Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE @ DIRECT®

ELSEVIER

Theoretical Computer Science 299 (2003) 151–178

Theoretical Computer Science

www.elsevier.com/locate/tcs

Optimal three-dimensional orthogonal graph drawing in the general position model ☆

David R. Wood[1]

School of Computer Science, Carleton University, Ottawa, Canada

Received 15 January 2001; received in revised form 29 October 2001; accepted 27 November 2001
Communicated by G. Ausiello

Abstract

Let G be a graph with maximum degree at most six. A three-dimensional orthogonal drawing of G positions the vertices at grid-points in the three-dimensional orthogonal grid, and routes edges along grid lines such that edge routes only intersect at common end-vertices. In this paper, we consider three-dimensional orthogonal drawings in the general position model; here no two vertices are in a common grid-plane. Minimising the number of bends in an orthogonal drawing is an important aesthetic criterion, and is NP-hard for general position drawings. We present an algorithm for producing general position drawings with an average of at most $2\frac{2}{7}$ bends per edge. This result is the best known upper bound on the number of bends in three-dimensional orthogonal drawings, and is optimal for general position drawings of K_7. The same algorithm produces drawings with two bends per edge for graphs with maximum degree at most five; this is the only known non-trivial class of graphs admitting two-bend drawings. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Graph algorithm; Graph drawing; Orthogonal; Three-dimensional

1. Introduction

The aim of graph drawing is to display a given graph so that the inherent relational information of the graph is clear to the user. There has been substantial research into automatically drawing graphs in two dimensions (see [15]). Motivated by

☆ A preliminary version of this paper was presented at the 6th International Symposium on Graph Drawing (GD '98), Montréal, August 13–15, 1998.

[1] Research completed at Monash University (Melbourne, Australia), The University of Sydney (Sydney, Australia) where supported by the ARC, and while visiting McGill University (Montréal, Canada).

E-mail address: davidw@scs.carleton.ca (D.R. Wood).

PII: S0304-3975(02)00044-0

experimental evidence suggesting that displaying a graph in three dimensions is better than in two [35,36], there is a growing body of research in three-dimensional graph drawing. Proposed models for three-dimensional graph drawing include straight-line drawings [14,23,29], convex drawings [12,19], spline-curve drawings [24], multilevel drawings of clustered graphs [18], visibility representations [2,10], and of interest in this paper *orthogonal* drawings [4,6,13,16,20–22,26,28,31,33,38–40]; here the edges of the graph are drawn as polygonal chains composed of axis-parallel segments. This style of drawing has applications in three-dimensional VLSI circuit design (see for example [1,34]).

The *three-dimensional orthogonal grid* consists of *grid-points* in three-dimensional space with integer coordinates, together with the axis-parallel *grid-lines* determined by these points. A *three-dimensional orthogonal drawing* of a graph positions each vertex at a unique grid-point, and represents each edge by a polygonal chain (called an *edge route*) composed of contiguous sequences of axis-parallel segments contained in grid-lines, such that (a) the end-points of an edge route are the grid-points representing the end-vertices of the edge, and (b) distinct edge routes only intersect at a common end-vertex. The six directions, called *ports*, in which the edges incident with a vertex v can be routed are denoted by X_v^+, X_v^-, Y_v^+, Y_v^-, Z_v^+ and Z_v^-. We refer to a three-dimensional orthogonal drawing simply as a *drawing*.

Throughout this paper, we consider n-vertex m-edge undirected graphs G with vertex set $V(G)$ and edge set $E(G)$. A graph is assumed to be simple unless explicitly called a multigraph. The set of vertices adjacent to each vertex v is denoted by $V(v)$, and the set of edges incident to v is denoted by $E(v)$. A graph with maximum degree at most Δ is called a *Δ-graph*. Since there are six grid-lines extending from each grid-point, drawings can only exist for 6-graphs. To construct orthogonal drawings of graphs with degree greater than six, vertices can be represented by grid-boxes (see for example [5,9]), or by points in a multidimensional grid [37,38].

1.1. Aesthetic criteria

Every 6-graph has an infinite number of drawings. Various criteria have been proposed in the literature to evaluate the aesthetic quality of a particular drawing. Firstly, the volume of the drawing should be small. The *volume* of a drawing is the volume of the smallest axis-aligned box, called the *bounding box*, which encloses the drawing. So that a two-dimensional drawing has positive volume, we consider the dimensions of the bounding box to be the number of grid-points along each side (which is one more than the actual side length).

Drawings with many bends in the edges appear cluttered and are difficult to visualise, and in VLSI circuits, MANY BENDS increases the cost of production and the chance of circuit failure. Therefore minimising the number of bends is an important aesthetic criterion for orthogonal drawings. Using straightforward extensions of the corresponding two-dimensional NP-hardness results, minimising either the volume or the number of bends in a drawing is NP-hard [20] (also see [32]). This paper focuses on upper bounds for the number of bends in three-dimensional orthogonal drawings. A drawing with no more than b bends per edge is called a *b-bend drawing*.

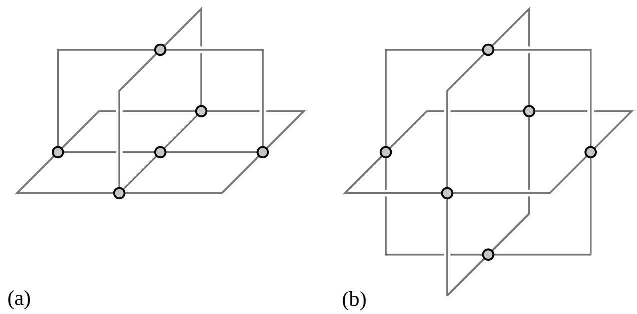

Fig. 1. Drawings of the octahedron graph: (a) with few bends and small volume, (b) orientation-independent.

We say an orthogonal graph drawing is *orientation-dependent* if, loosely speaking, the drawing is significantly different when viewed with respect to one particular dimension; otherwise we say it is *orientation-independent*. For example, in orientation-independent drawings the bounding box asymptotically is a cube, and the minimum-sized axis-aligned box surrounding the vertices asymptotically is a cube. Whether or not orientation-independence is a desirable quality in orthogonal drawings is an application-specific question. For example, for applications in VLSI circuit design, due to the limitations in present-day layering technology, drawings should have constant height. However, we shall take the view that orientation-*in*dependent orthogonal drawings are more aesthetically pleasing than orientation-dependent orthogonal drawings. Orientation-independent drawings are described in a desirable sense as being 'truly three-dimensional' in [5,13].

Other proposed aesthetic criteria include the total or maximum length of edge routes. A number of tradeoffs between aesthetic criteria, most notably between the maximum number of bends per edge route and the volume, have been observed in existing algorithms [22]. Fig. 1 shows two drawings of the octahedron graph.

1.2. Algorithms and bounds

We now summarise the known algorithms and bounds on the aesthetic criteria for three-dimensional orthogonal drawings, including those presented in this paper. We proceed by initially considering drawings with small volume, which typically have relatively many bends, through to drawings with few bends. Table 1 summarises these bounds. We classify drawings as follows. A drawing is *coplanar* if all the vertices are in a single grid-plane; it is a *general position* drawing if no two vertices are in a single grid-plane; and a drawing is *non-collinear* if no two vertices are in a single grid-line.

An early result due to Kolmogorov and Barzdin [26] established a lower bound of $\Omega(n^{3/2})$ on the volume of a drawing (also see [4,34]). This lower bound is asymptotically matched by algorithms of Biedl [4], Eades et al. [20] and Eades et al. [21,22] which produce drawings with $O(n^{3/2})$ volume. The Compact algorithm of Eades et al. [21,22], which routes each edge with at most seven bends, uses the least number of bends out of these algorithms. The above algorithms all produce coplanar

Table 1
Upper bounds for three-dimensional orthogonal graph drawing

Algorithm	Graphs	Max. (avg.) bends	Bounding box	Volume	Orientation independent	Ref.
KB-Algorithm	Multigraph	16	$O(\sqrt{n}) \times O(\sqrt{n}) \times O(\sqrt{n})$	$\Theta(n^{3/2})$	No	Eades et al. [20]
Twente-Algorithm	Multigraph	14	$O(\sqrt{n}) \times O(\sqrt{n}) \times O(\sqrt{n})$	$\Theta(n^{3/2})$	No	Biedl [8]
Non-Collinear	Multigraph	8	$O(\sqrt{n}) \times O(\sqrt{n}) \times O(\sqrt{n})$	$\Theta(n^{3/2})$	Yes	Wood [38]
Compact	Multigraph	7	$O(\sqrt{n}) \times O(\sqrt{n}) \times O(\sqrt{n})$	$\Theta(n^{3/2})$	No	Eades et al. [21,22]
Compact 1	Multigraph	6	$O(\sqrt{n}) \times O(\sqrt{n}) \times O(n)$	$O(n^2)$	No	Eades et al. [22]
Dynamic Spiral	Multigraph	7	$O(\sqrt{n})O \times O(\sqrt{n}) \times O(n)$	$O(n^2)$	No	Closson et al. [13]
Dynamic Staircase	Multigraph	6	$O(n) \times O(n) \times O(1)$	$O(n^2)$	No	Closson et al. [13]
Staircase	Multigraph	5	$O(n) \times O(n) \times O(1)$	$O(n^2)$	No	Closson et al. [13]
BJSW-Algorithm	Multigraph	4	$O(n) \times O(n) \times O(1)$	$O(n^2)$	No	Biedl et al. [6]
Compact 4	Multigraph $\Delta \leqslant 4$	3	$O(n) \times O(n) \times O(1)$	$O(n^2)$	No	Eades et al. [22]
Compact 2	Multigraph	5	$O(\sqrt{n}) \times O(n) \times O(n)$	$O(n^{5/2})$	No	Eades et al. [22]
Compact 3	Multigraph	4	$O(n) \times O(n) \times O(n)$	$O(n^3)$	No	Eades et al. [22]
Diag. Layout & Move.	Simple	4 $(2\frac{2}{7})$	$O(n) \times O(n) \times O(n)$	$2.13n^3$	Yes	Theorem 10
3-Bends	Multigraph	3	$O(n) \times O(n) \times O(n)$	$8n^3$	Yes	Eades et al. [21,22]
Incremental	Multigraph	3	$O(n) \times O(n) \times O(n)$	$4.63n^3$	No	Papakostas and Tollis [31]
Modified 3-Bends	Multigraph	3	$O(n) \times O(n) \times O(n)$	$n^3 + o(n^3)$	Yes	Wood [38,40]
Diag. Layout & Move.	Simple $\Delta \leqslant 5$	2	$O(n) \times O(n) \times O(n)$	n^3	Yes	Theorem 10

drawings with the vertices positioned in a $O(\sqrt{n}) \times O(\sqrt{n})$ plane grid. Such drawings are inherently orientation-dependent. An $O(n^{3/2})$ volume bound is also achieved by the NON-COLLINEAR Algorithm of Wood [38, Chap. 10], which has the advantage of producing orientation-independent drawings, at the expense of using eight bends for some edge routes. This algorithm produces non-collinear drawings with at most $\lceil\sqrt{n}\rceil$ vertices in each grid-plane.

Eades et al. [22] present a series of refinements of their COMPACT algorithm, referred to as COMPACT 1, COMPACT 2 and COMPACT 3 in Table 1, which explore the tradeoff between the volume and the maximum number of bends per edge route. COMPACT 1 produces drawings in a $O(\sqrt{n}) \times O(\sqrt{n}) \times O(n)$ bounding box with at most six bends per edge; COMPACT 2 produces drawings in a $O(\sqrt{n}) \times O(n) \times O(n)$ bounding box with at most five bends per edge; and COMPACT 3 produces drawings in a $O(n) \times O(n) \times O(n)$ bounding box with at most four bends per edge. The COMPACT and the NON-COLLINEAR algorithms depend on a vertex-colouring of a certain conflict graph to determine the 'height' of edge routes; this step takes $O(n^{3/2})$ time. A method based on edge-colourings by Biedl and Chan [6] reduces the time to $O(n \log n)$.

For drawings with $O(n^2)$ volume, the upper bound of six on the maximum number of bends per edge established by the COMPACT 1 algorithm was improved to five by the STAIRCASE algorithm of Closson et al. [13], and subsequently improved to four by Bield et al. [8]. These algorithms position the vertices along a two-dimensional diagonal, and produces coplanar non-collinear drawings with an $O(n) \times O(n) \times O(1)$ bounding box.

The DYNAMIC STAIRCASE algorithm in [13], also using a two-dimensional diagonal vertex layout, routes each edge vw with at most six bends using arbitrary unused ports at v and w. It follows that this algorithm is fully dynamic; that is, it supports the on-line insertion and deletion of vertices and edges in $O(1)$ time. The DYNAMIC SPIRAL algorithm in [13], which is also fully dynamic, starts with the vertices in an $O(\sqrt{n}) \times O(\sqrt{n})$ plane grid, and then assigns each vertex a unique height in a spiral manner. Thus, the drawings are non-collinear and have a $O(\sqrt{n}) \times O(\sqrt{n}) \times O(n)$ bounding box. At the expense of allowing seven-bend edges, this algorithm produces more orientation-independent drawings than the DYNAMIC STAIRCASE algorithm.

That every 6-graph has a 3-Bend drawing was established by the 3-BENDS algorithm of Eades et al. [21,22] and the INCREMENTAL algorithm of Papakostas and Tollis [31]. The INCREMENTAL algorithm,[2] which supports the on-line insertion of vertices in constant time, produces drawings with $4.63n^3$ volume. The 3-BENDS algorithm produces general position drawings with $27n^3$ volume; by deleting grid-planes not containing a vertex or a bend the volume is reduced to $8n^3$. This algorithm positions the vertices along the main diagonal of a cube in an arbitrary order. By choosing an appropriate ordering of the vertices and using a modified edge routing strategy, the algorithm of Wood [38,40] reduces the volume to $n^3 + o(n^3)$.

Wood [37] shows that every drawing of K_5 has an edge with at least two bends, and it is well known that every drawing of the 2-vertex 6-edge multigraph has an edge

[2] The INCREMENTAL algorithm, as stated in [31], only works for simple graphs, however with a suitable modification it also works for multigraphs [A. Papakostas, private communication, 1998].

with at least three bends (see [38] for a formal proof). Hence the following problem is of considerable interest:

2-*Bends problem*: Does every 6-graph have a 2-bend drawing? [21,22].

Eades et al. [21] conjecture that the answer to this question is false; that is, there exists a 6-graph which does not have a 2-bend drawing. They originally conjectured that K_7 was such a graph, however 2-bend drawings of K_7, along with the other 6-regular complete multi-partite graphs $K_{6,6}$, $K_{3,3,3}$ and $K_{2,2,2,2}$, have since been found by Wood [37,38].

This paper makes the following contributions to the study of three-dimensional orthogonal graph drawings. In Section 2, we describe a framework for generating general position drawings. In particular, we present an algorithm which, given a general position 'drawing' with distinct port assignments but with edge-crossings, produces a crossing-free drawing with no more bends than the original 'drawing'. This framework is used as the basis for the Diagonal Layout & Movement algorithm, presented in Section 3. This algorithm, which produces orientation-independent general position drawings, solves the 2-bends problem in the affirmative for simple 5-graphs; this is the only known non-trivial class of graphs for which 2-bend drawings exist. For simple 6-graphs, the same algorithm uses an average of at most $2\frac{2}{7}$ bends per edge, which is the best known upper bound for the number of bends in three-dimensional orthogonal drawings. Since every general position drawing of K_7 has at least $\frac{16}{7}|E(K_7)|$ bends [39], this upper bound is tight for general position drawings.

Wood [39] also constructs infinite families of 2- and 4-connected 6-graphs with $2m+\Omega(n)$ bends in every general position drawing. Hence the general position model cannot solve the 2-bends problem for graphs with connectivity at most four. Furthermore, it is shown that the natural variation of this model where pairs of vertices share a common grid-plane, cannot solve the 2-bends problem for graphs with connectivity at most two.

2. The general position model

In this section we present a data structure for representing general position drawings; that is, drawings with no two vertices in a common grid-plane. This framework suggests a number of approaches for producing such drawings which we explore in detail. We then describe an algorithm which takes an instance of the data structure and constructs a general position drawing. We are interested in the following problem: given a 6-graph G, determine a general position drawing of G with the minimum number of bends. This problem is NP-hard for bipartite multigraphs [40].

2.1. Representation

Associated with a graph G is the *arc set* $A(G)=\{(v,w),(w,v)\colon \{v,w\}\in E(G)\}$. We often denote the arc (v,w) by $\overrightarrow{vw}$, and $\overrightarrow{vw}$ is called the *reversal* of $\overrightarrow{wv}$. We say an outgoing arc $\overrightarrow{vw}$ incident to a vertex $v\in V(G)$ is *at* v. The port at v used by an edge route vw is referred to as the port *assigned* to the arc $\overrightarrow{vw}$, and is denoted by $port(\overrightarrow{vw})$. We call this mapping from $A(G)$ to the ports of G a *port assignment* for G.

A total ordering $<$ on $V(G)$ induces a numbering $(v_1, v_2, \dots, v_n)$ of $V(G)$ and vice versa. We refer to both $<$ and $(v_1, v_2, \dots, v_n)$ as a *vertex-ordering* of G. We represent a general position drawing of a 6-graph G by the following three-part data structure:

- A (*general position*) *vertex-layout* of G, consisting of three vertex-orderings $<_X$, $<_Y$ and $<_Z$ of G with corresponding numberings $(x_1, x_2, \dots, x_n)$, $(y_1, y_2, \dots, y_n)$ and $(z_1, z_2, \dots, z_n)$, which are called the *X-*, *Y-*, and *Z-orderings*.
- An *arc-colouring* of G, consisting of a 3-colouring of $A(G)$ with colours $\{X, Y, Z\}$ such that for each vertex $v \in V(G)$, there are at most two outgoing arcs at v receiving the same colour.
- An *arc-orientation* of G, defined to be a function $\theta : A(G) \to \{+, -\}$ such that for each vertex $v \in V(G)$ and for each $d \in \{+, -\}$, there are at most three arcs $\overrightarrow{vw}$ at v having $\theta(\overrightarrow{vw}) = d$.

The vertex-layout represents the relative coordinates of the vertices. An arc-colouring and an arc-orientation define the ports used by the edge routes. In particular, if the arc $\overrightarrow{vw}$ is coloured $I \in \{X, Y, Z\}$ and $\theta(\overrightarrow{vw}) = d \in \{+, -\}$, then the General Position Drawing algorithm below will assign port$(\overrightarrow{vw}) = I_v^d$. Since each port can be assigned to at most one arc, we have the following condition which an arc-colouring and arc-orientation must satisfy.

$$\text{For all vertices } v \in V(G), \text{ for all dimensions } I \in \{X, Y, Z\}, \text{ for all } d \in \{+, -\}, \text{ there is at most one outgoing arc } \overrightarrow{vw} \in A(G) \text{ at } v \text{ coloured } I \text{ with } \theta(\overrightarrow{vw}) = d. \tag{1}$$

An arc-colouring and arc-orientation satisfying (1) defines a port assignment for G. Of course, simply assigning ports does not define the full edge routes. The algorithm presented in this section, given a general position vertex-layout and port assignment, determines in linear time a drawing which preserves this layout and port assignment and with the minimum number of bends. By a sequence of port assignment swaps, the algorithm then removes all edge-crossings from the drawing in quadratic time.

The above representation suggests a number of avenues for constructing general position drawings, which differ in the order in which the three components of the representation are determined. For instance, an algorithm which initially determines a vertex-layout, followed by an arc-orientation and arc-colouring, which we call *layout-based*, is used in the 3-Bends algorithm of Eades et al. [21,22], and in the algorithm of Wood [40]. This latter algorithm, given a fixed diagonal general position vertex-layout (that is, the *X*-, *Y*- and *Z*-orderings are equal), determines a drawing with the minimum number of bends in $O(n)$ time. A *routing-based* approach first determines an arc-colouring, then a vertex-layout, and finally an arc-orientation. The routing-based algorithm in [38, Algorithm 5.7] determines a general position drawing with at most $2m + \frac{3}{2}n$ bends and at most $3.375n^3$ volume.

The Diagonal Layout & Movement algorithm presented in this paper, in some sense, combines the layout- and routing-based approaches. First, it computes an initial

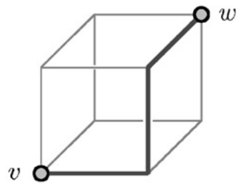

Fig. 2. 2-bend edge route vw.

vertex-layout, then an arc-orientation, and finally an arc-colouring; the arc-colouring also defines the movement of vertices into the final vertex-layout. The following algorithm can form the basis of any algorithm for producing a general position drawing based on the above representation.

Algorithm GENERAL POSITION DRAWING
Input: vertex-layout, arc-colouring and arc-orientation θ of a 6-graph G satisfying (1).
Output: general position drawing of G.

---

(1) Initially position each vertex $v = x_i = y_j = z_k$ at $(3i, 3j, 3k)$.
(2) For each arc $\overrightarrow{vw} \in A(G)$ coloured $I \in \{X, Y, Z\}$, set port$(\overrightarrow{vw}) \leftarrow I_v^{\theta(\overrightarrow{vw})}$.
(3) Apply algorithm EDGE CONSTRUCTION in Section 2.2.
(4) Apply algorithm Crossing Removal in Section 2.3.
(5) Delete each grid-plane not containing a vertex or a bend.

2.2. Constructing edge routes

The following algorithm, for a given port assignment, determines each edge route with the minimum number of bends. For each vertex v and dimension $I \in \{X, Y, Z\}$, we say the I_v^+ (respectively, I_v^-) port *points toward* a vertex w if $v <_I w$ ($w <_I v$); otherwise the port *points away from* w.

Algorithm EDGE CONSTRUCTION
Input: vertex-layout and port assignment of a 6-graph G.
Output: general position 'drawing' of G (possibly with edge-crossings).

For each edge $vw \in E(G)$,
(1) If port$(\overrightarrow{vw})$ is perpendicular to port$(\overrightarrow{wv})$, port$(\overrightarrow{vw})$ points toward w, and port$(\overrightarrow{wv})$ points toward v then route vw with the 2-bend edge route shown in Fig. 2.
(2) If exactly one of port$(\overrightarrow{vw})$ or port$(\overrightarrow{wv})$ points away from w or v, respectively, then, supposing $\overrightarrow{vw}$ does, use the 3-Bend edge route for vw shown in Fig. 3, said to be *anchored* at v.

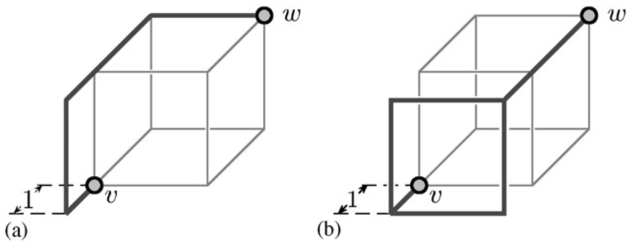

Fig. 3. 3-Bend edge routes anchored at v with (a) perpendicular or (b) parallel ports.

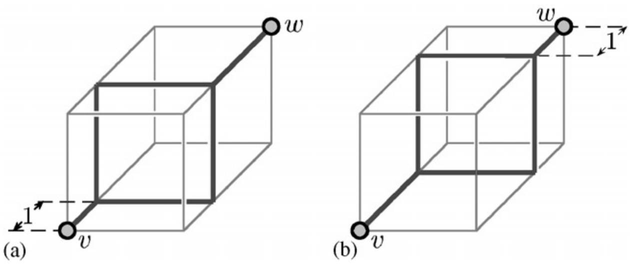

Fig. 4. 3-Bend edge routes anchored at (a) v or (b) w.

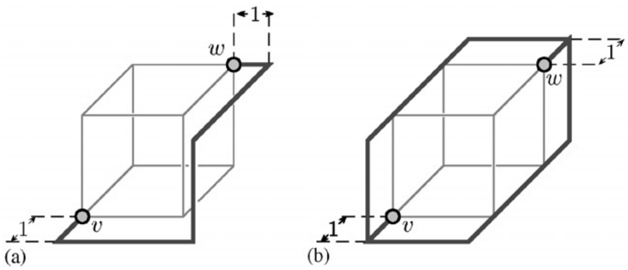

Fig. 5. 4-bend edge routes anchored at v and w with (a) perpendicular or (b) parallel ports.

(3) If port($\overrightarrow{vw}$) points toward w, port($\overrightarrow{wv}$) points toward v, and port($\overrightarrow{vw}$) is parallel to port($\overrightarrow{wv}$), then choose v or w arbitrarily and, route vw with the 3-Bend edge route shown in Fig. 4, said to be *anchored* at the chosen vertex.

(4) If port($\overrightarrow{vw}$) points away from w and port($\overrightarrow{wv}$) points away from v then use the 4-bend edge route for vw shown in Fig. 5, said to be *anchored* at v and at w.

If the edge route vw is anchored at v then we say the arc $\overrightarrow{vw}$ is *anchored*. Since a b-bend edge route contributes $b-2$ anchored arcs, the produced drawings have $2|E(G)|+k$ bends, where k is the number of anchored arcs.

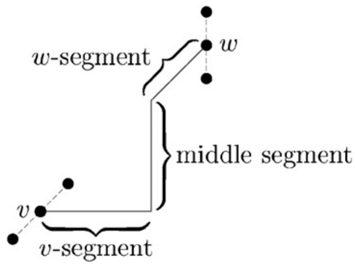

Fig. 6. Segments of the 2-bend component of an edge route.

2.3. Removing edge-crossings

We now describe how to remove the edge-crossings produced by the algorithm Edge Construction. As shown in Fig. 6, each edge route can be considered to consist of a 2-bend edge route possibly with unit-length segments attached at either end. The segments of the 2-bend component of an edge route vw in order from v to w are called the *v-segment*, the *middle segment*, and the *w-segment* of vw.

For a vertex $v = x_i = y_j = z_k$, we say the $(X = 3i - 1)$-plane, the $(X = 3i)$-plane and the $(X = 3i + 1)$-plane *belong* to v, and similarly for the Y- and Z-planes. Note that the middle segment of an edge route vw contains grid-points belonging to v and w and no other vertices. Grid-points contained in the v-segment of vw, except for the grid-point at the intersection of the v-segment of vw and the middle segment of vw, only belong to v.

Suppose in a drawing produced by the Edge Construction algorithm the edge routes vw and xy intersect at some grid-point besides a vertex, called an *edge-crossing*. Then the grid-point of intersection must belong to each of v, w, x and y, which implies that two of these vertices are coplanar. Since the vertices are in general position, two of $\{v, w, x, y\}$ are equal. Hence, intersecting edge routes must be incident to a common vertex. Suppose the edge routes vu and vw intersect at some grid-point other than v. In all edge routes, there are no consecutive unit-length segments, and an edge-crossing involving a unit-length segment must also involve the adjacent non-unit-length segment, thus we need only consider intersections between non-unit-length segments.

In what follows, we characterise the various ways in which vu and vw can intersect. In each case, by swapping the ports assigned to the arcs $\overrightarrow{vu}$ and $\overrightarrow{vw}$, and rerouting the edges vu and vw according to the Edge Construction algorithm the edges no longer cross. However, in some cases doing so may introduce new edge-crossings elsewhere in the drawing. We, therefore, introduce the following potential function as a means of establishing that the algorithm will remove all crossings. Define

$$\Phi = 3n \cdot k + \sum_{vw \in E(G)} \text{length of the middle segment of } vw ,$$

where k is the number of anchored arcs. In each of the following cases, the crossing between vu and vw is removed, and either no new edge-crossings are created or Φ is reduced.

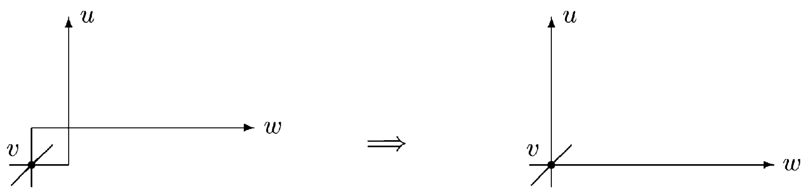

Fig. 7. Rerouting intersecting v-segments.

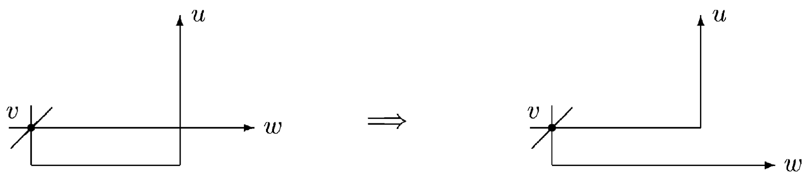

Fig. 8. Rerouting intersecting v-segment of vw and middle segment of vu if $\overrightarrow{vw}$ is not anchored.

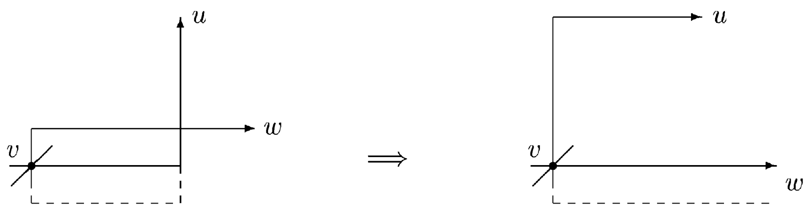

Fig. 9. Rerouting intersecting v-segment of vw and middle segment of vu if $\overrightarrow{vw}$ is anchored.

Case 1—*The v-segments of vu and vw intersect*: Clearly both of $\overrightarrow{vu}$ and $\overrightarrow{vw}$ are anchored, and the intersection is as shown in Fig. 7. Swapping ports eliminates the edge-crossing and introduces no new edge-crossings (and also removes both anchored arcs).

Case 2—*The v-segment of vw intersects the middle segment of vu*:

Case 2(a)—$\overrightarrow{vw}$ *is not anchored*: Clearly $\overrightarrow{vu}$ must be anchored, and the intersection is as shown in Fig. 8. By swapping ports (thus anchoring $\overrightarrow{vw}$ and unanchoring $\overrightarrow{vu}$) the edge-crossing is removed. The new edge routes contain no new grid-points belonging to u or w; thus there are no new edge-crossings introduced.

Case 2(b)—$\overrightarrow{vw}$ *is anchored*: Clearly the intersection is as shown in Fig. 9 ($\overrightarrow{vu}$ may or may not be anchored). By swapping ports the edge-crossing is removed. $\overrightarrow{vu}$ is now not anchored; if $\overrightarrow{vu}$ was anchored then $\overrightarrow{vw}$ is now anchored, and if $\overrightarrow{vu}$ was not anchored then $\overrightarrow{vw}$ is now not anchored. Hence at least one anchored arc is eliminated, and the length of all middle segments except that of vu remains constant. The length of the middle segment of vu increases by no more than $3n$. Hence Φ is reduced.

Case 3—*The middle segments of vu and vw intersect* (see Fig. 10): By swapping ports (and thus swapping any anchors) the edge-crossing is removed. The sum of the lengths of the new middle segments of vu and vw is strictly less than the previous sum (see the segments in bold in Fig. 10), the lengths of the middle segments of all other

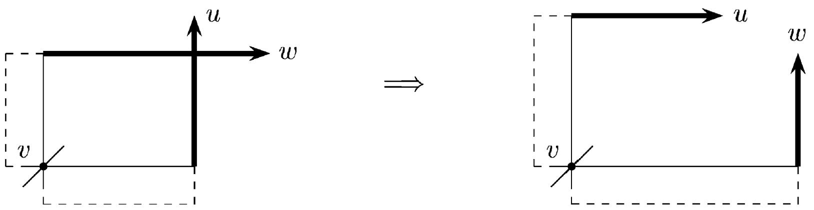

Fig. 10. Rerouting intersecting middle-segments.

edges remains unchanged, and the number of anchored arcs is unchanged. Thus Φ is reduced.

Lemma 1. *There is an* $O(n^2)$ *time* Crossing Removal *algorithm which given a 'drawing' (with edge-crossings) produced by the* Edge Construction *algorithm, produces a (crossing-free) drawing with no more bends than the original.*

Proof. The Crossing Removal algorithm operates in two phases. Consider the following algorithm for the first phase in which only Case 2(b) and Case 3 edge-crossings are eliminated. Initialise $V_{\text{check}} \leftarrow V(G)$. While $V_{\text{check}} \neq \emptyset$ choose a vertex $v \in V_{\text{check}}$; if there is a Case 2(b) or Case (3) crossing between edges $vu, vw \in E(v)$ then swap the ports assigned to $\overrightarrow{vu}$ and $\overrightarrow{vw}$, reroute the edges vu and vw according to the Edge Construction algorithm, and set $V_{\text{check}} \leftarrow V_{\text{check}} \cup \{u, w\}$; otherwise set $V_{\text{check}} \leftarrow V_{\text{check}} \setminus \{v\}$.

Applying Case 2(b) or Case 3 may only create new edge-crossings between uv and some other edge route incident to u, or between wv and some other edge route incident to w. Thus at all times during the first phase, V_{check} contains those vertices v in a Case 2(b) or Case 3 edge-crossing between edges incident to v. Note that the length of a middle segment is at most $3n$, and $k \leqslant 2m$. Thus $\Phi \leqslant 3n \cdot 2m + m \cdot 3n$, and since $m \leqslant 3n$ for 6-graphs, $\Phi \in O(n^2)$. When Case 2(b) or Case 3 is applied Φ is reduced; thus there are $O(n^2)$ Case 2(b) or Case 3 port swaps. With each port swap, three vertices are inserted into V_{check}. Hence there are $O(n^2)$ insertions into V_{check}. Therefore within $O(n^2)$ iterations, $V_{\text{check}} = \emptyset$. At this point, there are no vertices v for which there may be a Case 2(b) or Case 3 edge-crossing between edges incident to v. Therefore all Case 2(b) and Case 3 edge-crossings are removed in the first phase. Checking Case 2(b) and Case 3 for a particular vertex v involves comparing the coordinates of $O(1)$ pairs of edge routes incident to v. Since the rerouting of $O(1)$ edges takes $O(1)$ time, the first phase of the algorithm takes $O(n^2)$ time.

In the second phase, swap the ports assigned to $\overrightarrow{vu}$ and $\overrightarrow{vw}$ for each Case (1) or Case 2(a) edge-crossing. Since doing so does not create any new edge-crossings, all Case 1 and Cases 2(a) edge-crossings are removed. For a particular vertex, Case 1 and Case 2(a) can be checked in constant time. Thus the second phase of the algorithm takes $O(n)$ time. Therefore, the overall algorithm removes all edge-crossings from the drawing in $O(n^2)$ time. In each case, the number of bends is not increased. □

Theorem 2. *Let G be a 6-graph with n vertices and m edges. Given a general position vertex-layout, arc-colouring and arc-orientation of G satisfying* (1), *the* General

Position Drawing *algorithm will, in* O(n^2) *time, construct a* 4-*bend drawing of* G *which preserves the vertex-layout, with at most* $2m + k$ *bends, and with at most* $(n + \frac{1}{3}k)^3$ *volume, where* k *is the number of anchored arcs.*

Proof. The drawings produced before crossings are removed have $2m+k$ bends. Since the Crossing Removal algorithm does not increase the number of bends, the final drawing has at most $2m + k$ bends. The $(X = 3i \pm 1)$-plane belonging to a vertex $v = x_i$ contains a bend if and only if there is an anchored arc $\overrightarrow{vw}$ assigned an X-port (that is, coloured X) with its v-segment lying in this plane. Similarly for Y-planes and Z-planes. Clearly, a grid-plane not containing a vertex or a bend can be removed without affecting the drawing (Step 5). Therefore after this step, the bounding box is $(n + k_X)\times(n + k_Y)\times(n + k_Z)$, where k_I is the number of anchored arcs coloured $I \in \{X, Y, Z\}$. It is well known that of the boxes with fixed sum of side length the cube has maximum volume (see, for example, [25]). Hence the volume is maximised when $k_X = k_Y = k_Z = \frac{1}{3}k$, and thus the volume is at most $(n + \frac{1}{3}k)^3$. By Lemma 1 the Crossing Removal algorithm runs in O(n^2) time; clearly all other steps of General Position Drawing run in O(n) time. □

Corollary 3. *Let* G *be a* 6-*graph with* n *vertices. There is an* O(n^2) *time algorithm which, given a general position 'drawing' of* G *with distinct port assignments but with edge-crossings, produces a drawing* (*without edge-crossings*) *which preserves the vertex-layout and with no more bends than the original.*

Proof. Using the vertex-layout and port assignment defined by the original 'drawing', construct an intermediate 'drawing' of G (possibly with edge-crossings) using the Edge Construction algorithm. Since these new edge routes have the minimum number of bends for a fixed port assignment, the intermediate 'drawing' has no more bends than the original. Applying the Crossing Removal algorithm, we obtain a crossing-free drawing of G with no more bends than the original. □

3. Diagonal Layout & Movement algorithm

In this section we describe the Diagonal Layout & Movement algorithm for three-dimensional orthogonal graph drawing. Initially, the vertices are placed along the main diagonal of a cube, and an arc-orientation followed by an arc-colouring is determined. This colouring also defines the movement of vertices away from the diagonal.

We first introduce some notation. Let $<$ be a vertex-ordering of a graph G. For each edge $vw \in E(G)$ with $v < w$, we say the arc $\overrightarrow{vw} \in A(G)$ is a *successor arc* of v, and w is a *successor* of v; similarly the arc $\overrightarrow{wv} \in A(G)$ is a *predecessor arc* of w, and v is a *predecessor* of w. For each vertex $v \in V(G)$, the number of successor and predecessor arcs of v are denoted succ(v) and pred(v), respectively.

As shown in Fig. 11, we say a vertex v is an (α, β)-vertex, where $\alpha = \min\{\text{pred}(v), \text{succ}(v)\}$ and $\beta = \max\{\text{pred}(v), \text{succ}(v)\}$; we say (α, β) is the *type* of v. v is *positive* if succ(v) > pred(v) and *negative* if pred(v) > succ(v). For positive vertices v and for

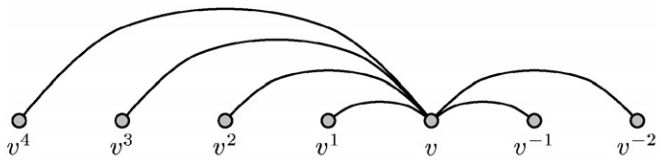

Fig. 11. v is a negative $(2,4)$-vertex.

$k>0$ (respectively, $k<0$), v^k denotes the $|k|$th successor (predecessor) of v to the right (left) of v in the ordering. For negative v and for $k>0$ (respectively, $k<0$), v^k denotes the $|k|$th predecessor (successor) of v to the left (right) of v in the ordering. For vertices v with $\text{succ}(v)=\text{pred}(v)$, it is convenient for v^k to denote the same vertex as if v was positive.

Algorithm Diagonal Layout & Movement
Input: 6-graph G.
Output: General position drawing of G.

---

(1) Initialise each of the X-, Y- and Z-orderings of a general position vertex-layout to be the vertex-ordering $<$ of G determined by the Balanced Ordering algorithm in Section 3.1.
(2) For each vertex $v\in V(G)$, depending on the type of v, label some of the arcs at v as *movement* or *special* arcs, according to Table 2.
(3) Determine a port assignment for G with the Port Assignment algorithm described in Section 3.2.
(4) For each movement arc $\overrightarrow{vw}$ coloured $I\in\{X,Y,Z\}$, move v to immediately past w in the I-ordering.
(5) Apply General Position Drawing algorithm.

The general strategy of the Diagonal Layout & Movement algorithm is to anchor at most one arc at each vertex. The port at a vertex assigned to an unanchored arc must point toward its destination. Thus, in the initial diagonal layout, there are three positive ports which can be assigned to unanchored successor arcs, and three negative ports which can be assigned to unanchored predecessor arcs. Thus, for a vertex v with $\max\{\text{succ}(v),\text{pred}(v)\}\leqslant 3$, all of the arcs at v need not be anchored (without moving v). If $\text{succ}(v)>3$ then the positive ports at v can be assigned to at most three unanchored successor arcs of v. The remaining successor arcs must be assigned a negative port at v. These are precisely the *movement* and *special* arcs defined in Table 2. If $\overrightarrow{vw}$ is a movement arc coloured I, then v is moved to immediately past w in the I-ordering (Step 4 of the algorithm). Thus the I_v^- port points toward w, which allows for $\overrightarrow{vw}$ to be assigned the I_v^- port without anchoring $\overrightarrow{vw}$. We shall prove later that it is precisely the special arcs which become anchored when the General Position Drawing algorithm is applied. Note that there is at most one special arc at

Table 2
Definition of movement and special arcs at a vertex v

Type of v	(0,4)	(1,4)	(0,5)	(2,4)	(1,5)	(0,6)
(v, v^1)	Movement	Movement	Movement	Special	Movement	Movement
(v, v^2)	—	—	Movement	—	Special	Movement
(v, v^3)	—	—	—	—	—	Special

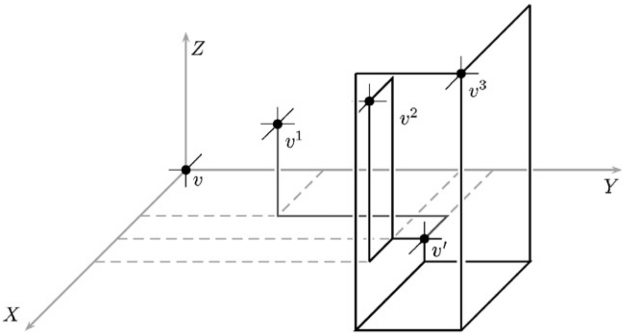

Fig. 12. v is a positive (0,6)-vertex, (v, v^1) is a movement arc coloured X, (v, v^2) is a movement arc coloured Y, (v, v^3) is a special arc coloured Z which becomes anchored; move v to v'.

each degree six vertex, and no special arcs at each vertex with degree at most five. In Fig. 12, we illustrate the movement and anchoring process in the case of a positive (0,6)-vertex.

3.1. Determining the initial vertex-layout

In this section we describe an algorithm for determining a vertex-ordering which is used by the Diagonal Layout & Movement algorithm to specify the initial vertex-layout along the main diagonal of a cube. This method may be of general interest; thus we describe it for graphs of arbitrary degree.

We shall see that the initial vertex-ordering used in the Diagonal Layout & Movement algorithm needs to satisfy the following two properties. First, the vertex-ordering should be 'balanced', meaning at each vertex the number of predecessors should be as equal as possible to the number of successors. The function $\sum_v |\text{succ}(v) - \text{pred}(v)|$ measures how well 'balanced' a vertex-ordering is. However, minimising $\sum_v |\text{succ}(v) - \text{pred}(v)|$ is NP-hard, and remains NP-hard for bipartite 6-graphs [7]. Note that a vertex v has even $|\text{succ}(v) - \text{pred}(v)|$ if and only if v has even degree, and hence $|\text{succ}(v) - \text{pred}(v)|$ for an odd degree vertex v is at least one. We, therefore, say a vertex v is *balanced* if $|\text{succ}(v) - \text{pred}(v)| \leqslant 1$. The second desirable property is to have many balanced vertices. To reflect these two features, we define the *imbalance*

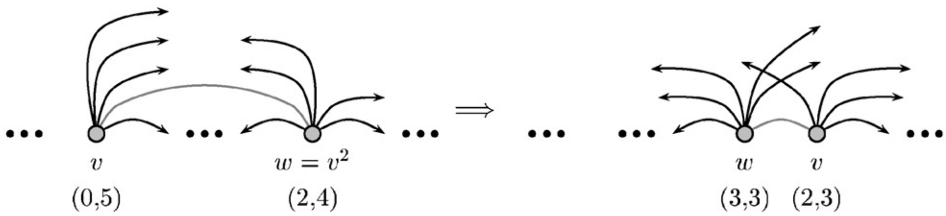

Fig. 13. **M1** for a positive $(0,5)$-vertex v and a negative $(2,4)$-vertex $w=v^2$.

of a vertex v in a vertex-ordering of a graph with maximum degree Δ to be

$$\phi(v) = \Delta|\mathrm{succ}(v) - \mathrm{pred}(v)| - \begin{cases} 1 & \text{if } v \text{ is balanced,} \\ 0 & \text{otherwise.} \end{cases}$$

The *total imbalance* of a vertex-ordering is the sum of the imbalance of all the vertices. The approach we take to determine a vertex-ordering uses the total imbalance as a potential function. Starting with an arbitrary vertex-ordering, we apply a number of rules for moving vertices within the ordering, each of which reduce the total imbalance. We continue until a vertex-ordering is determined in which none of the rules are applicable. The proof of the following introductory result is left as an exercise for the reader.

Lemma 4. *If a positive (respectively, negative) vertex v gains i predecessors (successors) and loses i successors (predecessors) for some i with $1 \leqslant i \leqslant \lfloor \frac{1}{2}|\mathrm{succ}(v) - \mathrm{pred}(v)| \rfloor$, then $|\mathrm{succ}(v) - \mathrm{pred}(v)|$ is reduced by $2i$.*

Consider the following rule, which takes as input an arc $\overrightarrow{vw}$, for moving a vertex v in a vertex-ordering. Two adjacent vertices v, w with $v < w$ are *opposite* if v is positive and w is negative.

M1($\overrightarrow{vw}$): If $w = v^i$ is opposite to v for some i with $1 \leqslant i \leqslant \lfloor \frac{1}{2}|\mathrm{succ}(v) - \mathrm{pred}(v)| \rfloor$, then move v to immediately past w, as shown in Fig. 13.

Suppose **M1** is applied. Assume without loss of generality that v is positive. By moving v, v loses i successors and gains i predecessors; thus by Lemma 4, $\phi(v)$ is reduced by at least $2i\Delta$. For each k, $1 \leqslant k \leqslant i-1$, $|\mathrm{succ}(v^k) - \mathrm{pred}(v^k)|$ is increased by at most two and since v^k may become unbalanced, $\phi(v^k)$ is increased by at most $2\Delta + 1$. Since w is opposite to v, $\phi(w)$ does not increase. The number of successors and predecessors, and hence the imbalance, of all other vertices remains unchanged. Thus, the total imbalance decreases by at least $2i\Delta - (i-1)(2\Delta+1) = 2\Delta - i + 1$, which is positive since $i \leqslant \frac{1}{2}\Delta$. Hence the total imbalance decreases when **M1** is applied.

We now present rules **M2** and **M3**, which take as input an edge vw, for moving the vertices v and w in a vertex-ordering.

M2(vw): If v is opposite to w and $v < w^j < v^i < w$ for some i with $1 \leqslant i \leqslant \lfloor \frac{1}{2}|\mathrm{succ}(v) - \mathrm{pred}(v)| \rfloor$ and j with $1 \leqslant j \leqslant \lfloor \frac{1}{2}|\mathrm{succ}(w) - \mathrm{pred}(w)| \rfloor$, then move v up to v^i and move w up to w^j, as shown in Fig. 14.

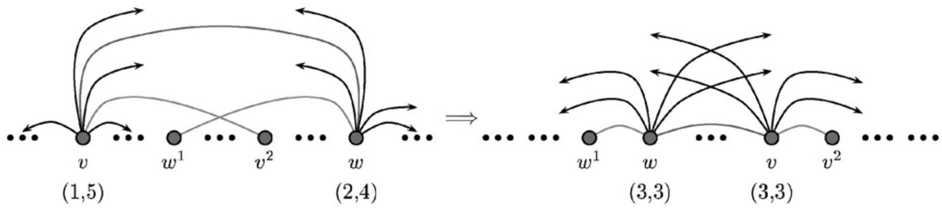

Fig. 14. **M2** for a positive (1,5)-vertex v and a negative (2,4)-vertex w.

By moving v and w under rule **M2**, v loses i successors and gains i predecessors; thus by Lemma 4, $\phi(v)$ is reduced by at least $2i\Delta$. For each k, $1 \leqslant k \leqslant i-1$, $|\text{succ}(v^k) - \text{pred}(v^k)|$ is increased by at most two; since v^k may become unbalanced, $\phi(v^k)$ is increased by at most $2\Delta + 1$. w loses j successors and gains j predecessors, hence $\phi(w)$ is reduced by at least $2j\Delta$. For each k, $1 \leqslant k \leqslant j-1$, $|\text{succ}(w^k) - \text{pred}(w^k)|$ is increased by at most two; since w^k may become unbalanced, $\phi(w^k)$ is increased by at most $2\Delta+1$. The number of successors and predecessors, and hence the imbalance, of all other vertices remains unchanged. Thus, the total imbalance decreases by at least $2i\Delta + 2j\Delta - (i-1)(2\Delta+1) - (j-1)(2\Delta+1) = 4\Delta - i - j + 2$, which is positive since $i, j \leqslant \frac{1}{2}\Delta$. Hence the total imbalance decreases when **M2** is applied.

M3(vw): If v is opposite to w and $v < v^i = w^j < w$ for some i with $1 \leqslant i \leqslant \lfloor \frac{1}{2}(|\text{succ}(v) - \text{pred}(v)| - 1) \rfloor$ and j with $1 \leqslant j \leqslant \lfloor \frac{1}{2}(|\text{succ}(w) - \text{pred}(w)| - 1)$, then move v to immediately past v^i and move w to immediately past w^j, as shown in Fig. 15.

By moving v and w under rule **M3**, v loses $i+1$ successors and gains $i+1$ predecessors. If $\deg(v)$ is odd and $i = \lfloor \frac{1}{2}(|\text{succ}(v) - \text{pred}(v)| - 1 \rfloor$, then v becomes a $(\frac{1}{2}(\deg(v) - 1), \frac{1}{2}(\deg(v) + 1))$ vertex; hence $|\text{succ}(v) - \text{pred}(v)| = 1$ and $\phi(v)$ has been reduced by $2i\Delta + 1$. Otherwise, it is easily seen that $\text{succ}(v) - \text{pred}(v) \geqslant 0$, $|\text{succ}(v) - \text{pred}(v)|$ is reduced by $2(i+1)$, and hence $\phi(v)$ is reduced by at least $2(i+1)\Delta$. In either case, $\phi(v)$ is reduced by at least $2i\Delta + 1$. For each k, $1 \leqslant k \leqslant i-1$, $|\text{succ}(v^k) - \text{pred}(v^k)|$ is increased by at most two, and since v^k may become unbalanced, $\phi(v^k)$ is increased by at most $2\Delta + 1$. Analogous to the case of v, $\phi(w)$ is reduced by at least $2j\Delta + 1$. For each k, $1 \leqslant k \leqslant j-1$, $|\text{succ}(w^k) - \text{pred}(w^k)|$ is increased by at most two, and since w^k may become unbalanced, $\phi(w^k)$ is increased by at most $2\Delta + 1$. The number of successors and predecessors, and hence the imbalance, of all other vertices remains unchanged. Thus, the total imbalance decreases by at least $2i\Delta + 1 + 2j\Delta + 1 - (i-1)(2\Delta+1) - (j-1)(2\Delta+1) = 4\Delta - i - j + 4$, which is positive since $i, j \leqslant \frac{1}{2}\Delta$. Hence the total imbalance decreases when **M3** is applied.

Our final rule decreases the number of unbalanced vertices with maximum degree. Papakostas and Tollis [30] employ a similar idea to produce so-called *bst*-orderings of 4-graphs.

M4(v): If $\deg(v) = \Delta$ and v^i is unbalanced for all i with $1 \leqslant i \leqslant \lfloor \frac{1}{2}|\text{succ}(v) - \text{pred}(v)| \rfloor$, then move v to immediately past $v^{\lfloor \frac{1}{2}|\text{succ}(v) - \text{pred}(v)| \rfloor}$, as shown in Fig. 16.

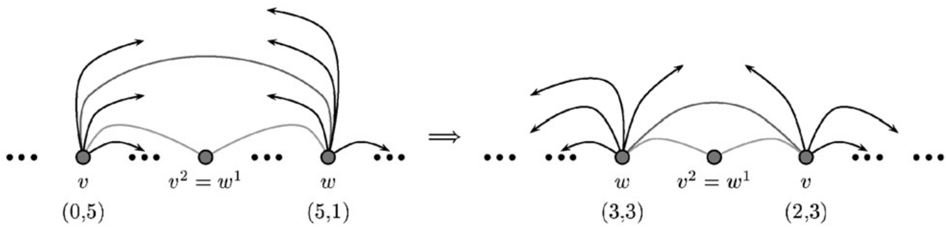

Fig. 15. **M3** for a positive $(0,5)$-vertex v and a negative $(1,5)$-vertex w.

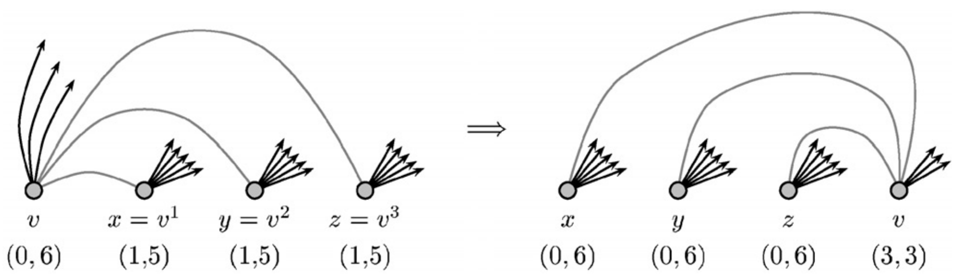

Fig. 16. **M4** for a $(0,6)$-vertex v.

If **M4** is applied, v becomes balanced, and thus by Lemma 4, $\phi(v)$ is reduced by $2\Delta\lfloor\frac{1}{2}|\text{succ}(v) - \text{pred}(v)|\rfloor + 1$. For each i $(1 \leqslant i \leqslant \lfloor\frac{1}{2}|\text{succ}(v) - \text{pred}(v)|\rfloor)$, $|\text{succ}(v^i) - \text{pred}(v^i)|$ increases by at most two, and since v^i was not balanced beforehand, $\phi(v^i)$ increases by at most 2Δ. The imbalance of all other vertices remains unchanged. Hence, the total imbalance is reduced by at least $2\Delta\lfloor\frac{1}{2}|\text{succ}(v)-\text{pred}(v)|\rfloor+1-2\Delta\lfloor\frac{1}{2}|\text{succ}(v)-\text{pred}(v)|\rfloor = 1$ whenever **M4** is applied.

Lemma 5. *A vertex-ordering of an n-vertex graph with maximum degree Δ in which* **M4** *cannot be applied has at most $\Delta/(\Delta + 1)n$ unbalanced vertices with maximum degree.*

Proof. Let V_b be the set of balanced vertices, and let V_u be the set of unbalanced vertices with degree Δ in a vertex-ordering in which **M4** is not applicable. If $v \in V_u$ then v must have a neighbour $w \in V_b$; in this case we say the arc $\overrightarrow{vw}$ is a *balancing arc*. A vertex $w \in V_b$ can have at most $\deg(w)$ incoming balancing arcs. Hence $|V_u| \leqslant \sum_{w \in V_b} \deg(w)$. Let δ be the average degree of the balanced vertices. Then $|V_u| \leqslant \delta|V_b|$, implying $(1 + \delta)|V_u| \leqslant \delta(|V_u| + |V_b|) \leqslant \delta n$, and hence $|V_u| \leqslant \delta/(\delta + 1)n \leqslant \Delta/(\Delta + 1)n$ since $\delta \leqslant \Delta$. □

Note that the focus on vertices of maximum degree in **M4** is not necessary. It is easily seen that a vertex-ordering of a graph with maximum degree Δ can be determined with at most $\Delta/(\Delta + 1)n$ unbalanced vertices (regardless of their degree).

Since applying each of **M1**–**M4** decreases the total imbalance, an algorithm which repeatedly attempts to apply **M1**–**M4** until they are not applicable will terminate. By maintaining the set of edges for which the rules need to be checked, the following algorithm which we call Balanced Ordering efficiently determines a vertex-ordering in which **M1**–**M4** are not applicable. Start with an arbitrary vertex-ordering of G and let $E_{\text{check}} \leftarrow E(G)$. While $E_{\text{check}} \neq \emptyset$ choose an edge $vw \in E_{\text{check}}$, and if one of **M1**($\overrightarrow{vw}$), **M1**($\overrightarrow{wv}$), **M2**(vw), **M3**(vw), **M4**(v) or **M4**(w), is applicable then do so, and set $E_{\text{check}} \leftarrow E_{\text{check}} \cup E(x)$ for all $x \in V(v) \cup V(w)$; otherwise set $E_{\text{check}} \leftarrow E_{\text{check}} \setminus \{vw\}$.

Lemma 6. *Given an m-edge graph G with maximum degree Δ, the* Balanced Ordering *algorithm determines, in* $O(\Delta^4 m)$ *time, a vertex-ordering of G in which* **M1**–**M4** *are not applicable.*

Proof. We shall prove that at all times the set E_{check} contains all edges in $E(G)$ for which one of the rules is possibly applicable. At the start of the algorithm this is true, since $E_{\text{check}} = E(G)$. Consider an adjacency list representation of G where each adjacency list is ordered according to the current vertex-ordering. Suppose the edge vw is chosen from E_{check}. If **M1**($\overrightarrow{vw}$), **M1**($\overrightarrow{wv}$), **M2**(vw), **M3**(vw), **M4**(v) and **M4**(w), are not applied then $E_{\text{check}} \setminus \{vw\}$ contains all edges in $E(G)$ for which any of **M1**–**M4** are possibly applicable.

Suppose a rule is applied so that a vertex v moves in the current vertex-ordering. The only vertices whose imbalance may change are v and its neighbours, and only the adjacency lists of v and its neighbours are changed. For an edge $pq \in E(G)$, where p and q are both not adjacent to v or one of the neighbours of v, the adjacency lists of p and q do not change, and the imbalance of every vertex adjacent to p or q does not change. Hence, if **M1**($\overrightarrow{pq}$), **M1**($\overrightarrow{qp}$), **M2**(pq), **M3**(pq), **M4**(p) and **M4**(q) are not applicable before moving v, then they will not be applicable after moving v.

Therefore, by adding to E_{check} the edges in $E(x)$ for each neighbour x of v (whenever a vertex v is moved), we maintain the condition that E_{check} contains all edges in $E(G)$ for which any of **M1**–**M4** are possibly applicable. The algorithm continues until $E_{\text{check}} = \emptyset$, at which point there are no edges for which any of **M1**–**M4** are applicable.

The total imbalance of a vertex-ordering is at most $\Delta \sum_v \deg(v) = 2\Delta m$. As proved above, each of the rules **M1**–**M4** reduce the total imbalance; thus a rule is applied at most $2\Delta m$ times. Whenever a rule is applied, $O(\Delta^2)$ edges are added to E_{check}. Hence the algorithm inserts $O(\Delta^2 \cdot \Delta m)$ edges into E_{check}, and therefore a rule is checked $O(\Delta^3 m)$ times.

Using the order maintenance algorithm of Dietz and Sleator [17], the vertex-ordering and adjacency lists of each vertex can be maintained in constant time under the move operation. Hence each rule can be checked in $O(\Delta)$ time; thus the algorithm runs in $O(\Delta^4 m)$ time. □

Since the initial vertex-ordering in the Diagonal Layout & Movement algorithm is determined by the Balanced Ordering algorithm, we have the following observations.

Lemma 7. *If $\overrightarrow{vw}$ is a movement or special arc then $\overrightarrow{wv}$ is not a movement or special arc.*

Proof. Let $\overrightarrow{vw}=(v,v^k)$. Then $k \leqslant \lfloor \frac{1}{2}\text{succ}(v) - \text{pred}(v)| \rfloor$ (see Table 2); thus **M1** is applicable to $\overrightarrow{vw}$ if v is opposite to w. Since **M1** is not applicable, w is not opposite to v in the initial vertex-ordering. Since movement and special arcs are successor (respectively, predecessor) arcs for positive (negative) vertices, the result follows. □

Lemma 8. *If v and w are opposite vertices in the initial vertex-ordering, then the movement arcs of v do not 'cross over' or have the same destination vertex as the movement arcs of w.*

Proof. Let (v,v^p) be the movement arc (if any) at v with maximum p, and let (w,w^q) be the movement arc (if any) at w with maximum q. Then $p \leqslant \lfloor \frac{1}{2}(|\text{succ}(v)-\text{pred}(v)|-1)\rfloor$ and $q \leqslant \lfloor \frac{1}{2}(|\text{succ}(w)-\text{pred}(w)|-1 \rfloor$ (see Table 2). Thus, the rule **M2** is applicable to the edge vw if $v<w^q<v^p<w$. Since **M2** is not applicable in the initial vertex-ordering, the movement arcs of v do not 'cross over' the movement arcs of w. Similarly, **M3** is applicable to the edge vw if $v<w^q=v^p<w$. Since **M3** is not applicable in the initial vertex-ordering, the movement arcs of v do not have the same destination vertex as the movement arcs of w. □

3.2. Determining a port assignment

This section describes how to compute a port assignment in Step 3 of the Diagonal Layout & Movement algorithm. To determine an arc-colouring we vertex-colour a graph H with vertex set $V(H)=A(G)$. Since the colour assigned to an arc determines the dimension of its port assignment, vertices are adjacent in H if the corresponding arcs must use perpendicular ports.

Algorithm Port Assignment
Input: • balanced vertex-ordering of 6-graph G
• classification of movement and special arcs.
Output: arc-colouring and arc-orientation of G satisfying (1).

---

(1) For each vertex $v \in V(G)$, Table 3 defines the nodes $v^A, v^B, \ldots, v^F$ depending on the type of v. (Note that (v,v^C) is a special arc for each unbalanced degree six vertex v, and only (v,v^A) and (v,v^B) are possibly movement arcs.)
(2) Define an arc-orientation as follows. For each non-negative (respectively, negative) vertex $v \in V(G)$ and for each $Q \in \{A,B,C\}$ set $\theta((v,v^Q)) = -(+)$, and for each $Q \in \{D,E,F\}$ set $\theta((v,v^Q)) = +(-)$; that is, the arcs (v,v^A), (v,v^B) and (v,v^C) will use negative (positive) ports at v, and the arcs (v,v^D), (v,v^E) and (v,v^F) will use positive (negative) ports at v.

(3) Construct a graph H with vertex set $V(H)=A(G)$. The vertices of H are referred to by the corresponding arc in $A(G)$. We distinguish four types of edges of H as follows.
   (a) For each vertex $v\in V(G)$, add cliques $\{(v,v^A),(v,v^B),(v,v^C)\}$ and $\{(v,v^D),(v,v^E),(v,v^F)\}$ to $E(H)$; these edges are said to be *unlabelled*. (These edges ensure that arcs which 'compete' for the same ports are coloured differently. Note that if $\deg(v)<6$ then these cliques may be empty or consist of a single edge.)
   (b) For each edge $vw\in E(G)$, if neither the arc $\overrightarrow{vw}$ nor its reversal arc $\overrightarrow{wv}$ are special then add the edge $\{\overrightarrow{vw},\overrightarrow{wv}\}$, called an *r*-edge, to $E(H)$. (This edge enables vw to have a 2-bend edge, which must have perpendicular ports.)
   (c) If $\overrightarrow{vw}$ and $\overrightarrow{wx}$ are both movement arcs for some vertices v, w and x, then add the edge $\{\overrightarrow{vw},\overrightarrow{wx}\}$, called a $*$-edge, to $E(H)$. (This ensures that v and w do not move in the same ordering.)
   (d) If v is a $(0,6)$-vertex or a $(0,5)$-vertex add the edge $\{(v,v^2),(v^1,v)\}$, called a $**$-edge, to $E(H)$. (In this case, (v,v^2) is a movement arc. If (v,v^2) is coloured I then v will move past v^1 in the I-ordering; the edge $\{(v,v^2),(v^1,v)\}$ ensures that (v^1,v) does not use an incorrect I-port at v^1. For example, in Fig. 12, (v^1,v) cannot use the $Y^+_{v_1}$ port.)

(4) As described in Lemma 9 below, repeatedly remove vertices from H with degree at most two (and their incident edges), and merge non-adjacent vertices $v,w\in V(H)$ in a $K_4\backslash\{vw\}$ subgraph of H, and replace any resulting parallel edges by a single edge.
(5) Determine a proper vertex 3-colouring of H with colours $\{X,Y,Z\}$ (see Lemma 9 below). If two vertices of H have been merged then we consider both vertices to have received the same colour as the merged vertex.
(6) Colour the removed vertices $v\in V(H)$ in reverse order of their removal with a colour in $\{X,Y,Z\}$ different from the ($\leqslant 2$) neighbours of v in H; insert v and its removed incident edges back into H.
(7) Determine an arc-colouring of G by colouring each arc $\overrightarrow{vw}\in A(G)$ with the colour assigned to the corresponding vertex of H.

Lemma 9. *The graph H is vertex 3-colourable in $O(n)$ time.*

Proof. If $K_4\backslash\{vw\}$ is a subgraph of H for some non-adjacent vertices v and w, then in any proper vertex 3-colouring of H, v and w must receive the same colour, thus merging these vertices preserves the 3-colourability of H. We now show that after repeatedly removing vertices with degree at most two, and merging pairs of vertices in a $K_4\backslash\{vw\}$ subgraph, H has maximum degree three, and is not K_4, thus by Brooks' Theorem [11], H is 3-colourable.

For all vertices $v\in V(G)$, let H_v be the subgraph of H consisting of the vertices $(v,v^A),(v,v^B)$ and (v,v^C) and their incident edges. We now show that H_v 'reduces' to a maximum degree three subgraph.

Table 3
Definition of v^A, v^B, v^C, v^D, v^E and v^F

v	v^A	v^B	v^C	v^D	v^E	v^F
$(\leqslant 3, \leqslant 3)$-vertex	v^{-3}	v^{-2}	v^{-1}	v^1	v^2	v^3
(0,4)-vertex	v^1	—	—	v^2	v^3	v^4
(1,4)-vertex	v^{-1}	v^1	—	v^2	v^3	v^4
(2,4)-vertex	v^{-2}	v^{-1}	v^1	v^2	v^3	v^4
(0,5)-vertex	v^1	v^2	—	v^3	v^4	v^5
(1,5)-vertex	v^{-1}	v^1	v^2	v^3	v^4	v^5
(0,6)-vertex	v^1	v^2	v^3	v^4	v^5	v^6

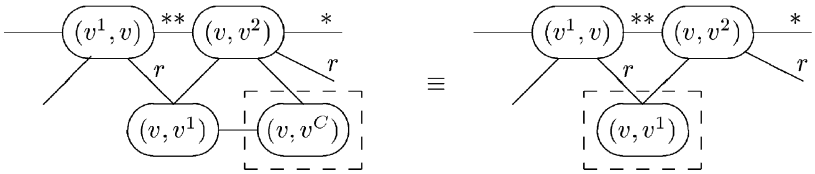

Fig. 17. The subgraph H_v for a (0,5)-vertex or a (0,6)-vertex v with v^1 balanced.

For a degree six unbalanced vertex v, the vertex of H corresponding to the special arc (v, v^C) is incident with at most two (unlabelled) edges, and therefore can be removed from H. Since a (0,6)-vertex and a (0,5)-vertex v both have (v, v^A) and (v, v^B) as movement arcs, H_v is the same for a (0,6)-vertex v (after removing (v, v^C)) and for a (0,5)-vertex v (see Figs. 17 and 18). Similarly, for (1,5)- and (2,4)-vertices, H_v is the same as for (1,4)- and (2,3)-vertices, respectively. That is, the result for graphs with unbalanced degree six vertices in the vertex-ordering reduces to the result for vertex-orderings without such vertices. We therefore need only consider (0,5)-, (1,4)- and (0,4)-vertices.

Consider a (0,5)-vertex v. v^1 may be balanced or a (1,4)-vertex. If v^1 is balanced then, as shown in Fig. 17, (v, v^1) has degree two and can be removed. In the remaining graph, (v, v^2) and (v^1, v) have degree three.

Now, if v^1 is a (1,4)-vertex then, as shown in Fig. 18, (v, v^2) and $(v^1, (v^1)^1)$ are the non-adjacent vertices in a $K_4 \backslash \{e\}$ subgraph. If we merge these vertices then (v^1, v) and (v, v^1) have degree two and can be removed. If v^2 is balanced then there is no edge $\{(v, v^2), (v^2, (v^2)^1\}$. If v^2 is unbalanced then v^2 must be a (1,4)-vertex, and therefore (v^2, v) and the r-edge $\{(v, v^2), (v^2, v)\}$ will be removed (see Fig. 19). In either case (v, v^2) $(= (v^1, (v^1)^1))$ has degree three.

Consider a (1,4)-vertex v, and assume that v^{-1} is not a (0,5)-vertex with $(v^{-1})^1 = v$ (we have already considered this case). As shown in Fig. 19, (v, v^{-1}) has degree two and can be removed. (v, v^1) now has degree at most three. For a (0,4)-vertex v, H_v simply consists of the degree one vertex (v, v^1), which can be removed.

Consider a vertex $(v, v^Q) \in V(H)$ for some $Q \in \{D, E, F\}$, or $Q \in \{A, B, C\}$ if v is balanced. (v, v^Q) is incident with at most two unlabelled edges and to at most one

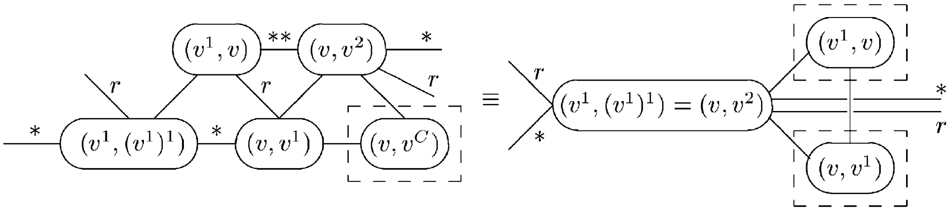

Fig. 18. The subgraph H_v for a (0,5)-vertex or a (0,6)-vertex v with v^1 a (1,4)-vertex.

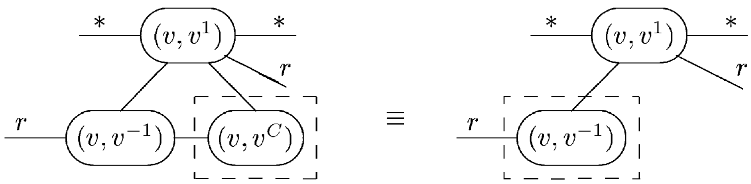

Fig. 19. The subgraph H_v of H for a (1,4)-vertex or a (1,5)-vertex v.

r-edge. Unless v^Q is a (0,5)- or (0,6)-vertex and $(v^Q)^1 = v$ (in which case (v, v^Q) is incident with a $**$-edge, and has already been considered), (v, v^Q) has degree at most three.

We have shown that all remaining vertices in H have degree at most three, and trivially $H \neq K_4$; thus by Brooks' Theorem [11], H is 3-colourable. Using the above case-analysis, the $K_4 \setminus \{vw\}$ subgraphs and vertices with degree at most 2 can be identified in $O(n)$ time. The proof of Brooks' Theorem due to Lovász [27] leads to an algorithm for computing a vertex 3-colouring of H in $O(|V(H)| + |E(H)| \subseteq O(n)$ time [3]. □

Theorem 10. *Given a* 6-*graph* G, *the* DIAGONAL LAYOUT & MOVEMENT *algorithm will determine, in* $O(n^2)$ *time, a* 4-*bend drawing of* G *with volume at most* $2.13n^3$ *and an average of at most* $2\frac{2}{7}$ *bends per edge. If* G *is a* 5-*graph then the volume is* n^3 *and each edge has two bends.*

Proof. We first prove the correctness of the algorithm. To do so, we prove the correctness of the PORT ASSIGNMENT algorithm; that is, it determines an arc-orientation and an arc-colouring of G satisfying (1). By Lemma 9, H is vertex 3-colourable. At a non-negative (respectively, negative) vertex v, the arcs (v, v^A), (v, v^B) and (v, v^C) have orientation $-(+)$ (see Step 2 of PORT ASSIGNMENT), and are pairwise coloured differently due to the unlabelled edges in H. Similarly, the arcs (v, v^D), (v, v^E) and (v, v^F) have orientation $+(-)$, and are pairwise coloured differently. Hence, at most three outgoing arcs $\overrightarrow{vw}$ incident to a vertex v have equal orientation; thus θ is an arc-orientation of G. At most two outgoing arcs incident to v receive the same colour; thus we have an arc-colouring of G. It follows that for all colours $I \in \{X, Y, Z\}$ and for all $d \in \{+,-\}$, there is at most one outgoing arc $\overrightarrow{vw} \in A(G)$ at v coloured I with $\theta(\overrightarrow{vw}) = d$. Thus, the arc-orientation and arc-colouring satisfy (1). Therefore, the conditions for the application of the General Position Drawing algorithm are satisfied when this algorithm is called in Step 5 of Diagonal Layout & Movement.

If a vertex v has two outgoing movement arcs then v is either a (0,5)- or (0,6)-vertex, and the two movement arcs are (v, v^A) and (v, v^B). Since $\{(v, v^A), (v, v^B)\}$ is an (unlabelled) edge of H, these arcs will be coloured differently. Hence in Step 4 of the Diagonal Layout & Movement algorithm, v will move within each of the orderings corresponding to the colours of these movement arcs. Therefore Step 4, and the entire algorithm, is valid.

We now prove it is precisely the special arcs which become anchored when the General Position Drawing algorithm is applied in Step 5 of the Diagonal Layout & Movement algorithm. Consider an arc $\overrightarrow{vw}$ coloured $I \in \{X, Y, Z\}$ with its reversal arc $\overrightarrow{wv}$ coloured $J \in \{X, Y, Z\}$. Without loss of generality the vertex v is non-negative.

Suppose $\overrightarrow{vw}$ is special. Then $w = v^C$, and in the initial vertex-ordering, $v < w$. The only possible movement arcs at v are (v, v^A) and (v, v^B). Since $\{(v, v^A), (v, v^C)\}$ and $\{(v, v^B), (v, v^C)\}$ are edges in H, v will not move in the I-ordering, and w does not move past v in any ordering as otherwise **M1**$(\overrightarrow{wv})$ would be applicable. Hence in the final I-ordering, $v <_I w$. Step 2 of the Port Assignment algorithm defines $\theta(\overrightarrow{vw}) = -$. Therefore $\text{port}(\overrightarrow{vw}) = I_v^-$, which points away from w. Thus, the Edge Construction algorithm anchors $\overrightarrow{vw}$.

We now show that arcs which are not special do not become anchored when the Edge Construction algorithm is applied. Suppose $\overrightarrow{vw}$ is not special. We consider the following three cases.

Case 1. *$\overrightarrow{vw}$ is a movement arc*: By Lemma 7, $\overrightarrow{wv}$ is not a movement or special arc, and by the definition of movement arcs in Table 2, in the initial vertex-ordering, $v < w$. In Step 4 of the Diagonal Layout & Movement algorithm, v will move past w in the I-ordering. For any movement arc $\overrightarrow{wx}$ there is a $*$-edge $\{\overrightarrow{vw}, \overrightarrow{wx}\}$ in H, which ensures that w does not also move in the I-ordering. Hence in the final I-ordering, $w <_I v$. Since $\overrightarrow{vw}$ is a movement arc, $w = v^A$ or $w = v^B$ (refer to Tables 2 and 3). Hence, Step 2 of the Port Assignment algorithm defines $\theta(\overrightarrow{vw}) = -$. Therefore $\text{port}(\overrightarrow{vw}) = I_v^-$, which points towards w. Since both $\overrightarrow{vw}$ and $\overrightarrow{wv}$ are not special arcs, $\{\overrightarrow{vw}, \overrightarrow{wv}\}$ is an edge in H, and hence $I \neq J$. Thus $\text{port}(\overrightarrow{vw})$ and $\text{port}(\overrightarrow{wv})$ are perpendicular, and therefore $\overrightarrow{vw}$ is not anchored.

Case 2. *$\overrightarrow{vw}$ is a successor arc which is not a movement arc*: Then $w \in \{v^D, v^E, v^F\}$, and in the initial vertex-ordering, $v < w$. Step 2 of the Port Assignment algorithm defines $\theta(\overrightarrow{vw}) = +$. Thus $\text{port}(\overrightarrow{vw}) = I_v^+$.

Case 2(*a*). *w is non-negative*: In Step 4 of the Diagonal Layout & Movement algorithm, if v moves then it will not move past w, and if w moves then it will move away from v. Hence in the final I-ordering, $v <_I w$.

Case 2(*b*). *$\overrightarrow{wv}$ is a movement arc, and* (i) $v = w^1$ *and* (w, w^2) *is not a movement arc, or* (ii) $v = w^2$: If w moves in the I-ordering, then it moves toward v, however, by Lemma 8, any movement arcs at v do not 'cross over' or have the same destinations as a movement arc at w (ignoring $\overrightarrow{wv}$ which is not coloured I). Hence, in the final I-ordering, $v <_I w$.

Case 2(*c*). *$\overrightarrow{wv}$ is a movement arc, $v=w^1$ and w^2 is a movement arc*: Then w is a negative $(0,5)$- or $(0,6)$-vertex. If the arc (w,w^2) is coloured K, then the $**$-edge $\{(w,w^2),\overrightarrow{vw}\}$ in H ensures that $K\neq I$. The unlabelled edge $\{(w,w^1),(w,w^2)\}$ ensures that $K\neq J$. Hence I, J and K are pairwise distinct. Thus, w will not move in the I-ordering, and if v moves in the I-ordering then it will not move past w (otherwise $\overrightarrow{vw}$ would also be a movement arc). Thus in the final I-ordering, $v<_I w$.

Case 2(*d*). *$\overrightarrow{wv}$ is not a movement arc*: By Lemma 8, in the initial vertex-ordering the movement arcs of v do not 'cross over' or have the same destination vertex as the movement arcs of w. Thus, in the final I-ordering, $v<_I w$.

We have shown that in each of the sub-cases of Case 2 that in the final I-ordering, $v<_I w$. Thus port$(\overrightarrow{vw})$ points toward w. Unless $\overrightarrow{wv}$ is special, the r-edge $\{\overrightarrow{vw},\overrightarrow{wv}\}$ in H ensures that $I\neq J$. Thus port$(\overrightarrow{vw})$ and port$(\overrightarrow{wv})$ are perpendicular. If $\overrightarrow{wv}$ is special then, as shown above, port$(\overrightarrow{wv})$ points away from v. Thus, the Edge Construction algorithm will not anchor $\overrightarrow{vw}$.

Case 3. *$\overrightarrow{vw}$ is a predecessor arc*: Then $w\in\{v^A,v^B,v^C\}$, $\overrightarrow{vw}$ is not a movement arc nor a special arc, and in the initial vertex-ordering, $w<v$. Step 2 of the Port Assignment algorithm defines $\theta(\overrightarrow{vw})=-$. Thus port$(\overrightarrow{vw})=I_v^-$.

Case 3(*a*). *$\overrightarrow{wv}$ is not a movement arc and is not a special arc*: If w moves it does not move past v. If v moves then it moves away from w. Thus in the final I-ordering, $w<_I v$.

Case 3(*b*). *$\overrightarrow{wv}$ is a movement arc, and* (i) *$v=w^1$ and w^2 is not a movement arc, or* (ii) *$v=w^2$*: w will only move in the J-ordering. If v moves then it moves away from w. Thus in the final I-ordering, $w<_I v$.

Case 3(*c*). *$\overrightarrow{wv}$ is a movement arc, $v=w^1$ and (w,w^2) is a movement arc*: (that is, w is a $(0,5)$- or $(0,6)$-vertex) If the arc (w,w^2) is coloured $K\in\{X,Y,Z\}$, then the $**$-edge $\{(w,w^2),\overrightarrow{vw}\}$ ensures that $K\neq I$. The unlabelled edge $\{(w,w^1),(w,w^2)\}$ ensures that $K\neq J$. Hence I, J and K are pairwise distinct. Thus, w will not move in the I-ordering, and if v moves in the I-ordering then it will move away from w. Thus in the final I-ordering, $w<_I v$.

Case 3(*d*). *$\overrightarrow{wv}$ is a special arc*: In this case it is possible for $I=J$. If w is a $(1,5)$- or $(0,6)$-vertex then w may move in the I-ordering, but it will not move past v. If v moves in the I-ordering then it will move away from w. Thus in the final I-ordering, $w<_I v$.

We have shown that in each of the sub-cases of Case 3, in the final I-ordering $w<_I v$. Thus port$(\overrightarrow{vw})$ points toward w. Unless $\overrightarrow{wv}$ is special, the r-edge $\{\overrightarrow{vw},\overrightarrow{wv}\}$ in H ensures that $I\neq J$. Thus, port$(\overrightarrow{vw})$ and port$(\overrightarrow{wv})$ are perpendicular. If $\overrightarrow{wv}$ is special then, as shown above, port$(\overrightarrow{wv})$ points away from v. Thus, the Edge Construction algorithm will not anchor $\overrightarrow{vw}$.

Hence an arc which is not special does not become anchored (before edge-crossings are removed), and thus the number k of anchored arcs is the number of special arcs. Theorem 2 asserts that the Diagonal Layout & Movement algorithm computes a 4-bend drawing of G with volume $(n+\frac{1}{3}k)^3$ and $2m+k$ bends. There is precisely one special arc for each vertex with degree six which is unbalanced in the initial vertex-ordering. Therefore by Lemma 5, $k\leqslant\frac{6}{7}n$. Hence the volume is at most $(\frac{9}{7}n)^3 \leqslant 2.13n^3$.

Let δ be the average degree of those vertices with no outgoing special arcs; that is, vertices which are balanced or with degree at most five. Then $\delta \leqslant 6$ and by Lemma 5 for 6-graphs, $k \leqslant \delta/(\delta+1)n$, which implies $\frac{1}{7}k(\delta+1) \leqslant \frac{1}{7}\delta n$ and $k - \frac{6}{7}k \leqslant \frac{1}{7}\delta(n-k)$. Hence $k \leqslant \frac{1}{7}(\delta(n-k)+6k)$, and since all vertices not contributing to δ have degree six, $k \leqslant 2m/7$. Therefore, the total number of bends $2m+k \leqslant \frac{16}{7}m$; that is, the average number of bends is at most $2\frac{2}{7}$.

For maximum degree five graphs, no special arcs are introduced by the Diagonal Layout & Movement algorithm, and hence the Edge Construction algorithm produces no anchored edge routes. With no anchored edge routes, no new anchors can be introduced by the Crossing Removal algorithm. Thus, the crossing-free drawing has two bends per edge route and volume n^3.

By Lemma 6, determining the initial vertex-ordering takes $O(\Delta^4 m)$ time, which is $O(n)$ for 6-graphs. By Lemma 9, the 3-colouring of H takes $O(n)$ time. By Theorem 2, algorithm General Position Drawing takes $O(n^2)$ time, and thus the Diagonal Layout & Movement algorithm takes $O(n^2)$ time. The drawings produced are orientation-independent since throughout the algorithm each dimension is 'equivalent'; in particular, the box surrounding the vertices asymptotically is a cube, as is the bounding box. □

Note that special arcs are precisely those arcs which become anchored, and if $\overrightarrow{vw}$ is a special arc then $\overrightarrow{wv}$ is not a special arc (Lemma 7). Thus, no 4-bend edge routes are initially constructed—it is only by removing edge-crossings that a 4-bend edge route is introduced.

4. Conclusion

This paper describes an algorithm which determines three-dimensional orthogonal drawing of a 6-graph with at most an average of $2\frac{2}{7}$ bends per edge. This is the best known upper bound on the number of bends in three-dimensional orthogonal graph drawings. The drawings produced are in the general position model. For general position drawings the above bound is tight for K_7. Since every edge in a general position drawing has at least two bends, the Diagonal Layout & Movement algorithm is an $\frac{8}{7}$-approximation for the problem of minimising the number of bends in a general position orthogonal drawing of a given 6-graph. For drawings not necessarily in general position, however, there is a substantial difference between our upper bound and the best lower bounds. For example, the best known lower bound is $\frac{20}{21}$ average bends per edge [39]. Closing this gap is an interesting open problem.

Acknowledgements

The advice, encouragement and help with proofs provided by the author's Ph.D. supervisor Dr Graham Farr is gratefully acknowledged.

References

[1] A. Aggarwal, M. Klawe, P. Shor, Multilayer grid embeddings for VLSI, Algorithmica 6 (1) (1991) 129–151.
[2] H. Alt, M. Godau, S. Whitesides, Universal 3-dimensional visibility representations for graphs, Comput. Geom. 9 (1998) 111–125.
[3] B. Baetz, D.R. Wood, Brooks' vertex-colouring theorem in linear time, See Technical Report CS-AAG-2001-05, Basser Department of Computer Science, The University of Sydney, 2001, submitted for publication.
[4] T.C. Biedl, Heuristics for 3D-orthogonal graph drawings, Proc. 4th Twente Workshop on Graphs and Combinatorial Optimization, Universiteit Twente, Netherlands, 1995, pp. 41–44.
[5] T.C. Biedl, Three approaches to 3D-orthogonal box-drawings, in: S. Whitesides (Ed.), Proc. 6th Internat. Symp. on Graph Drawing (GD'98), Lecture Notes in Computer Science, Vol. 1547, Springer, Berlin, 1998, pp. 30–43.
[6] T. Biedl, T. Chan, Cross-coloring: improving the technique by Kolmogorov and Barzdin, Technical Report CS-2000-13, Department of Computer Science, University of Waterloo, Canada, 2000.
[7] T. Biedl, T. Chan, Y. Ganjali, M. Hajiaghayi, D.R. Wood, Balanced vertex-orderings of graphs, Technical Report CS-AAG-2001-01, Basser Department of Computer Science, The University of Sydney, Australia, 2001.
[8] T. Biedl, J.R. Johansen, T. Shermer, D.R. Wood, Orthogonal drawings with few layers, in: P. Mutzel, M. Jünger, S. Leipert (Eds.), Proc. 9th Internat. Symp. on Graph Drawing (GD'01), Lecture Notes in Computer Sci., Vol. 2265, Springer, Berlin, 2002, pp. 297–311.
[9] T. Biedl, T. Thiele, D.R. Wood, Three-dimensional orthogonal graph drawing with optimal volume, in: J. Marks (Ed.), Proc. 8th Internat. Symp. on Graph Drawing (GD'00), Lecture Notes in Computer Science, Vol. 1984, Springer, Berlin, 2001, pp. 284–295.
[10] P. Bose, H. Everett, S. Fekete, M. Houle, A. Lubiw, H. Meijer, K. Romanik, G. Rote, T. Shermer, S. Whitesides, C. Zelle, A visibility representation for graphs in three dimensions, J. Graph Algorithms Appl. 2 (3) (1998) 1–16.
[11] R.L. Brooks, On colouring the nodes of a network, Proc. Cambridge Philos. Soc. 37 (1941) 194–197.
[12] M. Chrobak, M. Goodrich, R. Tamassia, Convex drawings of graphs in two and three dimensions, Proc. 12th Annual ACM Symp. on Computational Geometry, ACM, New York, 1996, pp. 319–328.
[13] M. Closson, S. Gartshore, J. Johansen, S.K. Wismath, Fully dynamic three-dimensional orthogonal graph drawing, J. Graph Algorithms Appl. 5 (2) (2001) 1–34.
[14] R.F. Cohen, P. Eades, T. Lin, F. Ruskey, Three-dimensional graph drawing, Algorithmica 17 (2) (1996) 199–208.
[15] G. Di Battista, P. Eades, R. Tamassia, I.G. Tollis, Graph Drawing: Algorithms for the Visualization of Graphs, Prentice-Hall, Englewood Cliffs, NJ, 1999.
[16] G. Di Battista, M. Patrignani, F. Vargiu, A split & push approach to 3D orthogonal drawing, J. Graph Algorithms Appl. 4 (3) (2000) 105–133.
[17] P.F. Dietz, D.D. Sleator, Two algorithms for maintaining order in a list, Proc. 19th Annual ACM Symp. on Theory of Computing (STOC'87), ACM, New York, 1987, pp. 365–372.
[18] P. Eades, Q.-W. Feng, Multilevel visualization of clustered graphs, in: S. North (Ed.), Proc. 4th Internat. Symp. on Graph Drawing (GD'96), Lecture Notes in Computer Science, Vol. 1190, Springer, Berlin, 1997, pp. 101–112.
[19] P. Eades, P. Garvan, Drawing stressed planar graphs in three dimensions, in: F.J. Brandenburg (Ed.), Proc. Internat. Symp. on Graph Drawing (GD'95), Lecture Notes in Computer Science, Vol. 1027, Springer, Berlin, 1996, pp. 212–223.
[20] P. Eades, C. Stirk, S. Whitesides, The techniques of Kolmogorov and Barzdin for three dimensional orthogonal graph drawings, Inform. Proc. Lett. 60 (2) (1996) 97–103.
[21] P. Eades, A. Symvonis, S. Whitesides, Two algorithms for three dimensional orthogonal graph drawing, in: S. North (Ed.), Proc. 4th Internat. Symp. on Graph Drawing (GD'96), Lecture Notes in Computer Science, Vol. 1190, Springer, Berlin, 1997, pp. 139–154.
[22] P. Eades, A. Symvonis, S. Whitesides, Three dimensional orthogonal graph drawing algorithms, Discrete Applied Math. 103 (2000) 55–87.

[23] A. Garg, R. Tamassia, P. Vocca, Drawing with colors, in: J. Diaz, M. Serna (Eds.), Proc. 4th Annual European Symp. on Algorithms (ESA'96), Lecture Notes in Computer Science, Vol. 1136, Springer, Berlin, 1996, pp. 12–26.
[24] P.L. Garvan, Drawing and labelling graphs in three-dimensions, in: M. Patel (Ed.), Proc. 20th Australasian Comput. Sci. Conf. (ACSC'97), Australian Computer Science Commissions, Vol. 19 (1), Macquarie University, 1997, pp. 83–91.
[25] N.D. Kazarinoff, Analytic Inequalities, Holt, Rinehart and Winston, New York, 1961.
[26] A.N. Kolmogorov, Y.M. Barzdin, On the realization of nets in 3-dimensional space, Problems Cybernetics 8 (1967) 261–268.
[27] L. Lovász, Three short proofs in graph theory, J. Combin. Theory Ser. B 19 (1975) 269–271.
[28] B.Y.S. Lynn, A. Symvonis, D.R. Wood, Refinement of three-dimensional orthogonal graph drawings, in: S. North (Ed.), Proc. 4th Internat. Symp. on Graph Drawing (GD'96), Lecture Notes in Computer Science, Vol. 1190, Springer, Berlin, 1997, pp. 308–320.
[29] J. Pach, T. Thiele, G. Toth, Three-dimensional grid drawings of graphs, in: G. Di Battista (Ed.), Proc. 5th Internat. Symp. on Graph Drawing (GD'97), Lecture Notes in Computer Science, Vol. 1353, Springer, Berlin, 1998, pp. 47–51.
[30] A. Papakostas, I.G. Tollis, Improved algorithms and bounds for orthogonal drawings, in: R. Tamassia, I.G. Tollis (Eds.), Proc. DIMACS Internat. Workshop on Graph Drawing (GD'94), Lecture Notes in Computer Science, Vol. 894, Springer, Berlin, 1995, pp. 40–51.
[31] A. Papakostas, I.G. Tollis, Algorithms for incremental orthogonal graph drawing in three dimensions, J. Graph Algorithms Appl. 3 (4) (1999) 81–115.
[32] M. Patrignani, On the complexity of orthogonal compaction, Comput. Geom. 19 (1) (2001) 47–67.
[33] M. Patrignani, F. Vargiu, 3DCube: a tool for three dimensional graph drawing, in: G. Di Battista (Ed.), Proc. 5th Internat. Symp. on Graph Drawing (GD'97), Lecture Notes in Computer Science, Vol. 1353, Springer, Berlin, 1998, pp. 284–290.
[34] A.L. Rosenberg, Three-dimensional VLSI: a case study, J. ACM 30 (2) (1983) 397–416.
[35] C. Ware, G. Franck, Viewing a graph in a virtual reality display is three times as good as a 2D diagram, in: A.L. Ambler, T.D. Kimura (Eds.), Proc. IEEE Symp. Visual Languages (VL'94), IEEE, Los Alamitos, CA, 1994, pp. 182–183.
[36] C. Ware, G. Franck, Evaluating stereo and motion cues for visualizing information nets in three dimensions, ACM Trans. Graphics 15 (2) (1996) 121–140.
[37] D.R. Wood, On higher-dimensional orthogonal graph drawing, in: J. Harland (Ed.), Proc. Computing: the Australasian Theory Symposium (CATS'97), Vol. 19(2), Australian Computer Science Commission, 1997, pp. 3–8.
[38] D.R. Wood, Three-dimensional orthogonal graph drawing, School of Computer Science and Software Engineering, Ph.D. Thesis, Monash University, Melbourne, Australia, 2000.
[39] D.R. Wood, Lower bounds for the number of bends in three-dimensional orthogonal graph drawings, in: J. Marks (Ed.), Proc. 8th Internat. Symp. on Graph Drawing (GD'00), Lecture Notes in Computer Science, Vol. 1984, Springer, Berlin, 2001, pp. 259–271.
[40] D.R. Wood, Minimising the number of bends and volume in three-dimensional orthogonal graph drawings with a diagonal vertex layout, See Technical Report CS-AAG-2001-03, Basser Department of Computer Science, The University of Sydney, 2001, submitted for publication.