tollis, "Algorithms for Drawing Graphs: An Annotated Bibliography", *Computational Geometry: Theory and Applications*, Vol. 4, 1994, pp. 235–282.
8. D. Dolev, F. T. Leighton, H. Trickey, "Planar Embeddings of Planar Graphs", *Advances in Computing Research 2* (ed. F. P. Preparata), JAI Press Inc., Greenwich CT, USA, 1984, pp. 147–161.
9. P. Eades, C. Stirk, S. Whitesides, The Techniques of Komolgorov and Bardzin for Three Dimensional Orthogonal Graph Drawings, *TR 95-07, Dept. of Computer Science, University of Newcastle NSW, Australia*, October 1995.
10. Shimon Even, *Graph Algorithms*, Computer Science Press, 1979.
11. S. Even and G. Granot, "Rectilinear Planar Drawings with Few Bends in Each Edge", *Technical Report 797, Computer Science Department, Technion*, 1994.
12. A. Garg, R. Tamassia, On the Computational Complexity of Upward and Rectilinear Planarity Testing, TR CS-94-10, Dept. of Computer Science, Brown University, 1994.
13. P. Hall, "On Representation of Subsets", *J. London Mathematical Society*, Vol. 10, 1935, pp. 26–30.
14. J. E. Hopcroft, R. M. Karp, "An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs", *SIAM J. Comput.*, Vol. 2 (4), 1973, pp. 225–231.
15. Goos Kant, "Drawing Planar Graphs Using the lmc-Ordering", Proc. 33^{rd} IEEE Symp. on Foundations of Computer Science, 1992, pp. 101-110.
16. A. N. Komolgorov, Ya. M. Bardzin, "About Realisation of Sets in in 3-Dimensional Space", *Problems in Cybernetics*, March 1967, pp. 261–268.
17. A. Papakostas and I. Tollis, "A Pairing Technique for Area-Efficient Orthogonal Drawings", these proceedings.
18. F. P. Preparata, "Optimal Three-Dimensional VLSI Layouts", *Mathematical Systems Theory*, Vol. 16, 1983, pp.1–8.

19. A. L. Rosenberg, "Three-Dimensional Integrated Circuitry", *Advanced Research in VLSI* (eds. Kung, Sproule, Steele), 1981, pp. 69–80.
20. A. L. Rosenberg, "Three-Dimensional VLSI: A Case Study", *Journal of the ACM*, Vol. 30 (3), 1983, pp. 397–416.
21. Markus Schäffter, "Drawing Graphs on Rectangular Grids", *Discrete Applied Math.* Vol. 63, 1995, pp. 75-89.
22. R. Tamassia, "On Embedding a Graph in the Grid with a Minimum Number of Bends", *SIAM J. Comput.*, Vol. 16 (3), 1987, pp. 421–443.
23. R. Tamassia and I. Tollis, "A Unified Approach to Visibility Representations of Planar Graphs", *Discrete and Computational Geometry*, Vol. 1, 1986, pp. 321–341.
24. R. Tamassia and I. Tollis, "Efficient Embeddings of Planar Graphs in Linear Time", *IEEE Symposium on Circuits and Systems*, 1987, pp. 495–498.
25. R. Tamassia and I. Tollis, "Planar Grid Embedding in Linear Time", *IEEE Transactions on Circuits and Systems*, Vol. 36 (9), 1989, pp. 1230–1234.
26. David Wood, "On Higher-Dimensional Orthogonal Graph Drawing", manuscript, 1996, Dept. of Computer Science, Monash U.