Three-Dimensional Orthogonal Graph Drawing

by

David R. Wood

B.Sc. (Hons), Dip.Ed. *W.Aust.*

A thesis submitted to the

School of Computer Science and Software Engineering

Monash University

for the degree of

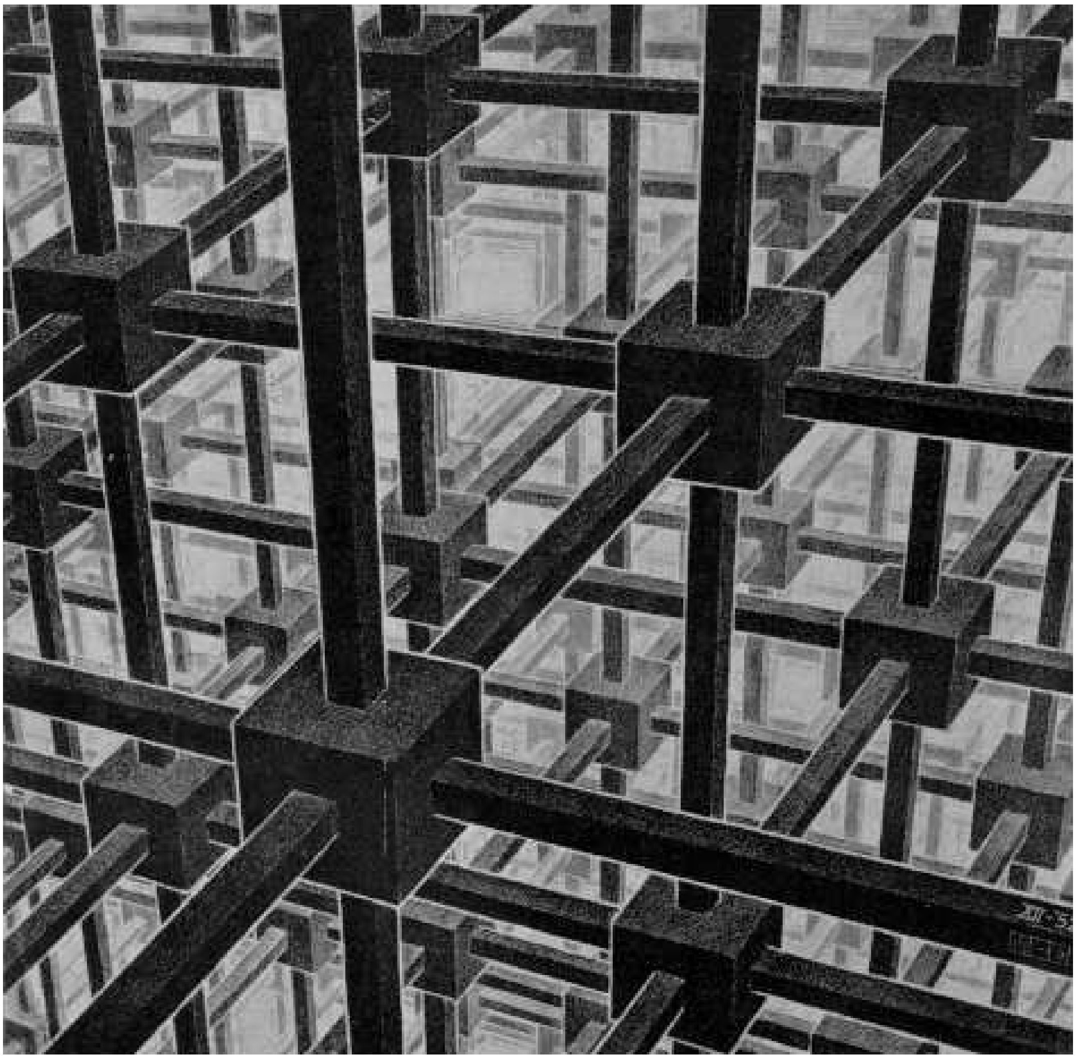
Doctor of Philosophy

August, 2000

"Design, which by another name is called drawing ...
is the fount and body of painting and sculpture and architecture
... and the root of all sciences."

Michelangelo, 1548[†].

[†]Holroyd [118, p.275]



M.C. Escher
Cubic Space Division, 1952
Lithograph

For Nanny
and the Kelly-Wood clan.

Contents

Abstract

The visualisation of relational information has many applications in diverse domains such as software engineering and cartography. Relational information is typically modelled by an abstract graph, where vertices are entities and edges represent relationships between entities. The aim of graph drawing is to automatically produce drawings of graphs which clearly reflect the inherent relational information.

Numerous graph drawing styles have been proposed in the literature. Orthogonal graph drawings have been widely studied due to their appropriateness in a variety of visualisation applications and in the design of VLSI circuitry. Most of the research conducted in graph drawing, including orthogonal drawings, has dealt with drawings in the plane. With the widespread availability of graphics workstations and the development of software systems for three-dimensional graphics, there has been recent interest in the design and analysis of algorithms for three-dimensional graph drawing.

This thesis is primarily concerned with problems related to the automatic generation of three-dimensional orthogonal graph drawings. Our methods also have application to two-dimensional orthogonal graph drawing and generalise to higher dimensional space.

In particular, we develop a number of models for three-dimensional orthogonal graph drawing, and within each model, algorithms are presented which explore trade-offs between the established aesthetic criteria. The main achievements include (1) an algorithm for producing three-dimensional orthogonal box-drawings with optimal volume for regular graphs, (2) an algorithm for producing degree-restricted three-dimensional orthogonal cube-drawings with optimal volume, (3) an algorithm which establishes the best known upper bound for the total number of bends in three-dimensional orthogonal point-drawings, and (4) an algorithm which establishes the best known upper bound for the volume of 3-D orthogonal point-drawings with three bends per edge route.

As a by-product of this investigation, we develop methods for a number of combinatorial problems of independent interest, including the balanced vertex ordering problem, equitable edge-colouring of multigraphs, and the maximum clique problem.

Declaration

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university or other institution.

To the best of my knowledge, this thesis contains no material previously published or written by another person, except where due reference is made in the text.

Where the material in this thesis is based upon joint research, the collaborating authors are disclosed.

————————————————

David R. Wood

August, 2000

Acknowledgements

I would like to express deep appreciation of my supervisor, Dr Graham Farr, firstly for accepting me as a PhD student and for allowing me the freedom to choose a research topic. Throughout my candidature his friendship, enthusiasm and confidence in my abilities, not to mention help with many proofs, have been a great motivation towards the completion of this thesis.

I am enormously grateful to Dr Antonios Symvonis for allowing me to finish this thesis while working as a research associate in the Basser Department of Computer Science at The University of Sydney.

To my collaborators Dr. Therese Biedl and Dr. Torsten Thiele, thank you for two stimulating days of research we shared together. Special thanks to Therese for thoroughly reading this thesis and providing numerous helpful suggestions.

Thanks to Prof. Peter Eades of The University of Newcastle for posing a fun-looking puzzle which sparked my interest in orthogonal graph drawing. Thanks also to Prof. Cheryl Praeger of The University of Western Australia and Dr. Robyn Wilson. Their input into my academic career is greatly appreciated.

I am grateful to the School of Computer Science and Software Engineering at Monash University for the provision of a scholarship and generous funding for conference travel. To my office mates at Monash University, Grace Rumantir, Julian Neil, Nathalie Jitnah and Linda McIver, thank you for providing a great atmosphere to work in and for relief from the rigours of postgraduate study. Life at Monash would not have been the same without the Meat and all of the ballhandlers.

Thanks to Cordon Art B.V., Tom Sawyer Software, Dynamic Diagrams and the NCSA for kindly allowing the reproduction of copyright material in this thesis.

My deepest gratitude to my parents and family for teaching me to think for myself, and always encouraging me in and appreciating whatever I chose to do.

Finally, I would like to thank Mandy. You're wonderful.

List of Figures

List of Tables

List of Algorithms

Part I

Orthogonal Graph Drawing

Chapter 1

Introduction

In this chapter we provide a broad overview of graph drawing applications and conventions, surveying the theoretical background to the development of algorithms for drawing graphs. This provides the setting and motivation for the results presented in the remainder of the thesis.

1.1 Graph Drawing

Graph drawing is concerned with the automatic generation of geometric representations of relational information, often for visualisation purposes. The typical data structure for modelling relational information is a graph whose vertices represent entities and whose edges correspond to relationships between entities. Most applications of graph drawing call for two-dimensional drawings, although with the widespread availability of graphics workstations, there has been considerable recent interest in three-dimensional graph drawing. As can be seen in the three-dimensional representation of network traffic in Figure 1.1, drawing graphs in three dimensions allows for more flexible drawings than if we restrict the drawing to the plane.

Software engineering has provided considerable motivation for the development of graph drawing algorithms. The method for laying out data-flow diagrams due to Knuth [128] was one of the first graph drawing algorithms for visualisation purposes. More recently, methods for drawing in three-dimensional space have been developed for visualising object-oriented class structures by Robertson *et al.* [180], Koike [131], Ware

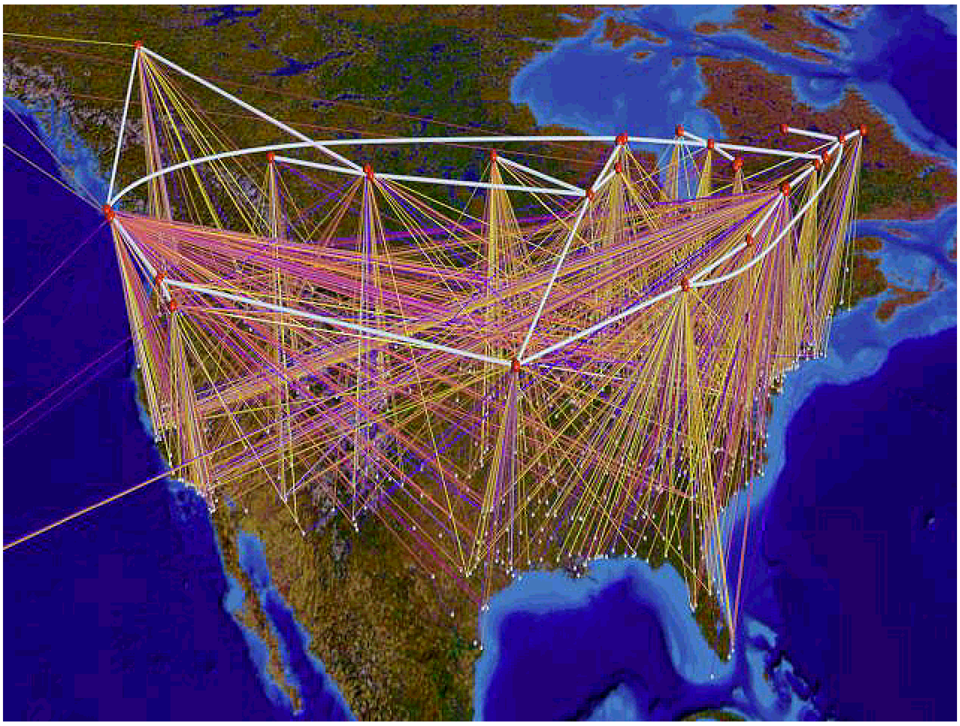


Figure 1.1: A 3-D drawing representing NSFNET traffic, courtesy of the NCSA. (http://www.ncsa.uiuc.edu)

et al. [214] and Reiss [179]. Batini *et al.* [15] present an algorithm for the display of entity-relationship diagrams in database systems. Munzner and Burchard [158] have explored the use of graph drawing techniques for visualising the world wide web in three dimensions, In Figure 1.2 we present a three-dimensional representation of the organisation of an internet site.

An important area for the application of graph drawing techniques is the automatic layout of VLSI circuit schematics. In two dimensions such algorithms have been developed by Quinn Jr. and Breuer [177], Leiserson [141], Bhatt and Leighton [22] and Schlag *et al.* [191] (see also Lengauer [143]). Three-dimensional VLSI layouts have been investigated by Preparata [173], Rosenberg [185, 186], Leighton and Rosenberg [140] and Aboelaze and Wah [1]. Three-dimensional field-programmable gate arrays (FPGAs) have been designed by Veretennicoff *et al.* [210], and in the Rothko project at Northeastern University, Leeser *et al.* [138, 139] and Meleis *et al.* [153] construct three-dimensional FPGAs with interconnections between layers of active devices.

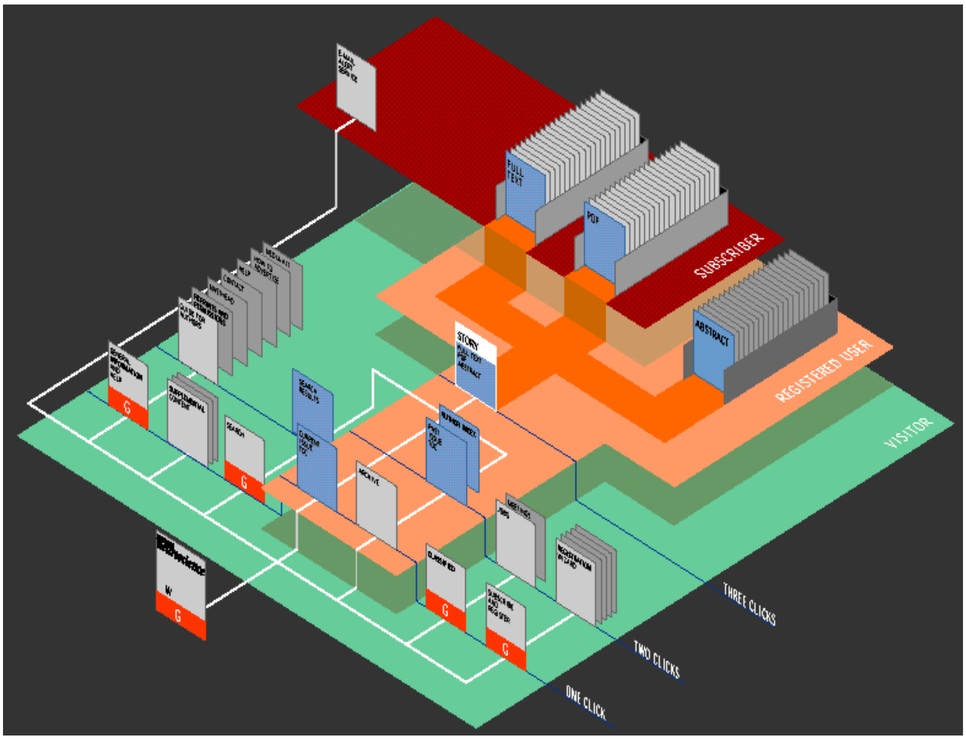


Figure 1.2: A 3-D drawing representing the organisation of part of the web site for the journal Nature Neuroscience, courtesy of Dynamic Diagrams (`http://www.dynamicdiagrams.com`).

Other scientific applications for graph drawing include biology (evolutionary trees), chemistry (molecular drawings), architecture (floor plan maps) and cartography (map schematics). The drawing of graphs which arise in mathematics, such as commutativity diagrams, is an often overlooked application domain for graph drawing.

1.2 Algorithmic Graph Theory

Algorithms for drawing graphs are typically based on some graph-theoretic decomposition or insight into the structure of the graph. We now survey the development of algorithmic graph theory, highlighting the algorithmic approaches employed in this thesis.

For many years in the shadow of topology, abstract graph theory is now a well-developed theory with important connections to number theory, logic, algebra, knot theory and probability (see Beineke and Wilson [18]). Recent deep structural results, most notably the *minor theorem* of Robertson and Seymour [182] (see Diestel [76] for a comprehensive overview), have placed graph theory at the forefront of combinatorics. Furthermore, graph theory is now providing new insights into topology including the simple graph-theoretic proof due to Thomassen [207] of the notoriously difficult Jordon-Schönflies Curve Theorem. Recent highlights in topological graph theory include a new proof of the four-colour theorem by Robertson *et al.* [181], and the discovery of forbidden minor characterisations of graphs admitting certain topological embeddings, as discussed below.

Graph theory is often used to model real world algorithmic problems, such as scheduling and transportation. Furthermore many important issues in computational complexity theory are illustrated with graph-theoretic problems. For example, three of the six basic NP-complete problems in Garey and Johnson [105] deal with graphs. The theory of computational complexity dates from the study of the fundamental capabilities and limitations of computation by logicians such as Gödel, Church and Turing. Our understanding of computational complexity made great advances with the development of the theory of NP-completeness (see Garey and Johnson [105]) in the 1970s. The explosion of interest in the theory of algorithms in the past three decades has motivated much research in the field of graph theory. The growth of graph drawing as a discipline of Computer Science is a natural byproduct of this development.

As we shall see many graph drawing problems are NP-complete. Exact solutions to NP-complete problems, using integer programming formulations or branch and bound techniques, have exponential time complexity. An example of this approach is given in Appendix C, where we provide a branch and bound algorithm for the maximum clique problem, which combined with efficient heuristics to provide lower and upper bounds, solves relatively small instance of the maximum clique problem in a realistic amount of time.

Unless P=NP, exact polynomial time algorithms cannot be obtained for NP-complete problems. Much recent research has focused on classifying the approximability of problems, and the development of approximation algorithms which guarantee near-optimal solutions or at least have tight worst case performance bounds. For many of the graph drawing problems investigated in this thesis, we present approximation algorithms and heuristics with tight worst case bounds. Graph algorithms, such as topological ordering, matching and vertex- and edge-colouring form the basis of the many of the methods presented in this thesis.

1.3 Graph Embeddings and Representations

Many approaches to graph drawing, for example the topology-shape-metrics approach discussed in Section 3.2.2, and the algorithms presented in Sections 9.1 and 5.5, are based on graph embeddings. A graph embedding describes the essential topological features of a graph drawing. We now provide a review of the principal results from the theory of graph embeddings, concentrating on three-dimensional graph embeddings.

Planar Embeddings

One of the most famous result in graph theory is Kuratowski's characterisation of planar graphs. Kuratowski [137] showed that a graph is is planar if and only if it contains neither K_5 nor $K_{3,3}$ as a topological minor. The result was extended to general minors by Wagner [212]. Since these early results, the theory of planar graphs has been widely studied. Notable are the linear time algorithms for recognising planar graphs, for example that of Hopcroft and Tarjan [119].

Recently, relationships between graph embeddings and an algebraic graph invariant μ introduced by Colin de Verdière [61, 62] have been discovered. Colin de Verdière shows that $\mu(K_n) = n - 1$ and characterises those graphs G with $\mu(G) \leq k$ for each $k \leq 3$. In particular, $\mu(G) \leq 1$ if and only if G is a disjoint union of paths; $\mu(G) \leq 2$ if and only if G is outerplanar; and $\mu(G) \leq 3$ if and only if G is a planar. For each fixed k, the class of graphs with $\mu \leq k$ is closed under taking minors, so by the minor theorem there is a finite forbidden minor characterisation of such graphs. Note that Colin de Verdière conjectures that $\mu(G) \geq \chi(G) - 1$, a result which implies the 4-colour theorem.

Surface Embeddings

Embeddings of graphs in surfaces provide a natural generalisation of plane graphs. Informally, the *genus* of a graph G is the minimum k such that there is a embedding of G in the surface constructed from the sphere with k 'handles'. The sphere with one handle, called the *torus*, can be thought of as a rectangle whose sides have been identified. The drawing in Figure 1.3 of K_7 embedded in the torus is an elegant example of a surface embedding.

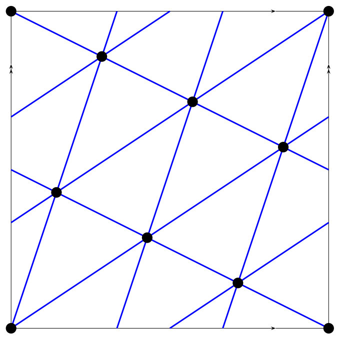


Figure 1.3: A straight-line drawing of K_7 on the 'square' torus.

A significant corollary of the minor theorem is that for every surface S there is a finite forbidden minor characterisation of those graphs embeddable in S [183]. Apart from the plane, the only surface where the complete list of forbidden minors is known is the projective plane, where the 35 minor-minimal graphs were discovered by Archdeacon [6]. Mohar [155] presents a linear time algorithm, which for a fixed surface S, finds an embedding of a given graph in S or identifies a subgraph homeomorphic to a forbidden minor for S.

Linkless Embeddings

A *spatial embedding* of a graph is an embedding in $\mathbb{R}^3$. A spatial embedding is *linkless* if there is no pair of disjoint linked cycles. A graph with a linkless embedding is said to be *linkless*, otherwise it is *self-linked*. Conway and Gordon [63] and Sachs [188] showed that K_6 is self-linked (see Figure 1.4).

A ΔY*-exchange* in a graph replaces a triangle by a 3-star, while a $Y\Delta$*-exchange*

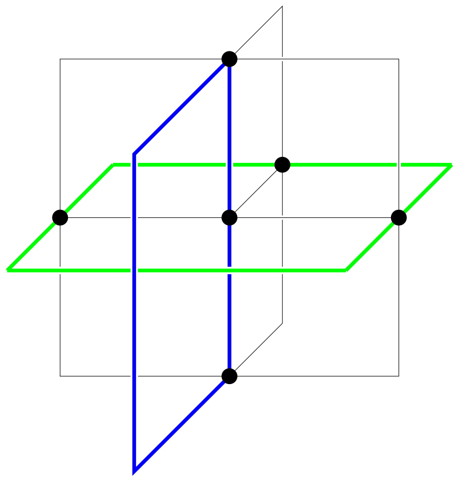


Figure 1.4: Linked spatial embedding of K_6.

replaces a 3-star by a triangle. Sachs [188] establishes that the six graphs obtained from K_6 by a sequence of ΔY-exchanges and $Y\Delta$-exchanges, called the *Petersen Family* (as the Petersen graph is a member), are also self-linked. Robertson *et al.* [184] show that these graphs comprise a forbidden minor characterisation of the class of linkless graphs[1]. Furthermore they show that a linkless graph has $\mu \leq 4$. Their conjecture that the converse is also true was established by Lovász and Schrijver [149].

Knotless Embeddings

A spatial embedding of a graph is said to *knotted* if there is a cycle which forms a non-trivial knot. We call a graph *knotless* if it has a spatial embedding which is not knotted, and *self-knotted* otherwise. Conway and Gordon [63] and Shimabara [196] respectively showed that K_7 and $K_{5,5}$ are self-knotted.

Up until the proof of the minor theorem it was unknown if there is an algorithm for deciding the knotlessness of a given graph. The class of knotless graphs is closed under taking minors, so by the minor theorem, remarkably there is an $O(n^3)$ algorithm to decide if a given graph is knotless, although no one knows what the algorithm is. It is a tantalising open problem to determine whether the knotless graphs are precisely those graphs with $\mu \leq 5$.

[1]The proof of this result announced by Motwani *et al.* [157] was refuted by Kohara and Suzuki [130].

Book Embeddings

A *book* consists of a line in 3-space, called the *spine*, and some number of *pages* (each a half-plane with the spine as boundary). A *book embedding* of a graph is a spatial embedding consisting of an ordering of the vertices, called the *spine ordering*, along the spine of a book and an assignment of edges to pages so that edges assigned to the same page can be drawn on that page without crossings; i.e., for any two edges vw and xy, if $v < x < w < y$ in the spine ordering then vw and xy are assigned different pages. The minimum number of pages in which a graph can be embedded is its *pagenumber*.

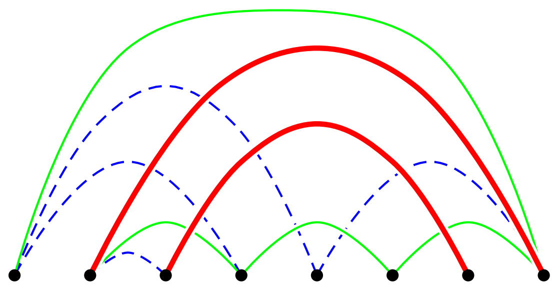


Figure 1.5: A 3-page book embedding of a graph

Yannakakis [226] showed that the maximum pagenumber of a planar graph is four. By the four-colour theorem [4, 5, 181], the maximum pagenumber and maximum chromatic number are equal for planar graphs. Similarly, Endo [88] showed that the pagenumber of a toroidal graph is at most seven. Since each toroidal graph is vertex 7-colourable [116], the maximum pagenumber is no more than the maximum chromatic number. It is a fascinating open problem (see [88]) to determine if the maximum pagenumber and maximum chromatic number are equal for all surfaces.

Heath and Istrail [115] proved that the pagenumber of a genus g graph is $O(g)$, and conjectured the correct bound is $O(\sqrt{g})$. This conjecture was confirmed by Malitz [150]. As a corollary of this result, and proved independently by Malitz [151], the pagenumber of a graph with m edges is $O(\sqrt{m})$. These results are non-deterministic in nature, and Las Vegas algorithms are presented to compute book embeddings with $O(\sqrt{g})$ pages. Book embeddings, and in particular these results of Malitz, form the basis of our algorithms presented in Sections 5.5 and 9.1.

Graph Representations

A *representation* of a graph, loosely speaking, describes the vertices by some set of geometric objects and the edges by some relationship between the objects. Examples include the visibility representations described in Section 3.2.1 and touching circle and sphere representations of graphs. Koebe [129] first proved that the vertices of a planar graph can be represented by non-overlapping circles in the plane, so that vertices are adjacent if and only if the corresponding circles are tangent. Kotlov *et al.* [134] have recently discovered relationships between the invariant μ and the touching sphere representations of graphs in $\mathbb{R}^3$.

1.4 Graph Drawing Conventions

We now describe the common conventions, or styles, of graph drawings for which algorithms have been developed. We concentrate on those conventions that have been used for three-dimensional graph drawing. For a complete summary see Di Battista, Eades, Tamassia, and Tollis [71]. While the criteria for deciding the quality of a given graph drawing is somewhat dependent on the application domain, for each graph drawing convention there is a commonly accepted set of aesthetic criteria by which the quality of a drawing is judged. For any graph and any style there is (typically) an infinite number of possible drawings. The goal of graph drawing algorithms is to produce drawings which satisfy the aesthetic criteria. More often than not we need to make a trade-off between the various aesthetic criteria. The study of trade-offs between various aesthetic criteria is at the heart of the study of graph drawing algorithms.

1.4.1 Grid Drawings

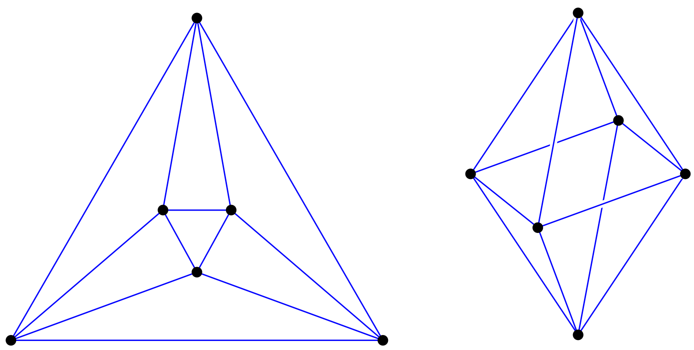
So that the area (or volume in three dimensions) of a graph drawing can be measured in a consistent fashion, we often require vertices to have integer coordinates. We say the vertices are placed at *grid-points* and such a drawing is called a *grid drawing.* The smallest rectangle (or box in three-dimensions) which surrounds a grid drawing is called the *bounding box.* The area (or volume) of the bounding box is perhaps the most commonly used quantity to measure the aesthetic quality of grid drawings. For example, drawings with small area can be drawn with greater resolution on a fixed-size page. In some three-dimensional applications, for example when visualising the drawing on a computer screen, it may be more important to minimise the 'depth' of the drawing. We therefore have the following possible aesthetic criteria for grid drawings.

- Minimise the bounding box volume.
- Minimise the minimum bounding box side length.
- Minimise the maximum bounding box side length.

An alternative to grid drawings is to stipulate that vertices are at least unit distance apart.

1.4.2 Straight Line Drawings

It is natural to draw each edge of a graph as a straight line between its end-vertices. So-called *straight-line* graph drawings are one of the earliest graph drawing conventions to be investigated. In Figure 1.6 we present examples of straight-line graph drawings.



(a) (b)

Figure 1.6: Straight-line drawings of the octahedron graph: (a) plane drawing, (b) 3-D drawing.

Aesthetic criteria for straight-line graph drawings include the following.

- Minimise edge crossings (in 2-D non-planar drawings).
- Maximise the *angular resolution*; i.e., the angle between edges incident at a common vertex.

- Minimise the *edge separation*; i.e., the distance between edges not incident to a common vertex.
- Minimise the total length of edge routes.
- Minimise the maximum length of an edge route.
- Preserve the symmetry of the graph.

Note that Purchase *et al.* [176] and Purchase [175] concluded from their experimental study of the human perception of 2-D graph drawings that minimising the number of edge crossings and minimising the number of bends were both significant aesthetic criteria for increasing the understandability of drawings of graphs.

That every planar graph has a straight-line plane drawing was proved independently by Wagner [211], Fáry [94] and Stein [198]. In a recent extension of this result, Brightwell and Scheinerman [45] show that a planar graph and its dual can be simultaneously represented in the plane with straight-line edge routes such that the edges of the graph cross the dual edges at right angles. These authors were only really interested in proving the existence of straight-line embeddings and not with producing algorithms for graph drawing. In particular, if we stipulate minimum unit distance between vertices then exponential area may be required by these methods. de Fraysseix *et al.* [66] and Schnyder [192] independently developed algorithms for planar straight-line grid drawing with $O(n^2)$ area.

Every simple graph has a straight-line 3-D grid drawing with no crossings, and for this reason we only consider crossing-free 3-D graph drawings. To construct such a drawing of a graph with vertex set $\{v_1, v_2, \ldots, v_n\}$, vertices are positioned along a *moment curve*; i.e., v_i is at $(i, i^2, i^3) \in \mathbb{Z}^3$. It is easily seen that no two straight lines between vertices can intersect. This drawing has $O(n^6)$ bounding box volume. Cohen *et al.* [60] showed that by placing vertex v_i at $(i \mod p, i^2 \mod p, i^3 \mod p) \in \mathbb{Z}^3$ for some prime p, $n < p < 2n$, no two edge routes cross and we obtain a grid drawing with $O(n^3)$ bounding box volume. This result has been strengthened by Pach *et al.* [161] who show that every k-colourable graph, for some fixed k, has a 3-D straight-line grid drawing with $O(n^2)$ volume. Instead of requiring vertices to be at grid-points, Garg *et al.* [108] stipulate that distinct vertices are at least unit distance apart in a

3-D straight-line graph drawing. Their algorithm establishes bounds on the bounding box volume, aspect ratio and edge separation of such drawings. Simulated annealing techniques for generating 3-D straight-line graph drawings have been developed by Cruz and Twarog [65] and Monien *et al.* [156].

One of the earliest graph drawing methods, namely the *barycentre method*, was developed by Tutte [208, 209]. Here a fixed set of vertices are placed on a strictly convex polygon, and the remaining vertices, said to be *free*, are repeatedly placed at the barycentre of their neighbours until the coordinates of the free vertices converges. If the input graph is triconnected and planar, then the drawing produced is planar and each face is a convex polygon. The barycentre method has been extended to produce 3-D straight-line graph drawings by Chilakamarri *et al.* [55].

The barycentre method is an example of the *force-directed* approach for graph drawing. Here the graph is viewed as a physical system with forces acting between the constituent bodies. For example, edges can be modelled as springs and vertices as charged particles which repel each other (see Di Battista *et al.* [71] for details and references). Force directed methods for producing 3-D graph drawings have been studied by Ostry [160] and Bruß and Frick [48]. As noted by Eades and Lin [83], an advantage of force directed algorithms is that symmetries of the graph are often preserved in the drawing.

A relationship between the force-directed approach to graph layout and graph connectivity was discovered by Linial *et al.* [144], later extended to the case of digraphs by Cheriyan and Reif [54]. They prove that a (di)graph G is k-connected ($k \geq 2$) if and only if for any $X \subseteq V(G)$ with $|X| = k$ there is a *convex-X embedding* of G; i.e., the vertices of G can be represented by points in general position in $\mathbb{R}^{k-1}$ (i.e., no k vertices are on a common hyperplane), so that each vertex, except for the k specified vertices in X, is in the convex hull of its (out)neighbours. This result generalises the notion of st-orderings (used extensively in graph drawing; see Sections 3.2.3 and 4.2) to arbitrary dimensions. The proof is based on a physical model where the edges are ideal springs and the vertices settle into equilibrium. Although the authors do not note this, for $k \geq 4$, edges drawn as straight lines cannot cross since the vertices are in general position.

An interesting graph invariant related to multi-dimensional straight-line graph drawing is that of the dimension of a graph. Erdős, Harary, and Tutte [89] define the *dimension* of a graph to be the minimum number of dimensions in which it can be embedded with each edge a unit length straight-line (possibly with crossings). They showed that the dimension of the complete graph K_n is $n-1$, and the dimension of the complete bipartite graph $K_{a,b}$ is four, among other results.

1.4.3 Orthogonal Drawings

In a *polyline* graph drawing each edge consists of a sequence of contiguous line segments. Di Battista *et al.* [71] describe algorithms for constructing planar polyline drawings. In a polyline grid drawing, the *bends* on edge routes as well as the vertices are required to be at grid points. If each segment of an edge in a polyline grid drawing is parallel to some axis then the drawing is called *orthogonal*. (Precise definitions are given in Chapter 2.) A feature of the orthogonal drawing style is its very good angular resolution. For this reason, it is commonly used for many applications including data-flow diagrams, and in VLSI circuit design where electrical wires must be axis-parallel. Examples of 'real-world' orthogonal graph drawings in two and three dimensions are shown in Figures 1.7 and 1.2, respectively.

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, the following properties are indicative of orientation-independent drawings.

- The bounding box is a cube.
- The box surrounding the vertices is a cube.
- It is equally likely that an edge incident with a particular vertex, is routed using any port on that vertex.

Whether or not orientation-dependence is a desirable quality in orthogonal drawings is often an application-specific question. We shall take the view that orientation-*in*dependent orthogonal drawings are more aesthetically pleasing than orientation-dependent orthogonal drawings. Orientation dependence is a particularly appropriate

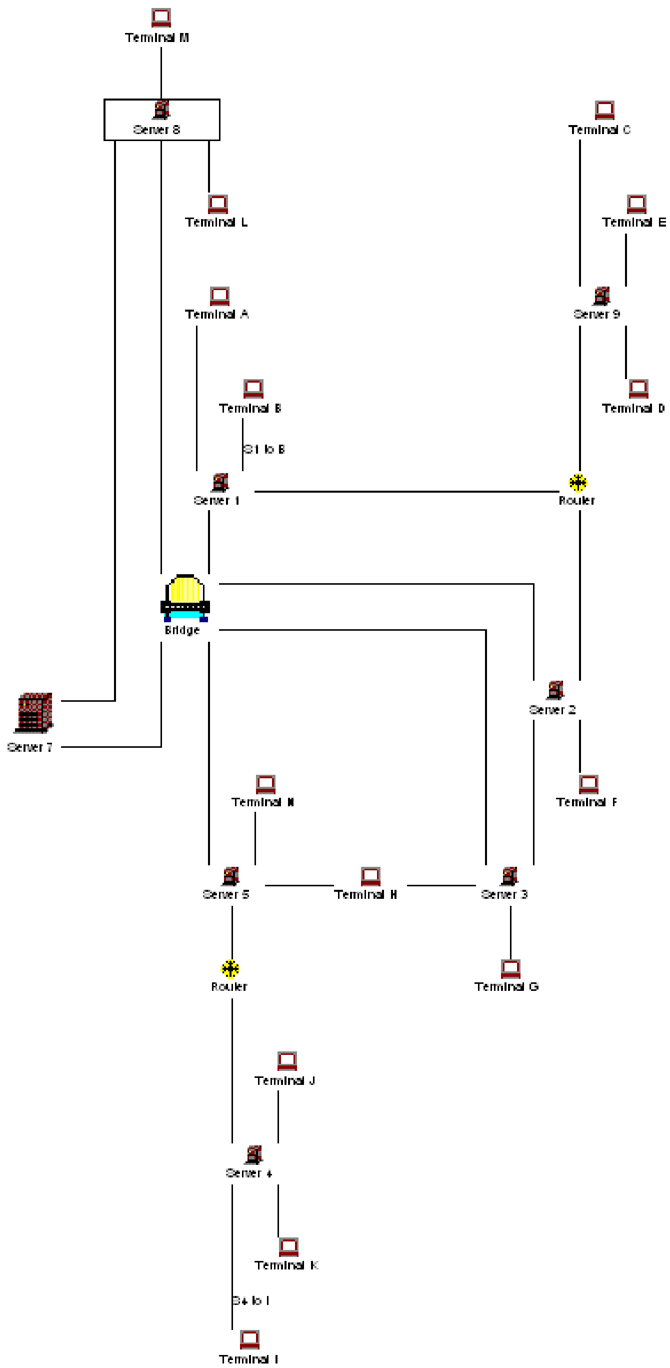


Figure 1.7: An orthogonal drawing of a computer network, courtesy of Tom Sawyer Software (`http://www.tomsawyer.com`)

consideration for 3-D orthogonal drawings. Biedl [27] describes orientation independent 3-D orthogonal drawings as being ‘truly three-dimensional’.

Orthogonal graph drawings with many bends appear cluttered and are difficult to visualise. Existing algorithms for two-dimensional orthogonal graph drawing have bounds on the maximum number of bends per edge route as well as the total number of bends. Up until now, algorithms for 3-D orthogonal graph drawing have concentrated only on the maximum number of bends per edge route. The algorithms for orthogonal graph drawing presented in Chapter 5 initiate the study of the total number of bends in 3-D orthogonal drawings. As well as the aesthetic criteria already discussed in the previous section, orthogonal graph drawings should exhibit the following properties.

- Minimise the maximum number of bends per edge route.
- Minimise the total number of bends.
- Drawings should be orientation-independent.

For orthogonal graph drawings a number of tradeoffs between aesthetic criteria, most notably between the maximum number of bends per edge route and the bounding box volume, have been observed in existing algorithms [87]. In this thesis we shall also observe a tradeoff between orientation-independence and bounding box volume, and between orientation-independence and the maximum number of bends per edge route. In Figure 1.8 we present orthogonal drawings of the octahedron which demonstrate some of the aesthetic criteria for such drawings.

If we represent each vertex by a point, as in the above examples, for a graph to admit a two-dimensional orthogonal drawing each vertex must have degree at most four. In three dimensions each vertex must have degree at most six. Overcoming this restriction has motivated the consideration of orthogonal box-drawing where vertices are represented by rectangles in two dimensions and by boxes in three dimensions. Box-drawings also have the advantage that a label can be attached to each vertex.

For orthogonal box-drawings the size and shape of the boxes representing the vertices is also considered an important measure of aesthetic quality. For the purposes of visualisation, the ideal shape for a box is a small cube, as this most closely resembles a point. How closely a vertex resembles a point can be measured by its *aspect ratio* which is defined to be the ratio of the length of the longest side to that of the shortest.

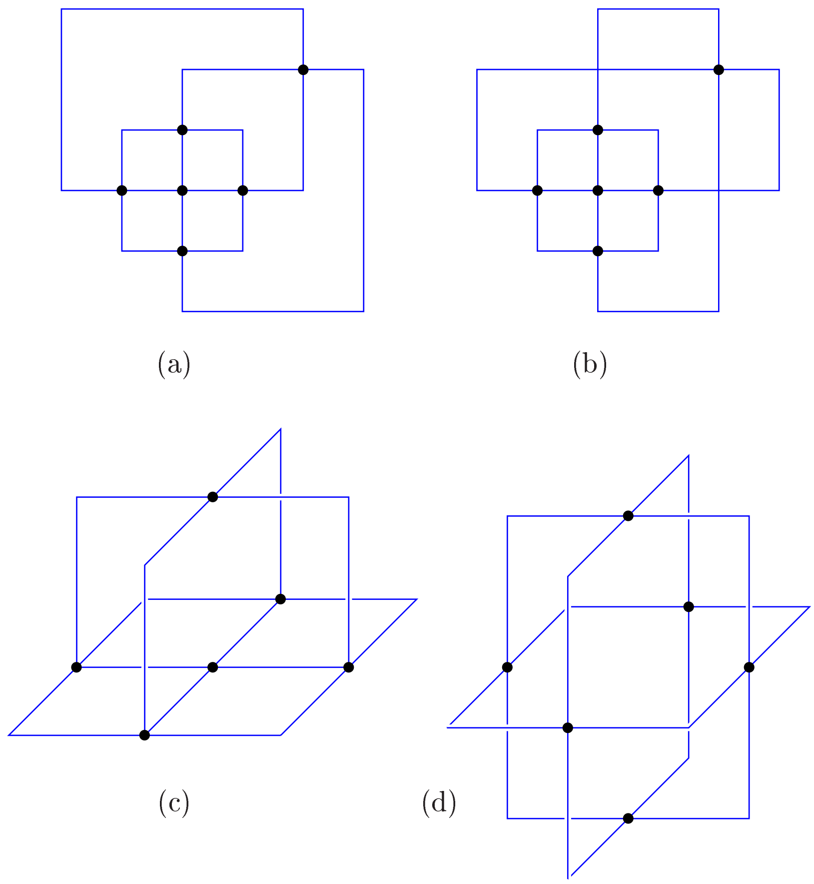


Figure 1.8: Orthogonal drawings of the octahedron graph: (a) 3-bend plane, (b) 2-bend planar with crossings, (c) 3-D with few bends and small volume, (d) 3-D orientation-independent.

While other applications, such as 3-D VLSI, may make different demands on the size and shape of vertices, we shall take the view that the following criteria are desirable features of orthogonal box-drawings.

- Vertex surface area is proportional to vertex degree.

- Vertices have bounded aspect ratio.

This thesis is concerned with the development of algorithms for orthogonal graph drawing. In Chapter 3 we survey existing algorithms and models for producing orthogonal graph drawings.

1.4.4 Other 3-D Graph Drawing Conventions

Three-dimensional graph drawings in the following styles have also been considered.

- Convex drawings [56, 80].
- Spline curve drawings [110].
- Multilevel drawings of clustered graphs [79, 97].
- Upward drawings [160].

1.5 Contributions and Outline of this Thesis

In this thesis we present and analyse methods for the generation of orthogonal graph drawings, concentrating on algorithms for producing 3-D drawings. We now outline the structure of this thesis and summarise the principal results obtained. Figure 1.9 illustrates this structure, highlighting the relationships between various parts of this thesis.

Part I: Orthogonal Graph Drawing

- Chapter 1 provides a broad overview of graph drawing, providing the motivation for the results presented in the remainder of this thesis.
- Chapter 2 introduces definitions and the notation used in this thesis.
- Chapter 3 surveys the existing results for orthogonal graph drawing, and compares these results with those presented in this thesis.

Part II: General Position Orthogonal Graph Drawing

- Chapter 4 presents heuristic and local minimum methods for solving the so-called balanced ordering problem. This one-dimensional problem is used as a basis for a number of 2-D and 3-D graph drawing algorithms presented in subsequent chapters.

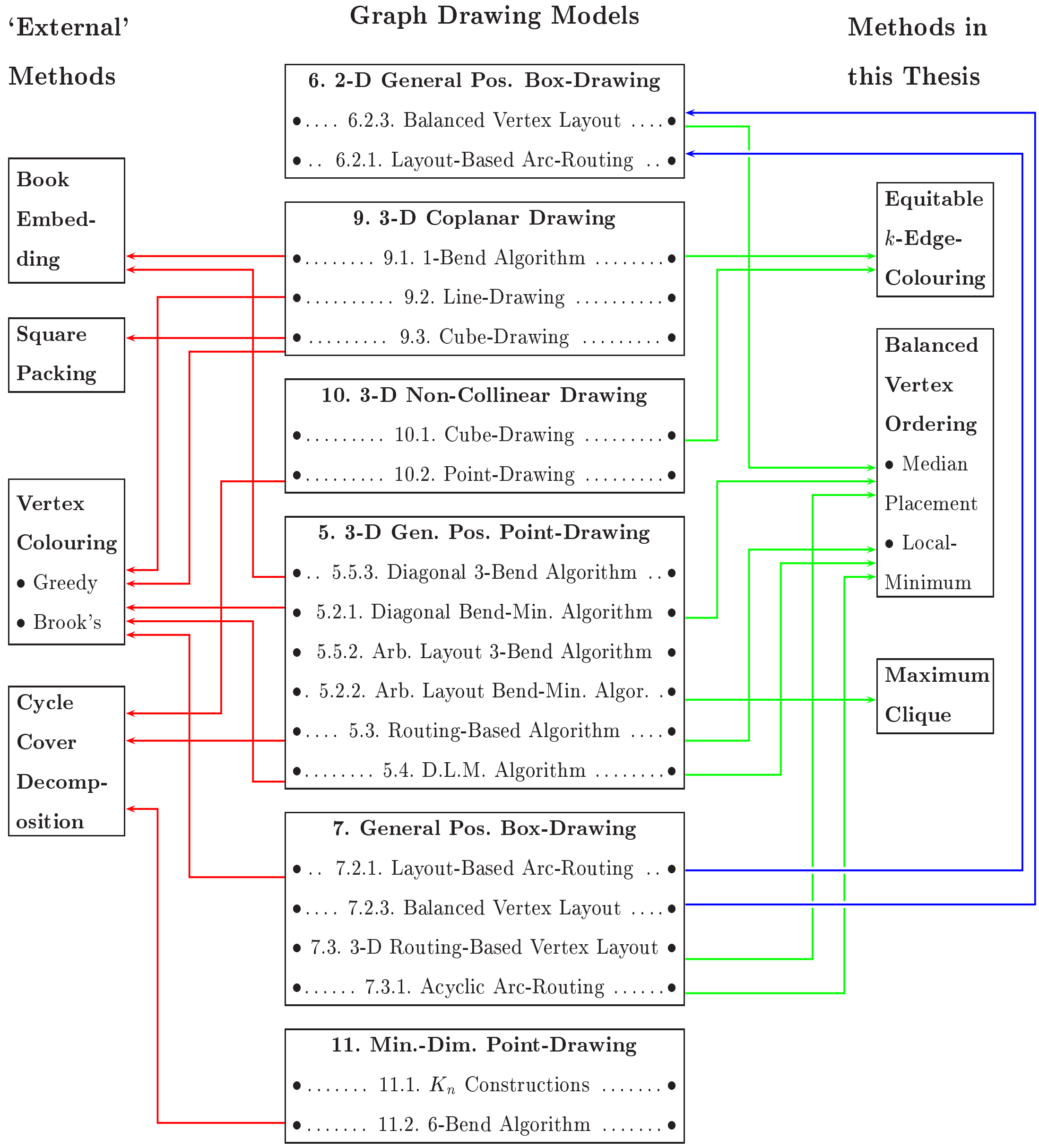


Figure 1.9: Dependence between sections of this thesis.

- Chapter 5 develops the general position layout model for 3-D orthogonal point-drawing. Achievements include an algorithm for minimising the total number of bends in diagonal layout 3-D orthogonal point-drawing (Section 5.2.1), establishing the best known upper bound for the total number of bends in 3-D orthogonal point-drawings (Section 5.4), and proving the best known upper bound for the volume of 3-bend 3-D orthogonal point-drawings (Section 5.5.3).

- Chapter 6 develops an algorithm for 2-D orthogonal graph drawing in the general position model which establishes the best known upper bound for the degree-restriction of vertices. This algorithm is generalised to multi-dimensional orthogonal graph drawing in Chapter 7.

- Chapter 7 develops the general position model for multi-dimensional orthogonal box-drawing, establishing the best known bound for the degree-restriction of 3-D orthogonal box-drawings.

Part III: Other Orthogonal Graph Drawing Models

- Chapter 8 provides an algorithm for equitable edge-colouring of multigraphs. This algorithm is used in the graph drawing algorithms presented in Section 9.1 and Chapter 10.

- Chapter 9 develops the coplanar vertex layout model for 3-D orthogonal drawing, providing algorithms for producing 3-D orthogonal box-drawings with one bend per edge route (Section 9.1), 3-D orthogonal box-drawings with optimal volume for regular graphs (Section 9.2), and degree-restricted 3-D orthogonal cube-drawings with optimal volume (Section 9.2).

- Chapter 10 introduces the non-collinear vertex layout model for producing orientation-independent 3-D orthogonal point-drawings with optimal volume, and 3-D orthogonal box-drawings with optimal volume for regular graphs.

- Chapter 11 presents an algorithm for multi-dimensional point-drawing with a bounded number of bends per edge route.

Part IV: Conclusion

- Chapter 12 summarises the main achievements of this thesis, the open problems in 3-D orthogonal graph drawing which have been identified, and discusses avenues for future work in 3-D graph drawing.

Part V: Appendices

- Appendix A provides the only known non-trivial lower bounds for the total number of bends in 3-D orthogonal point-drawings.
- Appendix B presents a number of 3-D orthogonal point-drawings with two bends per edge route. Some of these drawings were found using the algorithm for finding maximum cliques presented in Appendix C.
- Appendix C presents an algorithm for the maximum clique problem and provides an extensive experimental analysis of its performance. This algorithm which is of independent interest, has been applied to the search for 2-bend orthogonal point-drawings (see Section 5.2.2).

1.6 Publications

Much of the material in this thesis has appeared or will appear in the following publications.

Journal Publications:

- An Algorithm for Finding a Maximum Clique in a Graph, *Oper. Res. Lett.*, 21(5), pages 211-217, 1997. [218]
- (with T. Biedl and T. Thiele) Three-Dimensional Orthogonal Graph Drawing with Optimal Volume, submitted. (see [34])
- (with T. Biedl and M. Kaufmann) Area-Efficient Algorithms for Orthogonal Graph Drawing, in preparation. (see [30, 222])
- (with T. Biedl) Three-Dimensional Orthogonal Graph Box-Drawing with Few Bends, in preparation. (see [27, 222])
- Algorithms for Three-Dimensional Orthogonal Graph Drawing in the General Position Model, in preparation. (see [220, 221])

- Lower Bounds for the Number of Bends in Three-Dimensional Orthogonal Graph Drawings, in preparation. (see [224])

Refereed Conference Publications:

- (with T. Biedl and T. Thiele) Three-Dimensional Orthogonal Graph Drawing with Optimal Volume, In J. Marks (ed.), *Proc. 8th International Symposium on Graph Drawing* (GD'00), Lecture Notes in Comput. Sci., to appear. [34]

- Lower Bounds for the Number of Bends in Three-Dimensional Orthogonal Graph Drawings, In J. Marks (ed.), *Proc. 8th International Symposium on Graph Drawing* (GD'00), Lecture Notes in Comput. Sci., to appear. [224]

- Multi-Dimensional Orthogonal Graph Drawing with Small Boxes, In J. Kratochvil (ed.), *Proc. 7th International Symp. on Graph Drawing* (GD'99), Lecture Notes in Comput. Sci., vol. 1731, pages 311-322, Springer, 1999. [222][2]

- A New Algorithm and Open Problems in Three-Dimensional Orthogonal Graph Drawing, In R. Raman, J. Simpson (eds.), *Proc. 10th Australasian Workshop on Combinatorial Algorithms* (AWOCA'99), pages 157-167, Curtin University of Technology, Perth, 1999. [223]

- An Algorithm for Three-Dimensional Orthogonal Graph Drawing, In S. Whitesides (ed.), *Proc. of Graph Drawing : 6th International Symp.* (GD'98), Lecture Notes in Comput. Sci., vol. 1547, pages 332-346, Springer, 1998. [221]

- Towards a Two-Bends Algorithm for Three-Dimensional Orthogonal Graph Drawing, In V. Estivill-Castro (ed.), *Proc. 8th Australasian Workshop on Combinatorial Algorithms* (AWOCA'97), pages 102-107, Queensland University of Technology, 1997. [220]

- On Higher-Dimensional Orthogonal Graph Drawing, In J. Harland (ed.), *Proc. of Computing: the Australasian Theory Symp.* (CATS'97), pages 3-8, Macquarie University, 1997. [219]

[2]Awarded the best student paper prize at GD'99.

Chapter 2

Preliminaries

In this chapter we introduce definitions and the notation used in this thesis. Undefined terms from graph theory can be found in Chartrand and Lesniak [53], and from graph drawing in Di Battista et al. *[71].*

2.1 Graphs

Throughout this thesis $G = (V, E)$ is a graph with vertex set $V(G) = V$ and edge set $E(G) = E$. We assume G is undirected unless explicitly called a digraph. Graphs and digraphs are simple; i.e., there are no parallel edges, although a digraph may have a 2-cycle. A multigraph allows parallel edges but no loops, while a pseudograph is a multigraph possibly with loops. We denote the number of vertices of a graph G by $n = |V(G)|$ and the number of edges of G by $m = |E(G)|$. For a (di)graph G, the set of vertices $\{w : vw \in E(G)\}$ adjacent to a vertex $v \in V(G)$ is denoted by $V_G(v)$, and the set of (outgoing) edges $\{vw \in E(G)\}$ incident with v is denoted $E_G(v)$. The *(out)degree* $|G(v)|$ of a vertex $v \in V(G)$ is denoted $(\text{out})\deg(v)$. G has maximum (out)degree $\Delta(G)$. The subgraph of G induced by $S \subseteq V(G)$ is denoted $G[S]$.

Associated with any graph G is the digraph $\overleftrightarrow{G}$ with vertex set $V(\overleftrightarrow{G}) = V(G)$ and arc set $E(\overleftrightarrow{G}) = \{(v, w), (w, v) : \{v, w\} \in E(G)\}$. We denote $E(\overleftrightarrow{G})$ by $A(G)$. The arc $(v, w) \in A(G)$ is called the *reversal* of (w, v). The set of outgoing arcs $\{(v, w) \in A(G)\}$ at a vertex $v \in V(G)$ is denoted by $A^+_G(v)$ or simply $A_G(v)$, and set of incoming arcs $\{(w, v) \in A(G)\}$ at v is denoted by $A^-_G(v)$. For ease of notation, vw refers to the undirected edge $\{v, w\}$, and $\overrightarrow{vw}$ may refer to the directed edge (v, w) or the arc $(v, w) \in A(G)$ (for some graph G).

2.2 Cliques and Colourings

A *clique* of a graph is a set of pairwise adjacent vertices; i.e., a clique induces a complete subgraph. In Appendix C we present an algorithm for finding a clique of maximum size in a given graph.

A (*proper*) *vertex-colouring* of a graph is an assignment of colours, usually represented by positive integers, to the vertices such that adjacent vertices receive different colours. A vertex-colouring with k colours is called a *vertex k-colouring*.

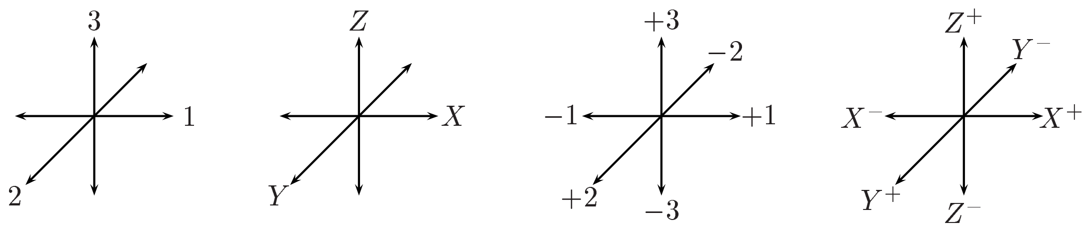
A sequential greedy strategy for vertex-colouring a graph is to assign to each vertex, in turn, the minimum colour not assigned to an adjacent vertex (see for example Biggs [35]). This is equivalent to assigning the first colour to every vertex available; repeating for the second colour, and so on, until all the vertices are coloured. This algorithm, which we call Greedy Vertex-Colour, applied to a graph G uses at most $\Delta(G) + 1$ colours.

An *edge-colouring* of a graph is an assignment of colours to the edges. If all edges incident to a common vertex receive different colours then the edge-colouring is *proper*.

Suppose $\text{col} : X \rightarrow C$ is a colouring of some class of objects X, e.g., vertices, edges or arcs. We denote the colour class of objects receiving some colour $c \in C$ by $X[c]$; i.e., $X[c] = \{x \in X : \text{col}(x) = c\}$. In particular, if $A(G)$ is coloured, then $\overleftrightarrow{G}[i]$, for some colour i, denotes the subgraph of $\overleftrightarrow{G}$ induced by the arcs coloured i.

2.3 Orthogonal Grid

The *D-dimensional orthogonal grid* ($D \geq 2$) is the D-dimensional cubic lattice, consisting of *grid-points* in $\mathbb{Z}^D$, together with the coordinate-axis-parallel *grid-lines* determined by these points. A positive integer i, $1 \leq i \leq D$, used to index the coordinates of a grid-point in $\mathbb{Z}^D$, is called a *dimension*, and a non-zero integer d, $1 \leq |d| \leq D$, is called a *direction*, as illustrated in Figure 2.1. For $D = 2$ and $D = 3$, we also refer to the dimensions as $\{X, Y\}$ and $\{X, Y, Z\}$, and directions as $\{X^{\pm}, Y^{\pm}\}$ and $\{X^{\pm}, Y^{\pm}, Z^{\pm}\}$, respectively.



(a) Dimensions (b) Directions

Figure 2.1: Dimensions and directions in the 3-D orthogonal grid.

The $(i = K)$-hyperplane, for some dimension i, $1 \leq i \leq D$, and integer $K \in \mathbb{Z}$, is called a *grid-hyperplane*. For $D = 3$ a grid hyperplane is called a *grid-plane*. For each dimension i, $1 \leq i \leq D$, a grid-line parallel to the i-axis is called an *i-line*, and a grid-(hyper)plane perpendicular to the i-axis is called an *i-(hyper)plane*.

A *grid-box* B in the D-dimensional orthogonal grid is a region

$$\left\{(a_1, a_2, \ldots, a_D) \in \mathbb{R}^D : l_i(B) \leq a_i \leq r_i(B), 1 \leq i \leq D\right\} \ .$$

for some $l_i(B), r_i(B) \in \mathbb{Z}$, $1 \leq i \leq D$. The grid-points $(l_1(B), l_2(B), \ldots, l_D(B))$ and $(r_1(B), r_2(B), \ldots, r_D(B))$ are referred to as the *minimum corner* and *maximum corner* of B, respectively. The *size* of B is $\alpha_1(B) \times \alpha_2(B) \times \cdots \times \alpha_D(B)$ where $\alpha_i(B) = r_i(B) - l_i(B) + 1$. Note that $\alpha_i(B)$ is the *not* the actual side length of B in dimension i. This convention enables us to consistently speak of the *volume* (and *area* in two dimensions) of a possibly degenerate grid-box as the number of grid-points in the box; i.e.

$$\text{volume}\,(B) = \prod_{1 \leq i \leq D} \alpha_i(B) \ .$$

For a two-dimensional $\alpha_X \times \alpha_Y$ box, the side lengths α_X and α_Y are called the *width* and *height* of the box, respectively. For a three-dimensional $\alpha_X \times \alpha_Y \times \alpha_Z$ box, the side lengths α_X, α_Y and α_Z are called the *width*, *depth* and *height* of the box, respectively.

For each direction d, $1 \leq |d| \leq D$, the set of grid-points in a grid-box B which are extremal in direction d is called the *d-face* of B. At each grid-point on the d-face of a box we say there is a *port*. A port is considered to extend out from the surface of the box *in direction* d, as illustrated in Figure 2.2.

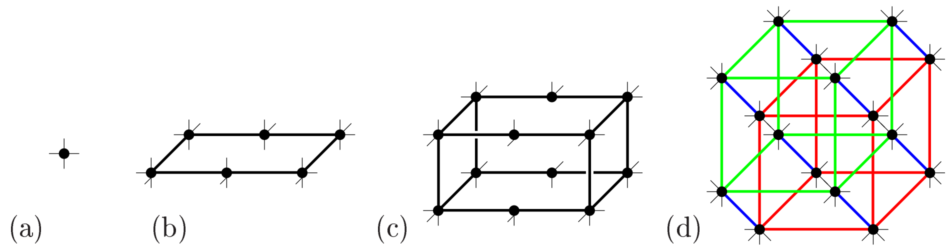


Figure 2.2: Ports on grid-boxes:

(a) 1×1 2-D point with volume 1 and surface 4,

(b) $3 \times 2 \times 1$ 3-D rectangle with volume 6 and surface 22,

(c) $3 \times 2 \times 2$ 3-D box with volume 12 and surface 32,

(d) $2 \times 2 \times 2 \times 2$ 4-D hyperbox with volume 16 and surface 64.

A port in direction d, $1 \leq |d| \leq D$, is called a d*-port*, and for any dimension i, $1 \leq i \leq D$, a $(\pm i)$-port is also called an i*-port*. The number of ports on the (i^+)-face of B (which obviously equals the number of ports on the (i^-)-face) is referred to as the $\text{surface}_i(B)$; i.e.,

$$\text{surface}_i(B) = \prod_{\substack{1 \leq j \leq D \\ j \neq i}} \alpha_j(B) \quad .$$

The total number of ports on B is the $\text{surface}(B)$; i.e.,

$$\text{surface}(B) = 2 \sum_{1 \leq i \leq D} \text{surface}_i(B) \quad .$$

2.4 Orthogonal Graph Drawing

A *D-dimensional orthogonal drawing* of a graph G, called an *orthogonal drawing*, represents each vertex $v \in V(G)$ by a grid box B_v such that

$$\forall v, w \in V(G),\ v \neq w \Rightarrow B_v \cap B_w = \emptyset \quad .$$

The graph-theoretic term 'vertex' will also refer to the corresponding box. Allowing vertices to degenerate to rectangles or lines is the approach taken in [27, 32, 33, 222, 223], but not in [166, 168]; enlarging vertices to remove this degeneracy increases the volume by a multiplicative constant.

A *grid-polyline* in the D-dimensional orthogonal grid is a polyline consisting of contiguous segments of grid-lines, possibly bent at grid-points. An orthogonal drawing of G represents each edge $vw \in E(G)$ by a grid-polyline, called an *edge route*, between grid-points on the boundaries of B_v and B_w, not intersecting any vertices except at these boundary points. The interior of edge routes are pairwise non-overlapping, and only for $D = 2$ are edge routes allowed to cross. A segment of an edge route parallel to the i-axis, for some dimension i, is called an *i-segment*.

Two-dimensional and three-dimensional orthogonal drawings are called *2-D* and *3-D* orthogonal drawings, respectively. A 2-D orthogonal drawing without edge crossings is a *plane 2-D orthogonal drawing*.

Port Assignment and Routings

An orthogonal drawing of a graph G assigns each arc $\overrightarrow{vw} \in A(G)$ a unique port at v, referred to as the *port*$(\overrightarrow{vw})$. The set of ports at a vertex v is denoted by *ports*(v), and we define *ports*(G) to be the set of ports of a graph G; i.e.,

$$\text{ports}\,(G) = \bigcup_{v \in V(G)} \text{ports}\,(v) \quad .$$

If, in a D-dimensional orthogonal drawing of a graph G, for some vertices $v, w \in V(G)$ and dimension i, $1 \leq i \leq D$, the (i^+)-face of v has i-coordinate less than the i-coordinate of the (i^-)-face of w then we say w is *in direction i^+ from v*, v is *in direction i^- from w*, an (i^+)-port at v *points toward* w, and an (i^-)-port at v *points away from* w.

If for some arc $\overrightarrow{vw} \in A(G)$ and dimension i, $1 \leq i \leq D$, the port$(\overrightarrow{vw})$ is an i-port then we consider $\overrightarrow{vw}$ to be *coloured i*. In this manner a D-dimensional orthogonal drawing of a G determines a D-colouring of $A(G)$. We call a D-colouring of $A(G)$ a (*D-dimensional*) *routing* of $A(G)$. An orthogonal drawing is *routing-preserving* if the drawing determines a given routing.

For point-drawings, at each vertex v and direction d, there is exactly one port at v in direction d. We denote this port by *port*(v, d). We say *port*(v, d) is *opposite* to

$port(v, -d)$, for each vertex v and direction d. A D-dimensional orthogonal point-drawing of G determines a routing with at most two outgoing arcs at each vertex receiving the same colour; i.e., $|A_G(v)[i]| \leq 2$ for every vertex v and dimension i, $1 \leq i \leq D$. We call a routing with this property a (D-*dimensional*) *point-routing* of $A(G)$.

Note that a routing of a graph G does not fully describe the edge routes in an orthogonal drawing of G. It merely describes the axes which the first and last segments of each edge route are parallel to. In the general position model (see Chapters 6, 5 and 7), we show that a routing suffices as a data structure for representing the edge routes.

Aesthetic Criteria

We now make precise definitions for the criteria by which we measure the aesthetic quality of an orthogonal box-drawing. The minimum-sized box enclosing an orthogonal drawing is called the *bounding box* of the drawing. We refer to the volume of the bounding box as the *volume* of the drawing. An orthogonal drawing with a maximum of b bends per edge route is called a *b-bend orthogonal drawing*. An orthogonal drawing with a particular "shape" of grid-box representing every vertex, e.g., point, line, rectangle, square, cube or hypercube, is called an orthogonal *shape*-drawing for each particular "shape", as illustrated in Figure 2.3.

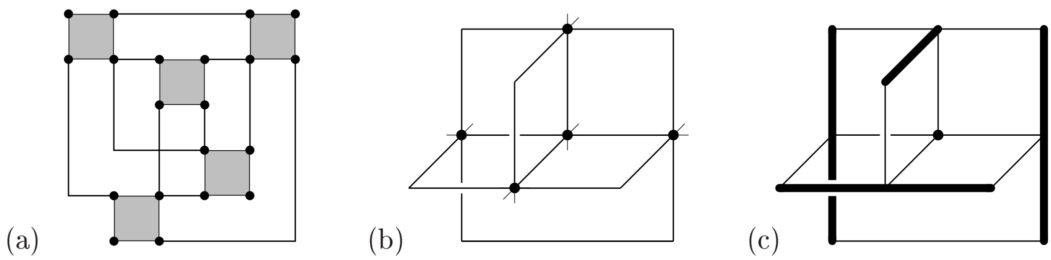


Figure 2.3: Orthogonal drawings of K_5: (a) 1-bend 2-D square-drawing, (b) 2-bend 3-D point-drawing, (c) 0-bend 3-D line-drawing.

A D-dimensional orthogonal drawing of a graph G is said to be *strictly α-degree-*

restricted if there exists a constant α such that for every vertex $v \in V(G)$,

$$\text{surface}\,(v) \leq \alpha \cdot \deg(v) \ \ .$$

Such a drawing is said to be *strictly degree-restricted*.

For some orthogonal graph drawing algorithm, the minimum α such that the drawings produced by the algorithm are strictly α-degree-restricted does not necessarily reflect the asymptotic relationship between the surface and the degree of the vertices. We therefore say that in an orthogonal drawing of a graph G, a vertex $v \in V(G)$ is α-*degree-restricted* if

$$\text{surface}\,(v) \leq \alpha \cdot \deg(v) + o\,(\deg(v)) \ \ .$$

If for some constant α, every vertex $v \in V(G)$ is α-degree-restricted, then the drawing is said to be (α)-*degree-restricted*. This definition enables us to compare the asymptotic behaviour of α for various algorithms.

Clearly, if a drawing is strictly degree-restricted then it is also degree-restricted. Conversely, it is easily seen that all degree-restricted drawings produced by algorithms presented in this thesis are also strictly degree-restricted. Hence for our purposes the two notions coincide, although one can contrive examples where this is not the case. It is necessary to distinguish the two terms as the lower bound in Theorem 3.2 is for strictly degree-restricted drawings.

The *aspect ratio* of a vertex v is:

$$\text{aspect ratio}\,(v) \ = \ \left(\max_{1 \leq i \leq D} \ \alpha_i(v)\right) \Big/ \left(\min_{1 \leq i \leq D} \ \alpha_i(v)\right) \ \ .$$

A hypercube has aspect ratio one, while a $k \times 1 \times 1 \times \cdots \times 1$ line has aspect ratio equal to k.

2.5 Cycle Cover Decomposition

A *cycle cover* of a digraph is a spanning subgraph consisting of directed cycles. We now describe an algorithm for the decomposition of a graph into cycle covers. This algorithm will often form the preprocessing step in the graph drawing algorithms to come. This step was first used by Eades *et al.* [86] in their 3-D orthogonal point-drawing algorithm for maximum degree six graphs. The following generalisation to arbitrary degree graphs can be found in [87, 219]. The result can be considered as repeated application of the classical result of Petersen that "every regular graph of even degree has a 2-factor" [172].

Theorem 2.1. *If G is a multigraph and $d = \lceil \Delta(G)/2 \rceil$ then there exists a directed multigraph G' such that:*

1. *G is a subgraph of the underlying undirected multigraph of G'.*
2. *Each vertex of G' has in-degree d and out-degree d.*
3. *The arcs of G' can be partitioned into d edge-disjoint cycle covers.*

G' and the edge-disjoint cycle covers can be computed in $O(\Delta^2 n)$ time.

Proof. Initially let $G' = G$. The number of vertices of odd degree in any multigraph must be even. So that each vertex of G' has even degree we pair the odd degree vertices and add an edge between each pair. For each vertex $v \in V(G')$, add $d - \deg(v)/2$ self-loops to v, to create a $2d$-regular pseudograph. Since each vertex of G' has even degree it is Eulerian. Direct the edges of G' by following an Eulerian tour through G'. Each vertex of G' now has in-degree d and out-degree d.

For each vertex $v \in V(G')$, define $V_{out} = \{v_{out} : v \in V(G')\}$, $V_{in} = \{v_{in} : v \in V(G')\}$, where $v_{out} = \{w \in V(G') : \overrightarrow{vw} \in E(G')\}$ and $v_{in} = \{u \in V(G') : \overrightarrow{uv} \in E(G')\}$. Now construct an undirected bipartite graph H with $V(H) = V_{out} \cup V_{in}$, and $E(H) = \{\{u_{out}, v_{in}\} : (u, v) \in E(G')\}$.

Since H is d-regular and bipartite, by Hall's Theorem [114], H contains a perfect matching; colour its edges 1 and remove them. The remaining graph is $(d-1)$-regular and bipartite, so it also contains a perfect matching; colour its edges 2 and remove them. Continue this process, to create d edge-disjoint perfect matchings in H. Colouring each arc $\overrightarrow{uv} \in E(G')$ the same colour given to $\{u_{out}, v_{in}\}$ in H gives each node of G' exactly one incoming arc and one outgoing arc for each colour. Hence the arcs of G' are partitioned into d distinct subgraphs $C_1, C_2 \ldots, C_d$, corresponding to each colour $1, 2, \ldots, d$, each of which is a cycle cover for G'. This partition into perfect matchings is sometimes referred to as *König's Theorem* [133].

Schrijver [194] describes an algorithm for determining all perfect matching of a k-regular n-vertex bipartite graph in $O(k^2 n)$ time. H is d-regular with $2n$ vertices, so the calculation of the perfect matchings which form the partition of H, which is the most time-consuming stage of the algorithm, takes $O(\Delta^2 n)$ time. $\square$

Chapter 3

Approaches to Orthogonal Graph Drawing

In this chapter we survey existing results for orthogonal graph drawing, describing the models and algorithms employed for the production of such drawings, and compare these results with those presented in this thesis.

This chapter is organised as follows. Section 3.1 reviews the known NP-hardness results for the optimisation of various aesthetic criteria in orthogonal graph drawings. 2-D orthogonal graph drawing is surveyed in Section 3.2, including an introduction to the general position model for 2-D orthogonal graph drawing which is the model employed in Chapter 6. Table 3.1 summarises the known bounds, including those presented in this thesis, for 2-D orthogonal drawings possibly with crossings. We then consider orthogonal graph drawing on surfaces (other than the plane) in Section 3.3.

Section 3.4 surveys models and algorithms for 3-D orthogonal graph drawing, and introduces the algorithms presented in this thesis. In Section 3.5 we conclude with a discussion of the known bounds and principal open problems for 3-D orthogonal graph drawing. Tables 3.2 and 3.3 summarise the known bounds for aesthetic criteria of 3-D orthogonal point-drawings and 3-D orthogonal box-drawings, respectively.

3.1 Complexity

It is NP-hard to optimise many of the aesthetic criteria for orthogonal graph drawings discussed in Chapter 1. In particular, for a given maximum degree four graph, minimising each of the following aesthetic criteria is NP-hard for 2-D orthogonal point-drawing.

- Total number of bends (Garg and Tamassia [106]).
- Bounding box area
  (Dolev *et al.* [78], Storer [199], Kramer and van Leeuwen [135]).
- Maximum edge length (Bhatt and Cosmadakis [21], Gregori [111]).

Garg and Tamassia [106] establish that it is NP-hard to even approximate the minimum number of bends in a planar graph with $O\left(n^{1-\epsilon}\right)$ error, for any $\epsilon > 0$. Shermer [195] shows that it is NP-complete to recognise weak rectangle visibility graphs (see Section 3.2.1), and hence it is NP-hard to minimise the number of bends in a 2-D orthogonal box-drawing of a given graph.

Using straightforward extensions of the corresponding 2-D NP-hardness results, Eades *et al.* [85] show that it is NP-hard to minimise each of the following aesthetic criteria in a 3-D orthogonal point-drawings.

- Bounding box volume.
- Total number of bends.
- Total edge length.

These methods can be applied with the NP-completeness result of Shermer [195] discussed above to show that it is NP-hard to minimise the total number of bends in a 3-D orthogonal box-drawing of a given graph.

3.2 2-D Orthogonal Drawings

Algorithms for producing 2-D orthogonal drawings have been extensively studied in the literature. We now discuss the principal approaches employed.

3.2.1 Visibility Approach

Plane Drawings

Plane orthogonal drawings with straight-line edge routes (with no bends) are aesthetically very pleasing since the relational information represented in the graph is clearly expressed. A closely related idea to that of a straight-line orthogonal drawing is that of a visibility representation. A (*weak*) *visibility representation* of a graph G represents each vertex $v \in V(G)$ by a horizontal segment in the plane, and represents each edge $vw \in E(G)$ by a vertical segment between the horizontal segments representing v and w and not intersecting any other horizontal segments. A graph admitting a visibility representation is clearly planar. Tamassia and Tollis [202] and Rosenstiehl and Tarjan [187] independently show that every planar graph has a visibility representation, and hence a straight-line orthogonal drawing, which can be computed in linear time.

Various types of visibility representations can be defined, depending on whether vertices are segments or intervals and whether visible vertices must be adjacent. Tamassia and Tollis [202] and Wismath [216] characterise those planar graphs which admit each possible type. The disadvantage of the visibility representation method for producing plane orthogonal drawings is that the vertices are not necessarily degree-restricted and have high aspect ratio.

Drawings with Crossings

In a *(weak) rectangle visibility representation* of a graph, vertices are represented by rectangles, and adjacent vertices can 'see' each other by some axis-aligned 'band of visibility' not intersecting any other vertex (see Dean and Hutchinson [67] for precise definitions). It follows that a graph has a straight-line 2-D orthogonal box-drawing if and only if it has a weak rectangle visibility representation. The subgraphs induced by the horizontal and vertical edges of such a graph are planar, so the graph has thickness at most two. Bose *et al.* [37] establish that numerous classes of graphs with thickness two admit straight-line 2-D orthogonal box-drawings. Since K_9 has thickness three (see Beineke [16]), the straight-line 2-D orthogonal drawing of K_8 presented by Dean and Hutchinson [67] is the largest complete graph admitting such a drawing. $K_{5,6}$ has a straight-line 2-D orthogonal box-drawing, as shown in Figure 3.1.

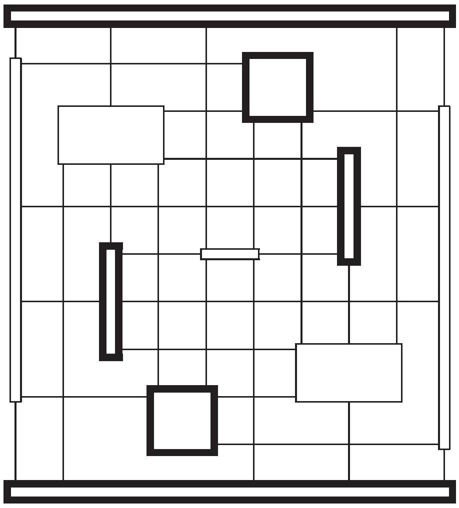


Figure 3.1: Straight-line 2-D orthogonal drawing of $K_{5,6}$.

Even though $K_{5,n}$ $(7 \leq n \leq 12)$ and $K_{6,n}$ $(6 \leq n \leq 8)$ have thickness two [17], it is unknown if these graphs admit 2-D straight-line orthogonal box-drawings. We conjecture that $K_{5,7}$ and $K_{6,6}$ do not admit such drawings. Bose *et al.* [37] show that $K_{4,n}$ $(n \geq 1)$ has a 2-D straight-line orthogonal box-drawing.

3.2.2 Topology-Shape-Metrics Approach

A number of algorithms for 2-D orthogonal graph drawing can be grouped under the so called *topology-shape-metrics approach* approach (see Di Battista *et al.* [71, chap. 5]). These methods consist of the following three main steps.

Planarisation: Determine a planar embedding of the graph with few crossings, and represent each crossing by a dummy vertex.

Orthogonalisation: Determine the *shape* of the drawing.

Compaction: Determine the coordinates of the vertices and bends to minimise the area.

The development of these algorithms can be traced to the classical algorithm of Tamassia [200] for determining a bend-minimum orthogonal point-drawing which preserves a given planar embedding of a graph with maximum degree four (see also Batini *et al.* [14]). This algorithm models the bend-minimisation problem using network flow techniques, and takes $O(n^2 \log n)$ time (subsequently improved to $O(n^{7/4}\sqrt{\log n})$ by Garg and Tamassia [107]). Biedl [26] has since obtained bounds on the area and the number of bends for this algorithm.

Tamassia *et al.* [201] present the GIOTTO algorithm for orthogonal drawing of non-planar graphs of arbitrary degree, which is based on Tamassia's algorithm for planar graphs. To cater for arbitrary degree vertices, each vertex v of degree $d \geq 4$ is replaced by a cycle of d vertices where each vertex of the cycle is incident to one of the edges formally incident to v, as illustrated in Figure 3.2. Experimental results confirming the success of this approach are reported in Di Battista *et al.* [72].

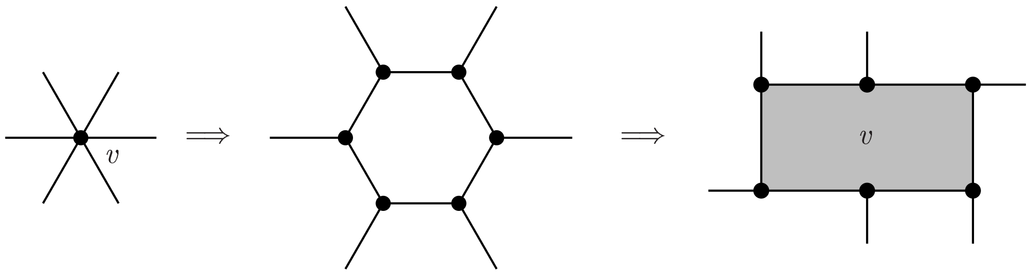


Figure 3.2: Replacing v by a cycle.

The KANDINSKY model for 2-D orthogonal drawings, which has been investigated by Fößmeier and Kaufmann [103, 104] and Fößmeier *et al.* [102], consists of a 2-D (sparse) grid with uniform distance λ between the grid lines. The vertices have side length less than λ, and the centres of the vertices are placed at the intersection of the grid lines; this ensures that no vertex is intersected by any grid line except those defining its position, and censequently no two vertices intersect. Edges are routed on the underlying orthogonal grid. Under the assumption that vertices are represented by uniformly small squares and that each face is a non-empty region, the algorithm in [103], given a planar graph embedding, minimises the number of bends in a 2-D orthogonal drawing in the KANDINSKY model. Fößmeier and Kaufmann [104] extend the KANDINSKY model to cater for non-planar graphs and to remove the requirement in [103] that vertices have the same size.

In recent developments the algorithm of Di Battista *et al.* [70] determines an embedding-preserving 2-D orthogonal drawing where the size of each vertex is specified by the user. The drawings produced have the minimum number of bends among a wide class of drawings.

Di Battista *et al.* [73] introduce the notion of *spirality* of planar orthogonal point-drawings and explore the connection between spirality and the number of bends. In particular, they present polynomial time algorithms for determining bend-minimum orthogonal point-drawings for series-parallel graphs and for planar graphs of maximum degree three. Bertolazzi *et al.* [20] and Didimo and Liotta [75] use advanced data structures to represent all the planar embeddings of a given graph in their algorithms to determine bend-minimum 2-D orthogonal drawing. Their algorithms run in time exponential in the number of vertices with degree greater than four.

3.2.3 Geometric Approach

We now describe algorithms for orthogonal graph drawing which are purely geometric, as opposed to the algorithms described above which are based on topological embeddings. Bertolazzi *et al.* [20] calls this the *draw-and-adjust* approach.

Plane Point-Drawings

Numerous algorithms have been proposed in the literature for drawing planar orthogonal point-drawings. Algorithms for drawing cubic graphs include those of Papakostas and Tollis [163], Rahman *et al.* [178], Calamoneri and Petreschi [50, 51] and Biedl [23]. For maximum degree four graphs, algorithms include those of Tamassia and Tollis [203], Liu *et al.* [146], Kant [124], Biedl [24] and Biedl and Kant [29]. We now outline two of the approaches used by these algorithms.

The algorithm of Tamassia and Tollis [203] for 2-D orthogonal point-drawing of planar graphs, is based on a visibility representation of the given graph. The horizontal segments representing vertices in the visibility representation are replaced by points and bends are added to the edge routes. The algorithm, which runs in linear time, produces 2-D orthogonal plane drawings with $O(n^2)$ area, at most four bends per edge route, and a total of at most $12n/5 + 2$ bends.

The algorithm of Biedl and Kant [29], for a biconnected graph G of maximum degree four, determines in linear time an orthogonal point-drawing with at most $2n + 2$ bends and $n \times n$ bounding box. Every edge has at most two bends (unless G is the octahedron graph which is shown by Even and Granot [91] not to have a 2-bend plane orthogonal point-drawing; see Figure 1.8(a)). This algorithm is based on an st-ordering of the vertices (see Section 4.2). A modified algorithm determines an orthogonal point-drawing of a connected graph G with at least one cut vertex with $(n-1) \times (n-1)$ bounding box, at most two bends per edge, and at most m bends in total. For triconnected graphs the algorithm of Kant [124], improved by Biedl [24], establishes an upper bound on the number of bends of $\lceil 4n/3 \rceil + 4$.

Point-Drawings with Crossings

Algorithms which do not guarantee plane drawings even for planar graphs have been considered by Schäffter [190] and Papakostas and Tollis [165]. The latter algorithm determines in linear time an orthogonal point-drawing of a given maximum degree four graph having area at most $0.76n^2$ and at most $2n + 2$ bends. Lower bounds for 2-D orthogonal point-drawing have been established by Tamassia *et al.* [205] and Biedl [25].

Plane Box-Drawings

Motivated by the desire to overcome the inherent restriction on the maximum degree of graphs admitting orthogonal point-drawings, there has been recent interest in the development of algorithms for 2-D orthogonal box-drawing.

Even and Granot [92] studied 2-D orthogonal box-drawings where the size of each vertex and the port assignments are given as part of the input. This approach is particularly applicable to VLSI layout problems where the components of the circuit have predefined sizes. They present two algorithms. The first, which is for planar drawings, is based on a visibility representation of the graph. The second algorithm employs a diagonal layout of the vertices. The drawings produced have at most four bends per edge and $(W+m) \times (H+m)$ bounding box, where W and H are respectively the total width and height of the boxes representing vertices.

Using the 'cycle of low degree vertices' method illustrated in Figure 3.2, the algorithm of Biedl and Kant [29] is extended to produce planar drawings of arbitrary degree planar graphs. The disadvantage of this approach is that the vertices are not necessarily degree-restricted. This algorithm can also cater for drawings of non-planar graphs.

Box-Drawings with Crossings

We now discuss box-drawing algorithms which are applicable to arbitrary graphs but do not guarantee a planar drawing even for planar graphs. This is the approach taken by the 2-D orthogonal box-drawing algorithm presented in Chapter 6. (In Chapter 7 this algorithm is generalised to a multi-dimensional setting.) Table 3.1 summarises the known upper bounds for this class of 2-D orthogonal graph drawings.

Table 3.1: Upper Bounds for 2-D Orthogonal Box-Drawing

Box Shape	Area	Max. Bends	Degree-Restriction	Aspect Ratio	Reference
line	$(m-1) \times (\frac{m+1}{2})$	1	2	$\leq \deg(v)/2$	[164, 169]
line	$(\frac{m+n}{2}) \times (\frac{m+n}{2})$	1	2	$\leq \deg(v)/2$	[30]
rectangle	$\left(\frac{3m+2n}{4}\right) \times \left(\frac{3m+2n}{4}\right)$	1	2	2	[30]
rectangle	$\left(\frac{3m+4n+2}{4}\right) \times \left(\frac{3m+4n+2}{4}\right)$	1	$\frac{3}{2}$	2	Theorem 6.3
square	$\left(\frac{3m}{4}+\frac{5n}{8}\right) \times \left(\frac{3m}{4}+\frac{5n}{8}\right)$	1	2	1	Theorem 6.4

The algorithms of Papakostas and Tollis [164, 169] and Biedl and Kaufmann [30] (which is an example of the unified approach to orthogonal graph drawing called the *three-phase method* [31]) were the first to produce degree-restricted 2-D orthogonal box-drawings. Each vertex v has aspect ratio at most $\deg(v)/2$ and each edge route has at most one bend. For sparse graphs ($m < (1+\sqrt{2})n$ to be precise), the algorithm in [164, 169] requires less area than that in [30]. A second algorithm in [30] produces drawings in which each vertex has aspect ratio at most two, at the expense of an increase in area.

The algorithm of Biedl and Kaufmann [30] produces drawings such that no two vertices are intersected by a single grid-line. We call such drawings *general position* 2-D orthogonal drawings. An introductory version of the algorithm of Papakostas and Tollis [164, 169] also produces general position 2-D orthogonal drawings; in a refined version certain pairs of vertices share a row or column.

The algorithms presented in Chapter 6 also produce general position 2-D orthogonal drawings. In Section 3.4.4 we introduce the general position model for D-dimensional orthogonal graph drawing and classify algorithms for producing such drawings as *layout-* or *routing-based*. The algorithms in [30] and [164, 169] can be classified as routing-based.

Maintaining the aspect ratio bound of two in [30], the layout-based algorithm presented in Section 6.2.3 produces 3/2-degree-restricted 2-D orthogonal drawings. Using a diagonal layout, our algorithm described in Section 6.2.4 produces 2-degree-restricted 2-D orthogonal square-drawings. Note that 2-D diagonal layouts have been employed by Even and Granot [91] and Schäffter [190]. Our bounding box area bounds are slightly above those in [30].

Interactive Drawing

As well as considering the aesthetic criteria already discussed for static orthogonal graph drawing, interactive graph drawing algorithms should 'preserve the mental map' of the viewer of the drawing when vertices and edges are inserted or deleted (see Misue *et al.* [154], for example). Interactive orthogonal point-drawing has been studied by Papakostas *et al.* [162], Fößmeier [100], Bridgeman *et al.* [44], Brandes and Wagner [42] and Papakostas and Tollis [167]. Biedl *et al.* [31] also describe how the three-phase method can be extended to an interactive setting.

3.3 Orthogonal Drawings on Surfaces

A natural, yet little studied generalisation of plane orthogonal drawings, is that of orthogonal drawings on surfaces. An embedding of a graph in an orientable surface other than the plane can be drawn in an *orthogonal surface*, as illustrated in Figure 3.3

(see Garrido and Márquez [109]). Consider the following open problem.

Problem 3.1. SURFACE POINT-DRAWING

Instance: An embedding Φ of a graph G (with maximum degree four) in the orientable surface of genus g, and a positive integer $B \in \mathbb{Z}^+$.

Question: Is there an orthogonal point-drawing of G in the orthogonal surface of genus g which preserves Φ and with at most B bends?

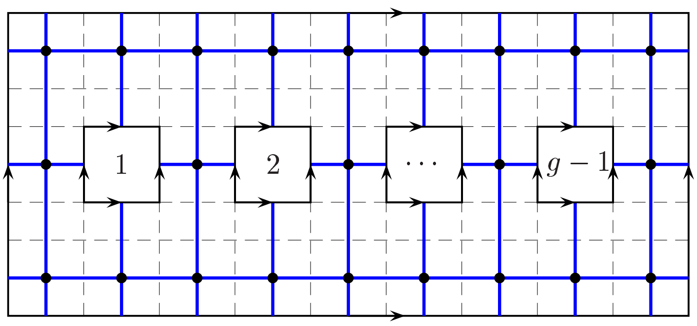


Figure 3.3: An orthogonal drawing of a graph in the surface of genus g.

Garrido and Márquez [109] sketch proofs, that for any fixed orientable surface S (except the plane), it is NP-complete to test whether a given graph embedding in S has an *essentially equivalent*[1] straight-line orthogonal point-drawing in an orthogonal surface corresponding to S. Hence minimising the number of bends in an orthogonal drawing essentially equivalent to a given embedding is NP-hard.

3.4 Models for 3-D Orthogonal Graph Drawing

In this section we survey models and algorithms for the generation of 3-D orthogonal graph drawings, including those presented in this thesis. We classify models for vertex layout by the minimum integers a and b, $1 \leq a, b \leq 2$ such that

- all vertices are intersected by a single a-dimensional orthogonal grid, and
- no two vertices are intersected by a single b-dimensional orthogonal grid.

[1]The term *essentially equivalent* is not precisely defined.

Here a 1-dimensional (respectively, 2-dimensional) grid refers to a grid-line (grid-plane) within the 3-dimensional orthogonal grid.

3.4.1 Visibility Representations

Visibility representations of graphs in the plane (see Section 3.2.1) naturally extend to three dimensions. In the so-called ZPR (**Z**-**P**arallel **R**epresentation) model for straight-line 3-D orthogonal graph drawing, each vertex is a rectangle parallel to the XY-plane, and edges are routed parallel to the Z-axis. Bose *et al.* [38] showed that there does not exist a ZPR of K_n for $n > 56$. The proof is based on deep results concerning unimaximal subsequences. They also found a ZPR of K_{22} using simulated annealing techniques. Representing vertices by squares of the same size, Fekete *et al.* [95] showed that K_7 has a ZPR, but K_n for $n \geq 8$ does not. The ZPR model was extended to arbitrary dimensions by Cobos *et al.* [59], establishing that every graph has a ZPR in some number of dimensions.

In a straight-line D-dimensional orthogonal graph drawing, the axis each edge is parallel to defines a edge D-colouring of the graph. As pointed out by Biedl *et al.* [32, 33] in the case of $D = 3$, each colour class induces a ZPR, so by the above K_{56} ZPR non-existence result, it follows that there does not exist a 3-D straight-line orthogonal drawing of K_n for n greater than the Ramsey number $R(56, 56, \ldots, 56)$ (with D 56's). In three dimensions this upper bound has been significantly improved to K_{184} by Fekete and Meijer [96] (their proof is still based on the non-existence of a ZPR of K_{56}). Based on the ZPR of K_{22} mentioned above, Fekete and Meijer also construct the largest known straight-line 3-D orthogonal drawing of a complete graph, namely K_{56}, and establish a number of bounds on the size of complete graphs admitting such drawings when the shape of the boxes and the number of different sized boxes is restricted[2].

This K_{56} construction immediately generalises to multiple dimensions, providing a straight-line D-dimensional orthogonal box-drawing of $K_{22(D-1)+12}$. For $D \geq 2$ and $n \geq 1$, the bipartite graph $K_{2D,n}$ has a D-dimensional orthogonal drawing without

[2]The lower bound of K_{56} for 3-D straight-line orthogonal drawings and the upper bound of K_{56} for ZPR's is a coincidence. Hitchhikers are disappointed that the previous best lower bound of K_{42} due to Bose *et al.* [40] is not optimal.

bends. To construct this drawing, place the n vertices along a D-dimensional diagonal, and place the remaining vertices on the sides of the D-dimensional box surrounding the interior vertices. This construction is a generalisation of the case $D = 2$ due to Bose *et al.* [37].

We now provide a simple sufficient condition for the existence of a straight-line 3-D orthogonal line-drawing.

Theorem 3.1. *Every vertex 3-colourable graph has a straight-line 3-D orthogonal line-drawing.*

Proof. We will construct a straight-line 3-D orthogonal line-drawing of the complete tripartite graph $K_{n,n,n}$. Consider the vertices of $K_{n,n,n}$ to be coloured with colours $\{X, Y, Z\}$ with corresponding colour classes $\{u_1, u_2, \ldots, u_n\}$, $\{v_1, v_2, \ldots, v_n\}$ and $\{w_1, w_2, \ldots, w_n\}$. As illustrated in Figure 3.4, a vertex u_i, v_j or w_k, $1 \leq i, j, k, \leq n$ is represented by the following line parallel to the X-, Y or Z-axis, respectively.

- u_i: $(2, 2i+1, 2i) \rightarrow (2n+1, 2i+1, 2i)$
- v_j: $(2j, 2, 2j+1) \rightarrow (2j, 2n+1, 2j+1)$
- w_k: $(2k+1, 2k, 2) \rightarrow (2k+1, 2k, 2n+1)$

A vertex u_i has odd/even Y/Z-coordinates, a vertex v_j has even/odd X/Z-coordinates, and a vertex w_k has odd/even X/Y-coordinates, so no two vertices intersect.

The edge routes for the edges u_iv_j, u_iw_k and v_jw_k, $1 \leq i, j, k, \leq n$, are respectively parallel to the Z-, Y- and X-axes as follows.

- u_i: $(2j, 2i+1, 2i) \rightarrow (2j, 2i+1, 2j+1)$:v_j
- u_i: $(2k+1, 2i, 2i) \rightarrow (2k+1, 2k, 2i)$:w_k
- v_j: $(2j, 2k, 2j+1) \rightarrow (2k+1, 2k, 2j+1)$:w_k

An edge route u_iv_j has even/odd X/Y-coordinates, an edge route u_iw_k has odd/even X/Z-coordinates, and an edge route v_jw_k has even/odd Y/Z-coordinates, so no two edge routes intersect.

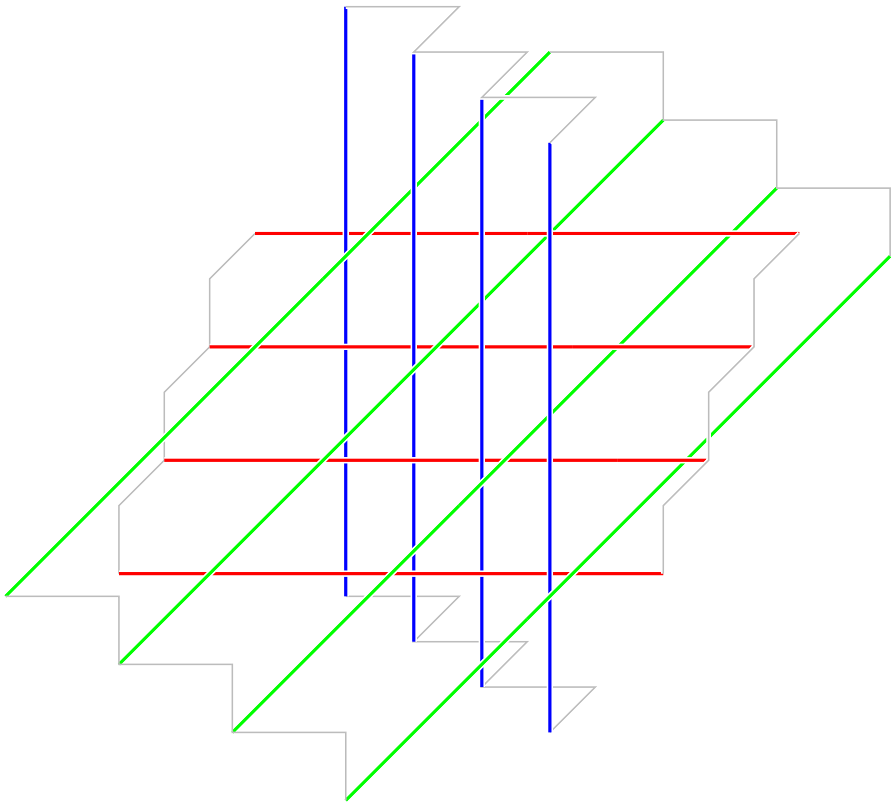


Figure 3.4: Vertex layout for a straight-line 3-D orthogonal line-drawing of a vertex 3-colourable graph.

Suppose an edge route $u_i v_j$ intersects some vertex x. Then x has a coordinate $(2j, 2i+1, Z_x)$, which implies that $x = u_i$ or $x = v_j$, and similarly for edge routes $u_i w_k$ and $v_j w_k$. Hence each edge route only intersects its end-vertices. $\square$

This result suggests the following open problem.

Open Problem 3.1. What is the maximum $k \in \mathbb{Z}^+$ such that every k-colourable graph has a straight-line 3-D orthogonal box-drawing? By Theorem 3.1 and since K_{184} does not have such a drawing we know $3 \leq k < 184$.

3.4.2 Coplanar Vertex Layout Model

A 3-D orthogonal graph drawing is in the *coplanar vertex layout model*, called a *coplanar* 3-D orthogonal graph drawing, if there exists a single grid-plane intersecting every vertex. Of course, such drawings are inherently orientation-dependent.

Coplanar Grid Vertex Layout

One strategy for producing 3-D orthogonal graph drawings in the coplanar vertex layout model, is to position the vertices in a plane grid. This model was first employed by Hagihara *et al.* [113] for producing degree-restricted 3-D orthogonal cube-drawings, although it is understood that all subsequent research in 3-D orthogonal graph drawing, including that presented in this thesis, was completed without knowledge of this paper.

The COMPACT algorithm of Eades *et al.* [86, 87] introduced this model for 3-D orthogonal point-drawing, and produced drawings with optimal volume. Vertices are positioned in the $(Z = 0)$-plane in a $O(\sqrt{n}) \times O(\sqrt{n})$ grid, and edges are routed either within, above or below the $(Z = 0)$-plane. A sequence of refined algorithms in [87] explore the tradeoff between bounding box volume and the maximum number of bends per edge route.

In Chapter 9 we present two algorithms for producing coplanar 3-D orthogonal drawings of arbitrary degree graphs. The first represents vertices by Z-lines in an $O(\sqrt{n}) \times O(\sqrt{n})$ grid, and produces drawings with optimal volume for regular graphs. The second algorithm positions vertices in the $(Z = 0)$-plane in a $O(\sqrt{m}) \times O(\sqrt{m})$ grid, and produces degree-restricted cube-drawings with optimal volume.

Non-Collinear Coplanar Vertex Layout

A second approach to producing coplanar 3-D orthogonal drawings is to position the vertices such that no two vertices lie in the same grid-line. A commonly used strategy for producing such drawings is to position the vertices along a 2-D diagonal.

Biedl *et al.* [32, 33] construct coplanar 3-D orthogonal line-drawings of K_n (and hence for any simple graph), using a 2-D diagonal layout with $O\big(n^3\big)$ volume and one bend per edge route[3]. Biedl [27] calls this the LIFTING-EDGES algorithm. This construction represents the vertices as Z-lines of length n positioned in a 2-D diagonal layout, and routes each edge with one bend in some Z-plane. In Chapter 9 we present an algorithm for producing 1-bend 3-D orthogonal drawings using a similar strategy

[3]Biedl *et al.* [32, 33] also describe 3-D orthogonal drawings of K_n with $O\big(n^3\big)$ volume and two bends per edge route. Since all the vertices in this construction are intersected by a single grid-line, we say this drawing is in the *collinear vertex layout model*.

based on book embeddings.

Biedl [27] introduced an algorithm called LIFTING-HALF-EDGES, which improves on the LIFTING-EDGES algorithm, for producing degree-restricted line-drawings with two bends per edge route. This algorithm starts with a 1-bend 2-D general position point-drawing possibly with overlapping edges (see Section 3.2.3), and extends the vertices to form Z-lines. X-segments are routed above the $(Z = 0)$-plane, Y-segments are routed below the $(Z = 0)$-plane, and Z-segments are added to the edges in such a way to avoid edge route crossings. A modified algorithm produces degree-restricted cube-drawings.

Closson *et al.* [58] present an algorithm for producing coplanar 3-D orthogonal point-drawings with a 2-D diagonal vertex layout, which supports the on-line insertion and deletion of vertices and edges. In Chapter 11 we present an algorithm for multi-dimensional orthogonal point-drawing with a bounded number of bends per edge which also positions the vertices in a 2-D diagonal.

3.4.3 Non-Collinear Model

A 3-D orthogonal graph drawing is in the *non-collinear vertex layout model*, called a *non-collinear* 3-D orthogonal drawing, if no two vertices lie in the same grid-line. The spiral layout algorithm of Closson *et al.* [58] for 3-D orthogonal point-drawing was the first for producing drawings in this model. This algorithm starts with the vertices in a $O(\sqrt{n}) \times O(\sqrt{n})$ grid, and then assigns each vertex a unique height in a spiral manner. The bounding box has volume $O(\sqrt{n}) \times O(\sqrt{n}) \times O(n)$, so the drawings are somewhat orientation-dependent.

In Chapter 10 we present algorithms for generating orientation-independent non-collinear orthogonal box- and point-drawings. Our vertex layout algorithm positions the vertices such that each grid-plane intersects at most $\lceil \sqrt{n} \rceil$ vertices. The point-drawings produced have optimal volume, and for regular graphs, the box-drawings produced also have optimal volume. These are the only known algorithms for producing orientation-independent drawings with optimal volume. For point-drawings with optimal volume, we observe a tradeoff between orientation-independence and the maximum number of bends per edge.

3.4.4 General Position Model

A D-dimensional orthogonal graph drawing ($D \geq 2$) is in the *general position model*, called a *general position* orthogonal drawing, if no two vertices are intersected by a single $(D-1)$-dimensional grid-hyperplane[4]. In a general position 2-D orthogonal drawing, no two vertices are intersected by a single grid-line (see Section 3.2.3), and in a general position 3-D orthogonal drawing, no two vertices are intersected by a single grid-plane[5]. A simple general position vertex layout is constructed by positioning the vertices along the main diagonal of a hypercube, called a *diagonal general position vertex layout*.

General position drawings typically have few bends per edge route (but relatively many bends in total) and are degree-restricted. Many algorithms for general position orthogonal graph drawing produce orientation-independent drawings. The disadvantage of this model is that the drawings necessarily have large volume compared to the other models.

Chapters 5, 6 and 7 describe algorithms for producing general position 3-D point-drawings, general position 2-D box-drawings and general position D-dimensional ($D \geq 3$) box-drawings, respectively. Our algorithms for producing general position orthogonal drawings have the following three major steps, which loosely correspond to those in the *three-phase method* [31].

Vertex Layout: Determine the relative positions of the vertices.

Arc Routing: Determine the 'shape' of each edge route.

Port Assignment: Construct vertex boxes, assign ports for each edge route, and remove edge crossings.

We classify algorithms for generating general position orthogonal graph drawings as being *layout-* or *routing-based*. In a layout-based algorithm, the vertex layout stage

[4]In computational geometry a set of points in $\mathbb{R}^D$ are in *general position* if no $D+1$ points are in a common $(D-1)$-dimensional hyperplane. Strictly speaking we should therefore say a general position orthogonal drawing is in *general grid position*.

[5]This is called the *Unique Coordinates Model* in [221].

is completed initially followed by the arc routing step. In a routing-based algorithm, the vertex layout is determined with respect to a pre-determined arc-routing. The port assignment stage is always completed last.

Point-Drawings

A 3-D diagonal vertex layout is used by the 3-Bends algorithm of Eades *et al.* [86, 87] for orthogonal point-drawing. We present a layout-based algorithm for 3-D orthogonal point-drawing in Section 5.2.1, which given a fixed diagonal layout, minimises the total number of bends. A modification of the 3-bends algorithm of Eades *et al.* [86, 87] described in Section 5.5.3, produces 3-bend point-drawings with the best known volume upper bound.

A routing-based algorithm for 3-D orthogonal point-drawing is presented in Section 5.3. The Diagonal Layout and Movement (Dlm in Table 3.2) algorithm presented in Section 5.4 combines the layout- and routing-based approaches, and establishes the best known upper bound for the total number of bends in 3-D orthogonal point-drawings.

Box-Drawings

Algorithms for producing general position 3-D orthogonal box-drawings with two bends per edge route have been developed by Papakostas and Tollis [166, 168] and Biedl [27]. The incremental algorithm in [166, 168] inserts each new vertex as a cube, and as new neighbours are inserted a vertex may grow in different directions, producing drawings which one would expect in practice to be orientation-independent. No bound on the aspect ratio of a vertex is established. We refer to this algorithm as Incremental.

Our layout-based algorithm for multi-dimensional orthogonal box-drawing, presented in Section 7.2, in the case of three dimensions, establishes improved bounds on the degree-restriction of vertices compared to the algorithms in [27, 166, 168]. A routing-based algorithm for general position 3-D orthogonal box-drawing is presented in Section 7.3.

3.4.5 Ad-hoc Methods for 3-D Point-Drawing

Other approaches for 3-D orthogonal point-drawing include that of Papakostas and Tollis [166, 168]. Their algorithm, which allows for the on-line insertion of vertices in constant time, produces 3-D orthogonal point-drawings with at most three bends per edge route. The *split and push* approach to 3-D orthogonal point-drawing, developed by Di Battista *et al.* [74], starts with a degenerate drawing with all vertices on one point and repeatedly inserts planes splitting the drawing apart until all crossings are removed. Experimental tests in [74, 168, 221] show this method works well only on relatively small graphs, and no bounds on the number of bends or volume are presented.

3.5 Bounds for 3-D Orthogonal Graph Drawing

We now summarise the known bounds for the number of bends and the volume of 3-D orthogonal drawings, initially for point-drawings and then for box-drawings.

3.5.1 Point-Drawings

Table 3.2 shows the tradeoff between the bounding box volume and the maximum number of bends per edge apparent in algorithms for 3-D orthogonal point-drawing of graphs of maximum degree $\Delta \leq 6$.

Bounds on the volume

An early result in 3-D orthogonal point-drawing due to Kolmogorov and Barzdin [132][6] established a lower bound of $\Omega(n^{3/2})$ for the bounding box volume. Rosenberg [186] independently proved the same result.

The Compact algorithm of Eades *et al.* [86, 87] determines orthogonal point-drawings in the coplanar vertex layout model with $O\big(n^{3/2}\big)$ bounding box volume and at most seven bends per edge route. As discussed above, this volume bound is asymptotically best possible. The same bound is achieved by the orientation-independent Non-Collinear algorithm presented in Chapter 10, at the expense of needing eight

[6] This paper has been repeatedly cited incorrectly in the literature, with the word 'set' replacing 'net' in the title.

Table 3.2: Upper Bounds for 3-D Orthogonal Point-Drawing

Algorithm	Max. (Avg.) Bends	Volume	Orientation Independent	Reference
Non-Collinear	8	$\Theta(n^{3/2})$	yes	Theorem 10.2
Compact	7	$\Theta(n^{3/2})$	no	[86, 87]
Compact1	6	$O(n^2)$	no	[87]
Dynamic	5	$O(n^2)$	no	[58]
Compact2	5	$O(n^{5/2})$	no	[87]
Compact3	4	$O(n^3)$	no	[87]
Dlm	4 (7/3)	$2.37n^3$	yes	Theorem 5.4
3-Bends	3	$8n^3$	yes	[86, 87]
Incremental	3	$4.63n^3$	yes	[166, 168]
Modified 3-Bends	3	$n^3 + o(n^3)$	yes	Theorem 5.6
Dlm ($\Delta \leq 5$)	2	n^3	yes	Theorem 5.4
Compact ($\Delta \leq 4$)	3	$O(n^2)$	no	[86]

bends for some edge routes. Improving the bound on the maximum number of bends per edge route in an $O(n^{3/2})$ volume 3-D orthogonal point-drawing is an interesting open problem.

Open Problem 3.2. Does every maximum degree six graph have a 6-bend 3-D orthogonal point-drawing with $O(n^{3/2})$ bounding box volume?

In a series of refinements of the Compact algorithm, referred to as Compact1, Compact2 and Compact3, the tradeoff between the bounding box volume and the maximum number of bends per edge route is explored. For $O(n^2)$ volume 3-D point-drawings, the Dynamic algorithm of Closson *et al.* [58] improves the upper bound for the maximum number of bends per edge route from six [87] to five.

Bounds on the maximum number of bends per edge

The 3-Bends algorithm of Eades *et al.* [86, 87] and the Incremental algorithm of Papakostas and Tollis [166, 168] established an upper bound of three for the maximum number of bends per edge route. Both algorithms take $O(n)$ time[7]. Note that the authors of the 3-Bends algorithm were not interested in improving the constant in the $27n^3$ bounding box volume bound — by deleting each grid plane not containing a vertex or a bend, it can easily be shown that the volume is at most $8n^3$. A modification of the 3-Bends algorithm presented in Section 5.5.3 improves this bound to $n^3 + o\left(n^3\right)$. This is the best known upper bound for the volume of 3-bend orthogonal point-drawings.

There are few non-trivial lower bounds for the number of bends in 3-D orthogonal point-drawings. Obviously any orthogonal point-drawing of K_3 has at least one bend. Less obvious is the result, proved in Theorem 11.1, that in any 3-D orthogonal point-drawing of K_5 there is an edge route with at least two bends. In Appendix A we give a formal proof of the well-known result that a 3-D orthogonal point-drawing of the multigraph consisting of two vertices and six edges requires an edge route with at least three bends.

The difference between the lower bound of two and the upper bound of three for the maximum number of bends per edge route in 3-D orthogonal point-drawings of maximum degree six graphs motivates the following *2-Bends Problem*.

Open Problem 3.3. [86, 87] Does every maximum degree six graph admit a 2-bend 3-D orthogonal point-drawing?

The Diagonal Layout and Movement algorithm (Dlm in Table 3.2) presented in Section 5.4 solves the 2-Bends Problem in the affirmative for graphs of maximum degree five. This result establishes the only known class of graphs for which 2-bend 3-D orthogonal point-drawings exist.

A natural candidate for a simple graph requiring an edge route with at least three bends in every 3-D orthogonal point-drawing is K_7, as conjectured by Eades *et al.* [86]. A counterexample to this conjecture, namely a 3-D orthogonal point-drawing of K_7

[7] In Eades *et al.* [86] an $O(n^{3/2})$ time bound is stated. In Eades *et al.* [87] this is reduced to $O(n)$ using the algorithm of Schrijver [194] in the calculation of the cycle cover decomposition (see Section 2.5).

with at most two bends per edge route, was first exhibited by Wood [219]. A more symmetric 3-D orthogonal point-drawing[8] of K_7 with at most two bends per edge route is shown in Figures 3.5 and 3.6 (see also Appendix B). This drawing has the interesting feature of rotational symmetry about the line $X = Y = Z$.

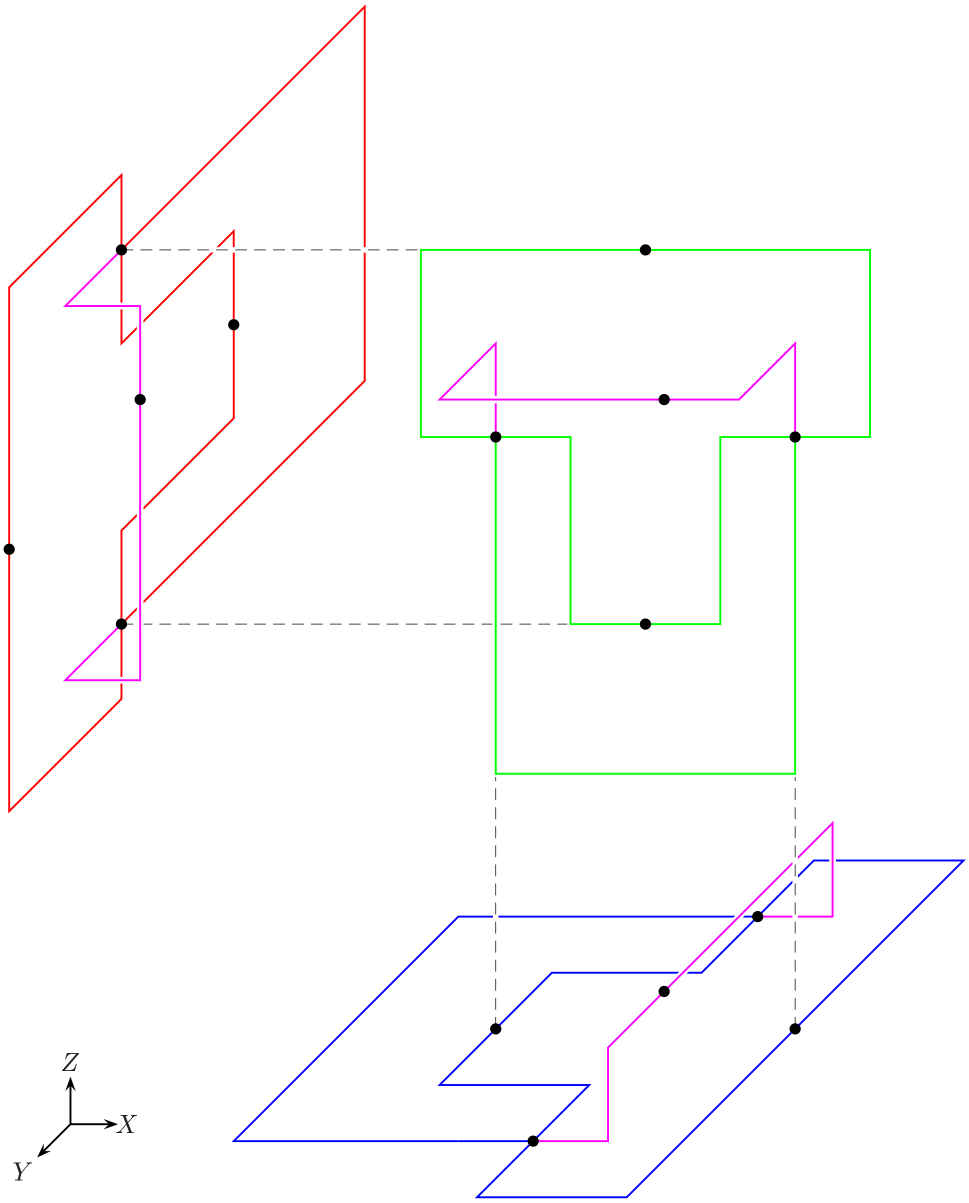


Figure 3.5: Components of a 2-bend 3-D orthogonal point-drawing of K_7.

[8] A physical model of this drawing is on display at the School of Computer Science and Software Engineering, Monash University, Clayton.

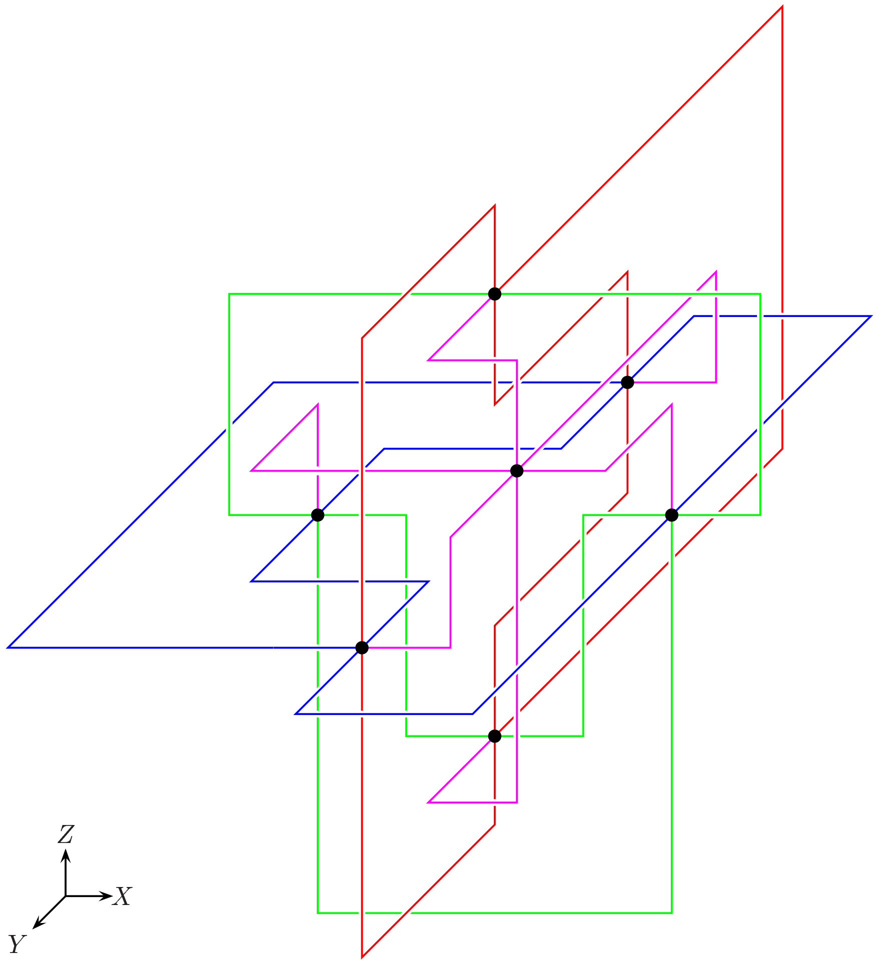


Figure 3.6: A 2-bend 3-D orthogonal point-drawing of K_7.

One may consider the other 6-regular complete multi-partite graphs $K_{6,6}$, $K_{3,3,3}$ and $K_{2,2,2,2}$ to be potential examples of simple graphs requiring an edge route with at least three bends. In Appendix B we present 2-bend 3-D orthogonal point-drawings of these graphs.

Bounds on the total number of bends

In certain applications it may be more important to minimise the total number of bends in 3-D orthogonal point-drawings rather than to minimise the maximum number of bends on any edge route. The Diagonal Layout and Movement algorithm presented in Section 5.4, which solves the 2-Bends Problem for graphs of maximum degree five, uses a total of at most $7m/3$ bends for drawings of m-edge simple graphs with maximum degree six. A related algorithm presented in Section 5.2.1 minimises the total number of bends in a 3-D orthogonal point-drawing for a fixed diagonal layout. Improving the upper bound for the total number of bends in a 3-D orthogonal point-drawing is an interesting open problem.

Open Problem 3.4. Does every maximum degree six graph with m edges have a 3-D orthogonal point-drawing with fewer than $7m/3$ bends?

In Appendix A we establish the first non-trivial lower bounds for the total number of bends in 3-D orthogonal point-drawings. In particular, we prove that a 3-D orthogonal point-drawing of K_5 has at least seven bends. (A drawing of K_5 with seven bends is shown in Figure 2.3(b) on page 28.) We also show that a 3-D orthogonal point-drawing of the multigraph consisting of two vertices and six edges has at least twelve bends. (Such a drawing is shown in Figure A.7 on page 228.)

Open Problem 3.5. Are there better lower bounds than $7m/10$ (for simple graphs) and $2m$ (for multigraphs) on the total number of bends in a 3-D orthogonal point-drawing of an m-edge graph with maximum degree six.

In Figure 3.7 we show a 3-D orthogonal point-drawing of K_7 with a total of 24 bends (compared with the total of 42 bends for the drawing shown in Figures 3.5 and 3.6). Most edge routes are straight-lines or have one bend, and three edge routes have four bends. We conjecture that there is no 3-D orthogonal point-drawing of K_7 with fewer than 24 bends.

3.5.2 Box-Drawings

Lower Bounds

The first lower bounds for 3-D orthogonal box-drawings were due to Hagihara *et al.* [113]. They show that the volume of a degree-restricted 3-D orthogonal cube-drawing of a simple graph is

$$\Omega\left(\max\left\{\Delta^2 n, (\Delta n/\log n)^{3/2}\right\}\right) \quad .$$

For an arbitrary graph G, let $\mathrm{vol}(G, r, \alpha)$ denote the minimum bounding box volume of the 3-D orthogonal drawings of G which are strictly α-degree-restricted and every

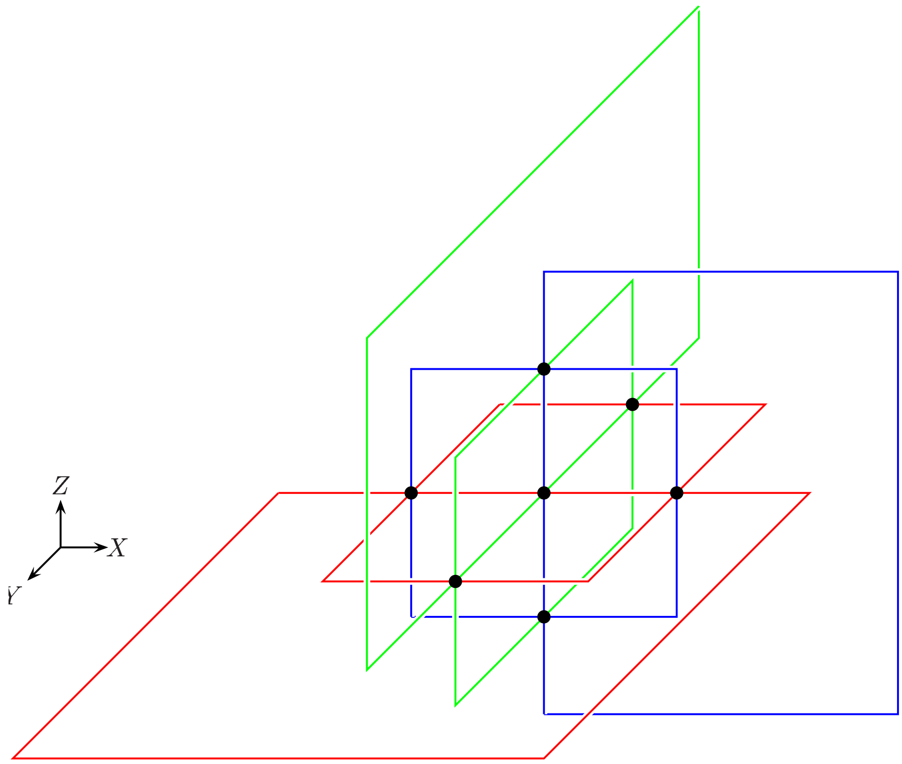


Figure 3.7: A 4-bend 3-D orthogonal point-drawing of K_7 with 24 bends.

vertex has aspect ratio at most r. Let $\text{vol}(n, m, r, \alpha)$ be the maximum of $\text{vol}(G, r, \alpha)$ where G is a graph with n vertices and m edges. Thus, $\text{vol}(n, m, r, \alpha)$ describes a volume bound within which all graphs with n vertices and m edges can be drawn such that each vertex v has aspect ratio at most r and surface at most $\alpha \cdot \deg(v)$. Biedl, Thiele, and Wood [34] establish the following results.

Theorem 3.2.

- $\text{vol}\,(n, m, \infty, \infty) = \Omega\,(m\sqrt{n})$
- $\text{vol}\,(n, m, r, \infty) = \Omega\left(m^{3/2}/\sqrt{r}\right)$
- $\text{vol}\,(n, m, \infty, \alpha) = \Omega\left(m^{3/2}/\alpha\right)$

Hence the volume of arbitrary 3-D orthogonal box-drawings is $\Omega(m\sqrt{n})$, and for degree-restricted drawings or drawings with each vertex having bounded aspect ratio, the volume is $\Omega(m^{3/2})$. This result includes the lower bound of $\Omega(n^{5/2})$ for the volume of 3-D orthogonal drawings of K_n due to Biedl *et al.* [32, 33]. In fact, the proof is based on techniques developed in that paper generalised for sparse graphs. Biedl *et al.* [32, 33] also establish the lower bound of $\Omega(n^2)$ for the number of bends in a 3-D orthogonal drawings of K_n. For general position 3-D orthogonal drawings, Biedl [27] establishes a lower bound of $\Omega\left(\max\left\{n^3, m^2\right\}\right)$ for the bounding box volume, and conjectures the lower bound of $\Omega(n^2 m)$.

Upper Bounds

The algorithm presented in Section 9.1, which generalises the Lifting-Edges algorithm of Biedl *et al.* [32, 33] for simple graphs, establishes that every multigraph has a 1-bend 3-D orthogonal box-drawing. As discussed in Section 3.4.1, there exist graphs with no straight-line 3-D orthogonal box-drawing, so these results are optimal for the maximum number of bends per edge route. Since the drawings produced are orientation-dependent and are not degree-restricted, the following open problem is of interest.

Open Problem 3.6. Does every graph have an orientation-independent or degree-restricted 3-D orthogonal box-drawing with at most one bend per edge route?

The algorithm of [34] produces 3-D orthogonal box-drawings with $O(m\sqrt{n})$ volume and at most four bends per edge route. By Theorem 3.2 this bound is optimal. A simplified version of this algorithm, presented in Section 9.2, produces drawings with $O\big(\Delta n^{3/2}\big)$ volume, which for regular graphs is the same as $O(m\sqrt{n})$. Reducing the number of bends in optimal volume box-drawings is an important open problem.

Open Problem 3.7. Does every graph have a 3-D orthogonal box-drawing with $O(m\sqrt{n})$ volume and at most three bends per edge route? (Note that K_n *does* have a 3-bend box-drawing with $O\big(n^{5/2}\big) = O(m\sqrt{n})$ volume [32, 33].)

We now consider upper bounds for the volume of degree-restricted 3-D orthogonal box-drawings. The Incremental algorithm of Papakostas and Tollis [166, 168] first established that every graph has a 2-bend degree-restricted 3-D orthogonal box-drawing. Their upper bound of $O\big(m^3\big)$ for the bounding box volume has subsequently been improved by the Lifting Half-Edges algorithm of Biedl [27] to $O\big(n^2\Delta\big)$.

The algorithm presented in Section 9.3 produces degree-restricted cube-drawings with $O\big((m+n)^{3/2}\big)$ volume. By Theorem 3.2 this upper bound is optimal for degree-restricted drawings or drawings with each vertex having bounded aspect ratio (assuming $m = \Omega(n)$, which is true for most graphs). This algorithm uses at most six bends per edge route. The following problem is therefore of interest.

Open Problem 3.8. Does every graph have a 5-bend degree-restricted 3-D orthogonal box-drawing with $O\big((m+n)^{3/2}\big)$ bounding box volume and bounded aspect ratio vertices?

Table 3.3 summarises the known bounds for 3-D orthogonal box-drawings (of n-vertex m-edge graphs with maximum degree Δ and genus g $(\leq m)$). We consider four groupings of algorithms, depending on which aesthetic criteria (out of orientation-independent, bounded aspect ratio and degree-restricted) are satisfied by the drawings produced. Within each grouping a tradeoff between the bounding box volume and the maximum number of bends per edge route is observed.

Table 3.3: Bounds for 3-D Orthogonal Box-Drawings.

Volume	Bends	Model	Graphs	Time	Reference
orientation-independent / bounded aspect ratio / degree-restricted					
$O\big((nm)^{3/2}\big)$	2	general position	simple	$O(m)$	[27] (Thms. 7.5,7.6)
$O\big((n\Delta)^{3/2}\big)$	6	non-collinear	multigraphs	$O(m)$	Theorem 10.1
orientation-dependent / bounded aspect ratio / degree-restricted					
$O\Big(nm\sqrt{\Delta}\Big)$	2	lifting $\frac{1}{2}$-edges	multigraphs	$O(m)$	[27]
$O(m(m+n))$	5	coplanar	multigraphs	$O(m)$	Theorem 9.5
$O\big((n\Delta)^{3/2}\big)$	10	coplanar	simple	?	[113]
$\Theta((m+n)^{3/2})$	6	coplanar	multigraphs	$O\big(m\sqrt{m+n}\big)$	Theorem 9.4
orientation-dependent / no bounds on aspect ratio / degree-restricted					
$O\big(n^2\Delta\big)$	2	lifting $\frac{1}{2}$-edges	simple	$O(m)$	[27]
$\Theta((m+n)^{3/2})$	6	coplanar	multigraphs	$O\big(m\sqrt{m+n}\big)$	Theorem 9.4
orientation-dependent / no bounds on aspect ratio / not degree-restricted					
$O\big(n^3\big)$	1	lifting edges	simple	$O(m)$	[32, 33]
$O\big(nm\sqrt{g}\big)$	1	diagonal coplanar	multigraphs	-	Theorem 9.1
$O\big(n^{5/2}\big)$	3	lifting edges	simple	$O(m)$	[32, 33]
$O(nm)$	3	coplanar	multigraphs	$O(m)$	Theorem 9.3
$\Theta(m\sqrt{n})$	4	coplanar	multigraphs	$O\big(m^2/\sqrt{n}\big)$	[34] (see Thm. 9.2)

Tables 3.4, 3.5 and 3.6 provide precise bounds on the aesthetic criteria for each the first three groups discussed above.

Table 3.4: Orientation-independent, Degree-restricted 3-D Orthogonal Drawing with Bounded Aspect Ratio.

Bends	Volume	Degree-Restriction	Aspect Ratio	Model	Reference
2	$O\big((nm)^{3/2}\big)$	6	1	general position	[27]
2	$O\big((nm)^{3/2}\big)$	5/3	2	general position	Theorem 7.5
2	$O\big((nm)^{3/2}\big)$	4	1	general position diagonal	Theorem 7.6
6	$O\big((n\Delta)^{3/2}\big)$	8	1	non-collinear [9]	Theorem 10.1

Table 3.5: Degree-restricted 3-D Orthogonal Cube-Drawing Algorithms.

Bends	Volume	Degree Restriction	Aspect Ratio	Model	Reference
2	$O\big(n^2 m\big)$	6	1	lifting $\frac{1}{2}$-edges	[27]
5	$O(m(m+n))$	12	1	coplanar layout	Theorem 9.5
6	$O\big((m+n)^{3/2}\big)$	12	1	coplanar layout	Theorem 9.4

Table 3.6: Degree-restricted 3-D Orthogonal Drawing with Unbounded Aspect Ratio.

Bends	Volume	Degree-Restriction	Aspect Ratio	Model	Reference
2	$O\big(m^3\big)$	6	-	incremental	[166, 168]
2	$O\big(n^2\Delta\big)$	2	$\deg(v)/2$	lifting $\frac{1}{2}$-edges	[27]
2	$O\big(n^2 m\big)$	2	$\deg(v)/2$	general position	[27]
2	$O\big(n^2 m\big)$	2	$\deg(v)/2$	general position diagonal	Theorem 7.7
2	$O\big(\Delta\,(nm)^{3/2}\big)$	2	$\deg(v)/4$	general position	Theorem 7.8

[9] 4-degree-restricted for simple graphs.

Part II

General Position Orthogonal Graph Drawing

Chapter 4

Balanced Vertex Ordering

In this chapter we describe and analyse methods for determining 'balanced' orderings of the vertices of a graph. Here balanced means that the neighbours of each vertex v are evenly distributed to the left and right of v in the ordering. This problem is of theoretic interest in its own right, and forms an important part of the graph drawing algorithms to be presented in Chapters 5, 6 and 7. In particular, we define the cost of a vertex ordering as a measure of its imbalance, and present a linear time heuristic with tight worst case bounds for the cost of the vertex orderings produced. Furthermore we establish useful properties of vertex orderings which locally minimise the cost.

4.1 Introduction

A number of the algorithms for producing general position orthogonal graph drawings involve the manipulation of an ordering of the vertices of a graph. Given a (di)graph G, a total ordering $<$ on $V(G)$ induces a numbering $(v_1, v_2, \ldots, v_n)$ of $V(G)$ and vice versa. We shall refer to both $<$ and $(v_1, v_2, \ldots, v_n)$ as a *vertex ordering* of G.

Consider 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 $\overrightarrow{wv}$ is a *predecessor arc* of w and v is a *predecessor* of w. Now consider a vertex ordering $<$ of a digraph G. For each edge $vw \in E(G)$, if $v < w$ we say $\overrightarrow{vw}$ is a *successor* *arc* of v and w is a *successor* of v, and if $w < v$ we say $\overrightarrow{vw}$ is a *predecessor arc* of v and w is a *predecessor* of v.

For each vertex $v \in V(G)$, the number of successor and predecessor arcs of v are denoted $s_<(v)$ and $p_<(v)$, respectively. Where the vertex ordering $<$ is clear from the context we use $s(v)$ and $p(v)$ instead of $s_<(v)$ and $p_<(v)$, respectively. Note that, for digraphs, we only count the outgoing edges at a vertex v in $p(v)$ and $s(v)$.

We say a vertex v in a given vertex ordering is *positive* if $s(v) > p(v)$, *negative* if $p(v) > s(v)$ and *balanced* if $s(v) = p(v)$. For positive and balanced vertices v and for $k > 0$ (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. Two adjacent vertices v, w with $v < w$ are *opposite* if v is positive and w is negative.

As illustrated in Figure 4.1, we shall say a vertex v is each of the following types.

- $p(v)$-$s(v)$ vertex
- $(\min\{p(v), s(v)\}, \max\{p(v), s(v)\})$-vertex
- $\max\{p(v), s(v)\}$-vertex.

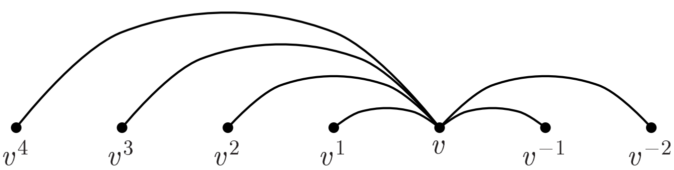


Figure 4.1: In a vertex ordering, v is a 4-2 vertex, a $(2, 4)$-vertex, and a 4-vertex.

In a vertex ordering of a (di)graph G, we measure the imbalance of a vertex by defining the *cost* of v to be $c(v) = |s(v) - p(v)|$. Note that a vertex has even cost if and only if it has even (out)degree, and the cost of an odd (out)degree vertex is at least one. We firstly note that,

$$2 \cdot \min\{s(v), p(v)\} + c(v) = \text{(out)deg}(v) = 2 \cdot \max\{s(v), p(v)\} - c(v) \qquad (4.1)$$

The *total cost* of a vertex ordering is the sum of the costs of the vertices. In a vertex ordering of an undirected graph G, the total cost is equal to the total cost of the same vertex ordering of the digraph $\overleftrightarrow{G}$. Hence we model a vertex ordering of an undirected graph G by a vertex ordering of the digraph $\overleftrightarrow{G}$. We are interested in the following problem.

Problem 4.1. BALANCED VERTEX ORDERING

Instance : A (di)graph G, integer $K \geq 0$.

Question : Does G have a vertex ordering with total cost $\sum_{v \in V(G)} c(v) \leq K$?

We conjecture that the BALANCED VERTEX ORDERING problem is NP-complete. To establish bounds for this problem we employ a heuristic approach in Section 4.3, and a local minimum approach in Section 4.4. Obviously any vertex ordering of the complete graph has the same total cost, thus providing an important lower bound for the balanced ordering problem.

Lemma 4.1. *In any vertex ordering of the complete graph K_n, the total cost*

$$\sum_v c(v) = \left\lfloor \frac{n^2}{2} \right\rfloor = m + \left\lfloor \frac{n}{2} \right\rfloor .$$

Proof. In a vertex ordering $(v_1, v_2, \ldots, v_n)$ the total cost is

$$\begin{aligned}
\sum_{1 \leq i \leq n} |s(v_i) - p(v_i)| &= 2 \sum_{1 \leq i \leq \lfloor n/2 \rfloor} (n - 2i + 1) \\
&= 2 \left(\lfloor n/2 \rfloor (n+1) - 2 \sum_{1 \leq i \leq \lfloor n/2 \rfloor} i \right) \\
&= 2\big(\lfloor n/2 \rfloor (n+1) - \lfloor n/2 \rfloor (\lfloor n/2 \rfloor + 1)\big) \\
&= \lfloor n^2/2 \rfloor \\
&= m + \lfloor n/2 \rfloor .
\end{aligned}$$

□

4.2 st-Orderings

A vertex ordering $(v_1, v_2, \ldots, v_n)$ of a (di)graph G is an st-ordering if $v_1 = s$, $v_n = t$, and for every other vertex v_i, $1 < i < n$, with (out)deg $(v_i) \geq 2$, we have $p(v_i) \geq 1$ and $s(v_i) \geq 1$. Lempel *et al.* [142] show that for any biconnected undirected graph G and for any $s, t \in V(G)$, there exists an st-ordering of G. Recently Cheriyan and Reif [54] extended this result to digraphs.

Even and Tarjan [93] develop a linear time algorithm to compute an st-ordering of an undirected biconnected graph. It is an open problem to develop a linear time algorithm for finding an st-ordering of a biconnected digraph. To determine a vertex ordering of a connected graph based on st-orderings of its biconnected components (blocks), number the blocks $B_1, B_2, \ldots, B_k$ according to a depth-first-search of the block-tree, and concatenate $s_i t_i$-orderings of each B_i, where s_i (respectively, t_i) is chosen wherever possible to be a cut-vertex with some block B_j, $j < i$ ($j > i$). We obtain the following easy result.

Lemma 4.2. *Every graph G has a vertex ordering, which can be computed in $O(n+m)$ time, with at most $c+k$ vertices v having $p(v) = 0$ or $s(v) = 0$, where c is the number of connected components of G, and k is the number of end-blocks in the block decomposition of G. (An end-block corresponds to a leaf of the block-forest. Note that an isolated edge contributes one connected component and one end-block.)* □

4.3 Median Placement Ordering

We now describe a heuristic for the balanced vertex ordering problem which provides a tight upper bound for the total cost of the vertex orderings produced, and forms a critical part of many of the graph drawing algorithms presented in this thesis. The algorithm inserts each vertex, in turn, mid-way between its already inserted neighbours. At any stage of the algorithm we refer to the ordering under construction as the *current ordering*. Similar methods were introduced by Biedl and Kaufmann [30] and Biedl *et al.* [31].

Algorithm 4.1. MEDIAN PLACEMENT ORDERING

Input: • (di)graph G.

• vertex ordering $(u_1, u_2, \ldots, u_n)$ of G (called the *insertion ordering*).

Output: vertex ordering of G.

---

for $i = 1, 2, \ldots, n$ **do**

Suppose the predecessors of u_i in the insertion ordering are ordered $w_1, w_2, \ldots, w_k$ in the current ordering.

if $k = 0$ **then** Insert u_i arbitrarily into the current ordering.

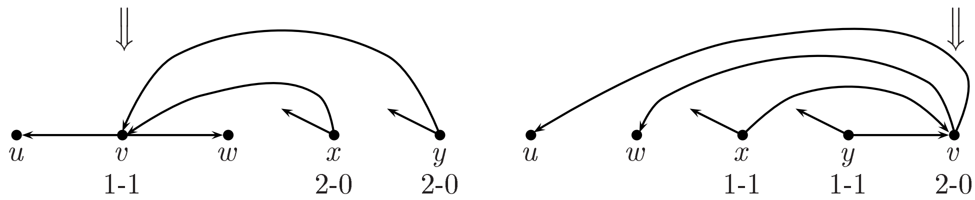
else if k is even **then** Insert u_i arbitrarily between $w_{k/2}$ and $w_{k/2+1}$.

else (k is odd) Insert u_i immediately before or after $w_{(k+1)/2}$.

end-for

Output the current ordering.

It is easily seen that for undirected graphs the MEDIAN PLACEMENT ORDERING algorithm, at each iteration, inserts the vertex u_i to minimise the total cost of the current ordering. For digraphs this is not the case, as the example in Figure 4.2 illustrates.



(a) median placement insertion (b) minimum cost insertion

Figure 4.2: Inserting vertex v into a vertex ordering of a digraph.

Lemma 4.3. *The algorithm* MEDIAN PLACEMENT ORDERING *determines a vertex ordering of a (di)graph G, in $O(m+n)$ time, with total cost*

$$\sum_{v \in V(G)} c(v) \;\leq\; k + \sum_{1 \leq i \leq n} s(u_i), \text{ and}$$

$$\sum_{v \in V(G)} \max\{s(v), p(v)\} \;\leq\; m + \frac{1}{2}\left(k + \sum_{1 \leq i \leq n} s(u_i)\right) .$$

where, in the insertion ordering, $s(u_i)$ is the number of successors of u_i and k is the number of vertices $u_i \in V(G)$ with odd $p(u_i)$.

Proof. When a vertex u_i is inserted into the current ordering it has cost $c(u_i) = 0$ if $p(u_i)$ is even and $c(u_i) = 1$ if $p(u_i)$ is odd. So, even if all the successors of u_i (in the insertion ordering) are inserted on the one side of u_i, in the final ordering, the cost

$c(u_i) \leq s(u_i)$ if $p(u_i)$ is even, and $c(u_i) \leq s(u_i) + 1$ if $p(u_i)$ is odd. So the total cost is at most $k + \sum_i s(u_i)$. By (4.1) we have

$$\sum_v \max\{s(v), p(v)\} \leq \sum_v \frac{\deg(v) + c(v)}{2} = m + \frac{1}{2}\left(k + \sum_i s(u_i)\right)$$

Using the median-finding algorithm of Blum *et al.* [36], and the algorithm of Dietz and Sleator [77] to maintain the vertex ordering and orderings of the adjacency lists of G, the algorithm can be implemented in $O(m+n)$ time. □

For an important class of graphs, if the insertion ordering is chosen carefully, the MEDIAN PLACEMENT ORDERING algorithm is optimal.

Theorem 4.1. *A minimum-cost vertex ordering of an acyclic (di)graph can be determined in $O(m+n)$ time.*

Proof. Using a reverse topological ordering as the insertion ordering in the MEDIAN PLACEMENT ORDERING algorithm, each vertex v has $s(v) = 0$ in the insertion ordering, so no neighbours of v are inserted into the current ordering after v. Hence $c(v) = 1$ if $p(v)$ is odd, and $c(v) = 0$ if $p(v)$ is even. Since $p(v) = \text{(out)deg}\,(v)$ the ordering has minimum cost. A topological ordering can be determined in $O(m+n)$ time [64], as can the algorithm MEDIAN PLACEMENT ORDERING (see Lemma 4.3). □

For undirected graphs, $\sum_i s(u_i) = m$ in any ordering, and since $k \leq n$, we obtain the following immediate corollary.

Corollary 4.1. *The* MEDIAN PLACEMENT ORDERING *algorithm, with any insertion ordering, determines a vertex ordering of an undirected graph G with total cost*

$$\sum_{v \in V(G)} c(v) \leq m + n \text{ , and } \sum_{v \in V(G)} \max\{s(v), p(v)\} \leq \frac{3m+n}{2} \text{ .}$$

If we choose a particular insertion ordering we can obtain improved upper bounds on the total cost of the vertex orderings produced by the MEDIAN PLACEMENT ORDERING algorithm. As indicated by Lemma 4.3, there are two approaches for determining a 'good' insertion ordering.

1. Determine an insertion ordering with a small number of vertices with an odd number of predecessors. We present an algorithm for doing so in Section 4.3.1.

2. Determine an insertion ordering with small $\sum_i s(u_i)$. For undirected graphs, $\sum_i s(u_i) = m$ in any ordering, so this approach is only applicable for digraphs. We describe methods for determining an insertion ordering with small $\sum_i s(u_i)$ in Section 4.3.2.

4.3.1 Vertices with an Odd Number of Predecessors

We now describe an algorithm for determining a vertex ordering with few vertices having an odd number of predecessors. The ordering is constructed from right to left; i.e., from v_n to v_1.

Algorithm 4.2. INSERTION ORDERING

Input: (di)graph G.

Output: vertex ordering of G.

Set $i \leftarrow |V(G)|$.

while $E(G) \neq \emptyset$ **do**

  Choose an edge $vw \in E(G)$.

  if (out)deg (v) is even **then** Set $u_i \leftarrow v$; $u_{i-1} \leftarrow w$; **else** Set $u_i \leftarrow w$; $u_{i-1} \leftarrow v$.

  Remove v and w (and their incident edges) from G.

  Set $i \leftarrow i - 2$.

end-while

while $V(G) \neq \emptyset$ **do**

  Choose $v \in V(G)$.

  Set $u_i \leftarrow v$.

  Remove v from G.

  Set $i \leftarrow i - 1$.

end-while

Output $(u_1, u_2, \ldots, u_n)$.

Lemma 4.4. *The algorithm* INSERTION ORDERING *determines a vertex ordering* $(u_1, u_2, \ldots, u_n)$ *of* G *with at most* $\lfloor n/2 \rfloor$ *vertices* u_i *having odd* $p(u_i)$.

Proof. Consider an iteration of the first while-loop in the algorithm. If (out)deg (v) is even then $p(v) = $ (out)deg (v), otherwise if (out)deg (v) is odd then $p(v) = $ (out)deg $(v) - 1$. In either case, the vertex v will have an even number of predecessors in $(u_1, u_2, \ldots, u_n)$. So at least half of the vertices added to the ordering in the first stage of the algorithm have an even number of predecessors. During the second while-loop every vertex v has $p(v) = 0$ and thus has an even number of predecessors in $(u_1, u_2, \ldots, u_n)$. The result follows. □

Combining Lemma 4.3 and Lemma 4.4 we obtain the following result.

Theorem 4.2. *Every undirected graph G has a vertex ordering, which can be computed in $O(n+m)$ time, with total cost*

$$\sum_{v \in V(G)} c(v) \;\leq\; m + \left\lfloor \frac{n}{2} \right\rfloor , \text{ and } \sum_{v \in V(G)} \max\{s(v), p(v)\} \;\leq\; \frac{3m}{2} + \frac{n}{4} \;. \qquad \square$$

By Lemma 4.1, the vertex ordering of the undirected complete graph K_n has total cost $m + \lfloor n/2 \rfloor$, so for K_n we have a tight bound on the total cost.

4.3.2 Feedback Arc Set Problem

We now describe the second method for improving the bound on the total cost of vertex orderings produced by the MEDIAN PLACEMENT ORDERING algorithm. This method is only applicable for digraphs. We wish to determine an insertion ordering $(u_1, u_2, \ldots, u_n)$ with small $\sum_i s(u_i)$.

A *feedback arc set* of a digraph G is a set of arcs of G whose removal makes the graph acyclic. A vertex ordering $<$ of a digraph determines a feedback arc set consisting of the edges $\{vw \in E(G) : v < w\}$. Conversely, given a feedback arc set $F \subset E(G)$, a topological ordering $<$ of $G[F]$ has $|\{vw \in E(G) : v < w\}| = |F|$. So determining an insertion ordering with minimum $\sum_i s(u_i)$ is equivalent to the problem of determining a feedback arc set of minimum size. This problem, called the FEEDBACK ARC SET problem, is NP-hard [125]. For any vertex ordering of a digraph,

$$\min\left\{\sum_v s(v), \sum_v p(v)\right\} \leq m/2 \;.$$

So trivially every digraph has a vertex ordering with $\sum_i s(u_i) \leq m/2$.

Berger and Shor [19] establish an asymptotically tight bound for the FEEDBACK ARC SET problem[1]. They show that, for digraphs of maximum degree Δ and without 2-cycles, the minimum of $\sum_i s(u_i)$ (taken over all vertex orderings) is $m/2 - \Theta(m/\sqrt{\Delta})$, and a vertex ordering with $\sum_i s(u_i) = m/2 - \Theta(m/\sqrt{\Delta})$ can be determined in $O(mn)$ time. Using such an ordering as the insertion ordering in algorithm MEDIAN PLACEMENT ORDERING, by Lemma 4.3 with $k \leq n$, we obtain the following result.

Theorem 4.3. *Every digraph without 2-cycles has a vertex ordering, which can be computed in $O(mn)$ time, with total cost*

$$\sum_v c(v) \leq n + \frac{m}{2} - \Theta\left(\frac{m}{\sqrt{\Delta}}\right) . \qquad \square$$

Only for small values of Δ is the constant in the $\Theta(m/\sqrt{\Delta})$ term evaluated. The linear time greedy heuristic for the FEEDBACK ARC SET problem due to Eades *et al.* [84] provides an exact bound on $\sum_i s(u_i)$, which in a number of instances, provides a tighter upper bound than that in [19]. They show that every digraph without 2-cycles has a vertex ordering $(u_1, u_2, \ldots, u_n)$ with $\sum_i s(u_i) \leq m/2 - n/6$. Using this ordering as the insertion ordering in algorithm MEDIAN PLACEMENT ORDERING, by Lemma 4.3 with $k \leq n$, we obtain the following result.

Theorem 4.4. *Every digraph without 2-cycles has a vertex ordering, which can be computed in $O(m+n)$ time, with total cost*

$$\sum_v c(v) \leq \frac{m}{2} + \frac{5n}{6} . \qquad \square$$

In the case of cubic graphs, the (more) greedy heuristic of Eades and Lin [82] determines, in $O(mn)$ time, a vertex ordering with $\sum_i s(u_i) \leq m/4$. Using this ordering as the insertion ordering in the MEDIAN PLACEMENT ORDERING algorithm produces a vertex ordering with total cost at most $n + m/4$.

[1] Berger and Shor actually consider the corresponding maximisation problem called the MAXIMUM ACYCLIC SUBGRAPH problem.

4.4 Local Minimum Approach

We now describe a method for the balanced ordering problem which finds a local minimum of the total cost. A vertex ordering $(v_1, v_2 \dots, v_n)$ of a graph G is *k-balanced* if moving any k vertices does not reduce the total cost of the ordering.

4.4.1 Undirected Graphs

Consider the following rule for moving a vertex in a vertex ordering.

M1(v,w): If $w = v^k$ is opposite to v for some k, $1 \leq k \leq \lceil c(v)/2 \rceil$ (except if $c(v)$ is odd, $k = \lceil c(v)/2 \rceil$ and $c(w) = 1$), then move v to immediately past w, as in Figure 4.3.

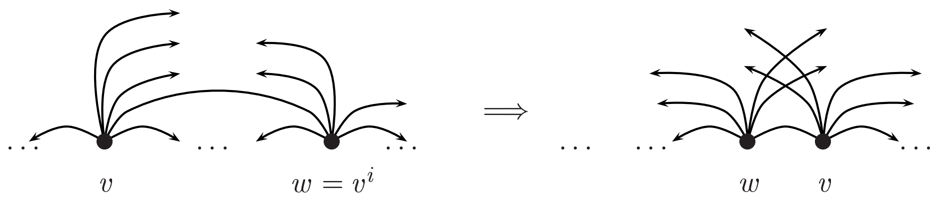


Figure 4.3: The move **M1** for a 1-5 vertex v and a 4-2 vertex $w = v^2$.

Lemma 4.5. *A vertex ordering is 1-balanced if and only if* ***M1*** *cannot be applied.*

Proof. Suppose a vertex v in a given vertex ordering, with $\gamma(v) = s(v) - p(v)$, gains α successors and loses α predecessors in the ordering. Then $c(v)$ becomes $|(s(v) + \alpha) - (p(v) - \alpha)| = |\gamma(v) + 2\alpha)|$, so the change in $c(v)$, denoted $c_\alpha(v)$, is $|\gamma(v) + 2\alpha| - |\gamma(v)|$. The following cases summarise the possible values of $c_\alpha(v)$.

1. $\gamma(v) + 2\alpha \geq 0$

   (a) $\gamma(v) \geq 0$: $c_\alpha(v) \;=\; \gamma(v) + 2\alpha + \gamma(v) \;=\; 2(\gamma(v) + \alpha)$

   (b) $\gamma(v) < 0$: $c_\alpha(v) \;=\; \gamma(v) + 2\alpha - \gamma(v) \;=\; 2\alpha$

2. $\gamma(v) + 2a < 0$

   (a) $\gamma(v) \geq 0$: $c_\alpha(v) \;=\; -\gamma(v) - 2\alpha + \gamma(v) \;=\; -2\alpha$

(b) $\gamma(v) < 0$: $c_\alpha(v) = -\gamma(v) - 2\alpha - \gamma(v) = -2(\gamma(v) + \alpha)$

Applying **M1** reduces $c(v)$ by at least $2k$ and for each i, $1 \leq i \leq k-1$, $c(v^i)$ is increased by at most two. The cost of all other vertices remains unchanged. Thus the total cost decreases by at least two. So if **M1** is applicable then the vertex ordering is not 1-balanced.

Now, suppose a given vertex ordering is not 1-balanced. Then there exists a vertex v and a neighbour $w = v^k$ of v such that moving v past w reduces the total cost. Each neighbouring vertex v^i, $1 \leq |i| \leq |k|$ (i the same sign as k), that v moves past will gain one successor and lose one predecessor if v moves to the right, or lose one successor and gain one predecessor if v moves to the left. In these respective cases the cost change at each v^i is

$$c_1(v^i) = \begin{cases} -2, & \text{if } \gamma(v^i) \leq -2; \\ 0, & \text{if } \gamma(v^i) = -1; \\ 2, & \text{if } \gamma(v^i) \geq 0. \end{cases} \qquad c_{-1}(v^i) = \begin{cases} 2, & \text{if } \gamma(v^i) \leq -2; \\ 0, & \text{if } \gamma(v^i) = 1; \\ -2, & \text{if } \gamma(v^i) \geq 0. \end{cases}$$

Suppose that v is balanced. Then the new cost of v will be $2|k|$. The cost of each vertex v^i will decrease by at most 2, so the total cost cannot decrease. Hence v cannot be balanced.

Suppose $k < 0$. Moving v past w will increase the cost of v by $2|k|$, while the decrease in cost for each vertex v^i, $k \leq i < 0$, is at most 2. Thus the increase in cost of v cannot be offset by the decrease in the cost of the neighbours of v. Hence $k > 0$.

We select the minimum $k \geq 1$ such that moving v past $w = v^k$ reduces the total cost; i.e., moving v past any $u = v^i$, $1 \leq i < k$, does not reduce the total cost. Since **M1** does reduce the total cost, each of the neighbours v^i, $1 \leq i \leq \lceil c(v)/2 \rceil$, must be not opposite to v (unless $c(v^i) = 1$ and $i = \lceil c(v)/2 \rceil$).

Suppose $k > \lceil c(v)/2 \rceil$. Then moving v past w increases the cost of each vertex v^i, $1 \leq i \leq \lceil c(v)/2 \rceil$, by 2. The new cost $c(v)$ becomes $2k - c(v)$, so the change in $c(v)$ is $2(k - c(v))$. The cost of v^i, $\lceil c(v)/2 \rceil \leq i \leq k$, can decrease by at most 2. Adding up the cost changes, it follows that the total cost cannot decrease. So $k \leq \lceil c(v)/2 \rceil$.

Suppose w is not opposite to v. Then the cost increase at w is 2 (unless $c(w) = 1$), so while $c(v)$ decreases by $2k$, the cost increase at v^i, $1 \leq i \leq k$, is 2. Hence the total cost change is 0. So w is opposite to v and $1 \leq k \leq \lceil c(v)/2 \rceil$ (except if $c(v)$ is odd, $k = \lceil c(v)/2 \rceil$ and $c(w) = 1$), and the result follows. □

We have the following immediate corollary.

Corollary 4.2. *For every vertex v in a 1-balanced vertex ordering, each of the vertices $v^1, v^2, \ldots, v^{\lfloor c(v)/2 \rfloor}$ is not opposite to v.*

We now present an algorithm for determining a 1-balanced vertex ordering. Let $\mathbf{M1}(\overrightarrow{vw})$ be a function which, for a given arc $\overrightarrow{vw} \in A(G)$, returns true if and only if v is moved past w by rule **M1**.

Algorithm 4.3. 1-BALANCED VERTEX ORDERING

Input: undirected graph G.

Output: vertex ordering of G.

Determine an arbitrary vertex ordering of G.

Set $A \leftarrow A(G)$.

while $A \neq \emptyset$ **do**

  Choose an arc $\overrightarrow{vw} \in A$.

  if $\mathbf{M1}(\overrightarrow{vw})$ **then**

    for $x \in V_G(v)$ **do** Set $A \leftarrow A \cup A_G^+(x) \cup A_G^-(x)$.

  else

    Set $A \leftarrow A \setminus \{\overrightarrow{vw}\}$.

  end-if

end-while

Output the current ordering.

Lemma 4.6. *The algorithm* 1-BALANCED VERTEX ORDERING *determines a 1-balanced vertex ordering of G in $O(\Delta^2 m)$ time.*

Proof. We shall prove that at all times the set A contains all arcs in $A(G)$ for which **M1** is possibly applicable. At the start of the algorithm this is true, since $A = A(G)$.

Consider an adjacency list representation of G where each adjacency list is ordered according to the current vertex ordering.

Suppose the arc $\overrightarrow{vw}$ is chosen from A. If $\mathbf{M1}(\overrightarrow{vw})$ is not applied then, of course, $E \setminus \{\overrightarrow{vw}\}$ contains all arcs in $A(G)$ for which **M1** is possibly applicable.

Suppose $\mathbf{M1}(\overrightarrow{vw})$ is applied, and v moves past w in the current vertex ordering. The only vertices whose cost may change are v and its neighbours, and only the adjacency lists of v and its neighbours are changed. For an arc $\overrightarrow{pq} \in A(G)$ 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 cost of every vertex adjacent to p or q does not change. Hence if $\mathbf{M1}(\overrightarrow{pq})$ is not applicable before moving v past w then $\mathbf{M1}(\overrightarrow{pq})$ will not be applicable after moving v past w.

Therefore, by adding to A the sets of arcs $A^+_G(x)$ and $A^-_G(x)$ for each neighbour x of v, we maintain the condition that A contains all arcs in $A(G)$ for which **M1** is possibly applicable. The algorithm continues until $A = \emptyset$, at which point there are no arcs for which **M1** is applicable. By Lemma 4.5, the final vertex ordering is 1-balanced.

The total cost of a vertex ordering is at most $2m$. **M1** reduces the total cost by at least two, so **M1** is applied at most m times. Whenever **M1** is applied, $O(\Delta^2)$ arcs are added to A. Hence the algorithm inserts $O(\Delta^2 m)$ arcs into A, so **M1** is checked $O(\Delta^2 m)$ times.

Using the order maintenance algorithm of Dietz and Sleator [77], the vertex ordering and adjacency lists of each vertex can be maintained in constant time under the move operation. Hence **M1** can be checked in constant time, so the algorithm runs in $O(\Delta^2 m)$ time. $\square$

We now present rules for moving two vertices in a vertex ordering.

M2: If v is opposite to w and $v < w^j < v^i < w$ for some i, j ($1 \leq i \leq \lceil c(v)/2 \rceil$, $1 \leq j \leq \lceil c(w)/2 \rceil$, $2i + 2j < c(v) + c(w) + 2$), then move v up to v^i and move w up to w^j, as in Figure 4.4.

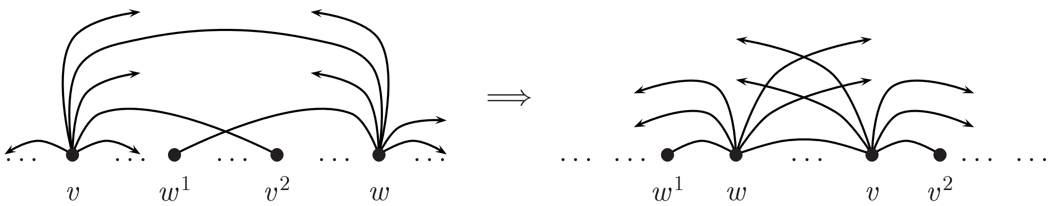


Figure 4.4: The move **M2** for a 1-5 vertex v and a 4-2 vertex w.

M3: If v is opposite to w and $v < v^i = w^j < w$ for some i, j $(1 \leq i \leq \lfloor c(v)/2 \rfloor$, $1 \leq j \leq \lfloor c(w)/2 \rfloor$, $2i + 2j < c(v) + c(w))$ then move v to immediately past v^i and move w to immediately past w^j, as in Figure 4.5.

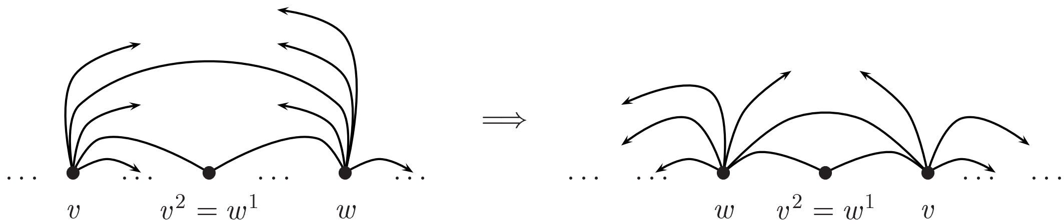


Figure 4.5: The move **M3** for a 0-5 vertex v and a 5-1 vertex w.

Applying **M2** or **M3** reduces $c(v)$ by at least $2i$ and for each k, $1 \leq k \leq i-1$, $c(v^k)$ is increased by at most two, $c(w)$ is reduced by at least $2j$ and for each k, $1 \leq k \leq j-1$, $c(w^k)$ is increased by at most two. The cost of all other vertices remains unchanged. Thus the total cost decreases by at least four.

Note that there are other rules for moving two vertices in a vertex ordering to reduce the total cost, thus **M1**, **M2** and **M3** alone cannot guarantee a 2-balanced vertex ordering. For our purposes, however, these rules suffice (see Algorithm 5.8 Diagonal Layout and Movement). Let **M2**(vw) and **M3**(vw) be functions that, given an edge $vw \in E(G)$, return true if and only if v and w move under rule **M2** and **M3**, respectively. The following algorithm determines a vertex ordering in which **M1**, **M2** and **M3** are not applicable.

Algorithm 4.4. ALMOST 2-BALANCED VERTEX ORDERING

Input: undirected graph G.

Output: vertex ordering of G.

Determine an arbitrary vertex ordering of G.

Set $E \leftarrow E(G)$.

while $E \neq \emptyset$ **do**

  Choose an edge $vw \in E$.

  if **M1**$(\overrightarrow{vw})$ or **M1**$(\overrightarrow{wv})$ or **M2**(vw) or **M3**(vw) **then**

    for $x \in V_G(v) \cup V_G(w)$ **do** Set $E \leftarrow E \cup E_G(x)$.

  else Set $E \leftarrow E \setminus \{vw\}$.

end-while

Output the current ordering.

Lemma 4.7. *The algorithm* ALMOST 2-BALANCED VERTEX ORDERING *determines a vertex ordering of* G *in* $O\big(\Delta^3 m\big)$ *time in which* ***M1***, ***M2*** *and* ***M3*** *are not applicable.*

Proof. The proof is essentially the same as that for Lemma 4.6 except that **M2** and **M3** take $O(\Delta)$ time. □

4.4.2 Directed Graphs

For a digraph without 2-cycles and of maximum outdegree two, a local minimum approach establishes the following bound for the total cost. We shall apply this result in Section 5.3.

Theorem 4.5. *A 2-balanced vertex ordering of a maximum outdegree two digraph* G *has total cost*

$$\sum_{v \in V(G)} c(v) \;\leq\; n \;.$$

Proof. In a vertex ordering of a maximum outdegree two digraph, each vertex is either a (0,2)-vertex, a (1,1)-vertex or a (0,1)-vertex. Consider a (0,2)-vertex v in a 1-balanced vertex ordering. If there is no arc $\overrightarrow{xv}$ with x between v and v^1, or there is such an x but x is opposite to v, then, as in Figure 4.6, we can move v past v^1 to reduce the total cost. ($c(v)$ becomes 0 and the cost of all other vertices does not increase.) Hence, in a 1-balanced vertex ordering, for every (0,2)-vertex v, there must be an arc $\overrightarrow{xv}$ from a (1,1)-vertex x between v and v^1. We say x *blocks* v.

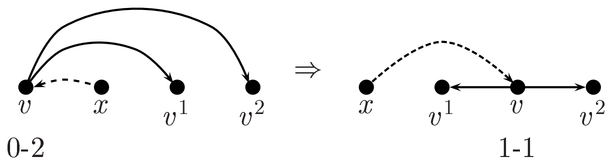


Figure 4.6: Move v past v^1.

Suppose a (1,1)-vertex x blocks distinct vertices v and w. x must be between v and w, as otherwise x would be a (0,2)-vertex. Suppose $v < x < w$. As in Figure 4.7, if we move v past v^1 and move w past w^1 then both v and w become balanced and $c(x)$ remains zero. The cost of all other vertices does not change. In particular, $c(v^1)$ and $c(w^1)$ do not change since the graph contains no 2-cycles.

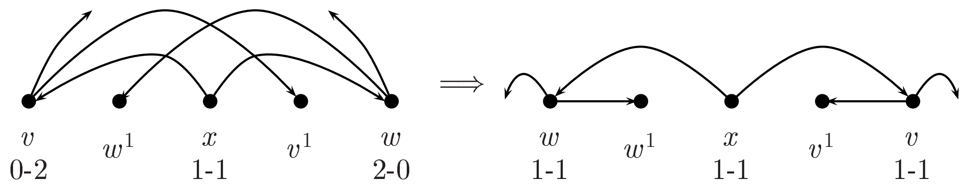


Figure 4.7: Move v past v^1 and move w past w^1.

Hence in a 2-balanced vertex ordering a (1,1)-vertex can block at most one (0,2)-vertex. The total cost of the ordering is twice the number of (0,2)-vertices plus the number of (0,1)-vertices. Since every (0,2)-vertex has a blocker which is a (1,1)-vertex, and a (1,1)-vertex blocks at most one (0,2)-vertex, the number of (0,2)-vertices is at most the number of (1,1)-vertices. So the total cost is at most the number of (0,2)-vertices plus the number of (0,1)-vertices plus the number of (1,1)-vertices, which is at most n. □

Using a similar analysis to that in Lemma 4.6, it is easily seen that the algorithm described in the previous proof runs in $O(n)$ time. We therefore have the following result.

Corollary 4.3. *A vertex ordering of a maximum outdegree two digraph with total cost at most n can be determined in $O(n)$ time.* □

Chapter 5

The General Position Model for Three-Dimensional Orthogonal Point-Drawing

In this chapter we describe the general position model for producing 3-D orthogonal point-drawings. We present a number of algorithms for producing orthogonal point-drawings in this model. Among other results we establish the best known upper bound for the total number of bends in 3-D orthogonal point drawings, and the best known upper bound for the volume of 3-bend orthogonal point-drawings.

A 3-D orthogonal point-drawing is said to be a *general position* 3-D orthogonal point-drawing if no two vertices lie in a common grid plane. We are interested in the following problem.

Problem 5.1. BEND-MINIMUM GENERAL POSITION 3-D POINT-DRAWING

Instance: A graph G with $\Delta(G) \leq 6$.

Output: A general position 3-D orthogonal point-drawing of G with the minimum number of bends.

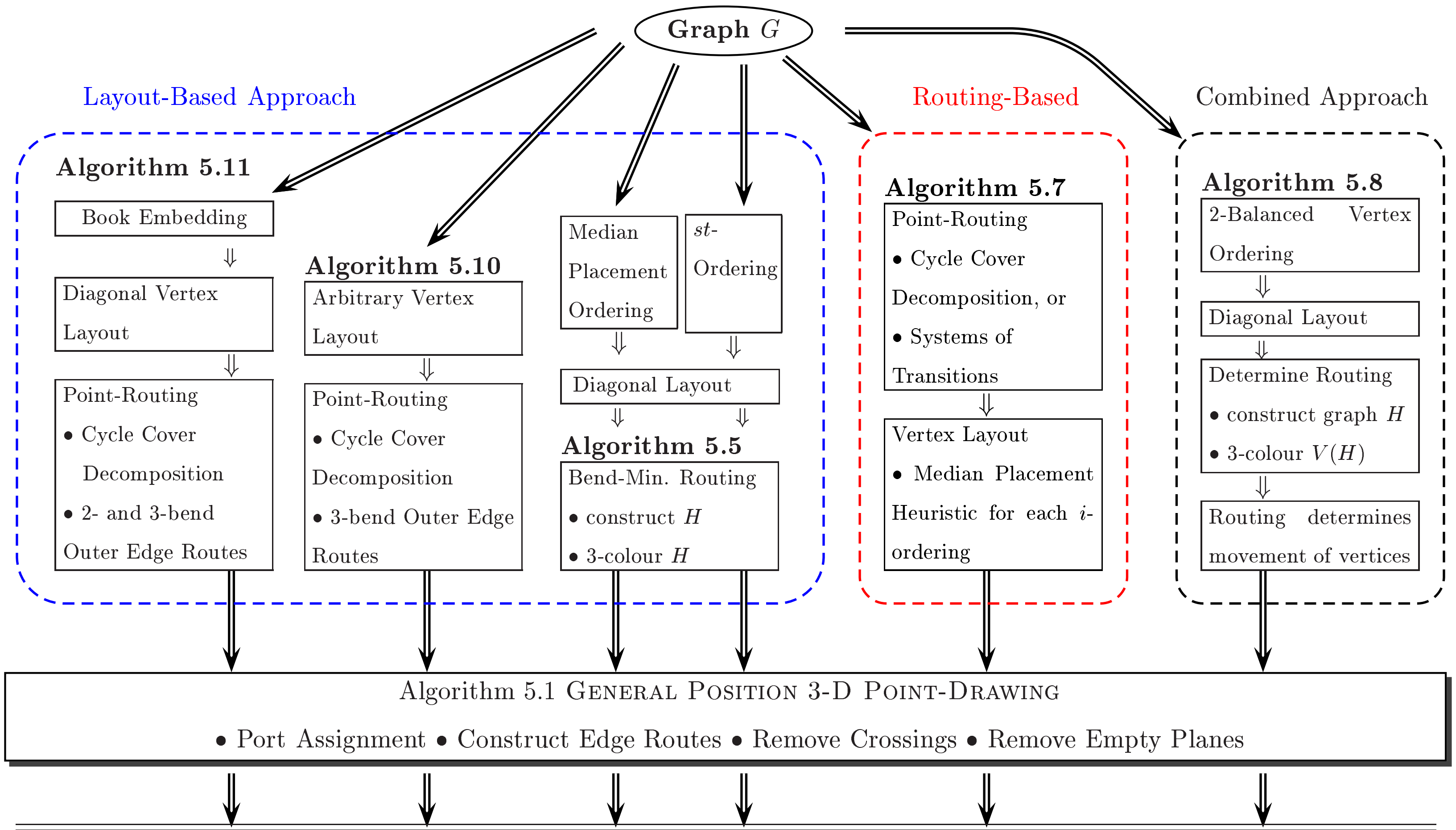
This chapter is organised as follows. Section 5.1 describes a representation for general position 3-D orthogonal point-drawings, thus forming a foundation for the main algorithms to follow. We initially concentrate on the problem of minimising the total number of bends per edge in general position 3-D orthogonal point-drawings. As discussed in Section 3.4.4, algorithms for producing general position orthogonal drawings can be classified as layout-based or routing-based.

In Section 5.2 we present our layout-based approach for 3-D orthogonal point-drawing. Firstly, we describe an algorithm which minimises the total number of bends for a fixed diagonal vertex layout. We also describe a method, based on a maximum-clique formulation, for searching for bend-minimum drawings given a fixed general position vertex layout.

Our routing-based approach for producing 3-D orthogonal point-drawings is described in Section 5.3. The DIAGONAL LAYOUT AND MOVEMENT algorithm described in Section 5.4 combines the layout- and routing-based approaches. It establishes the best known upper bound for the total number of bends in 3-D orthogonal point-drawings of simple graphs, and is a 7/6-approximation algorithm for the problem BEND-MINIMUM GENERAL POSITION 3-D POINT-DRAWING. Furthermore, the same algorithm produces 2-bend point-drawings for maximum degree five graphs.

In Section 5.5 we consider the problem of minimising the maximum number of bends per edge route in a orthogonal point-drawing. We present two algorithms, both of which follow the layout-based approach. The first algorithm, given a fixed general position vertex layout, determines an orthogonal point-drawing with three bends per edge. We then describe a modification of the 3-BENDS algorithm of Eades *et al.* [86, 87] which produces 3-D orthogonal point-drawings using a diagonal vertex layout with $n^3 + O(n^{5/2})$ volume. This is the best known upper bound for the volume of 3-bend 3-D orthogonal point-drawings.

Finally, in Section 5.6 we present lower bounds for the number of bends in general position orthogonal point-drawings. These results have important implications for the nature of any solution to the 2-bends problem (see Section 3.5.1). Figure 5.1 provides an overview of the algorithms presented in this chapter.



Max. Bends	3	3	4	4	4	4
Avg. Bends	3	3	$2\frac{7}{12}$	$2\frac{2}{3}$	$2\frac{1}{2}$	$2\frac{1}{3}$
Volume	$n^3 + O(n^{5/2})$	$8n^3$	$\left(\frac{19}{12}n\right)^3$	$\left(\frac{5}{3}n\right)^3$	$\left(\frac{3}{2}n\right)^3$	$\left(\frac{4}{3}n\right)^3$

Figure 5.1: Algorithms for general position 3-D orthogonal point-drawing. The bounds are for 6-regular graphs.

5.1 Representation

Consider a general position 3-D orthogonal point-drawing of a graph G with maximum degree $\Delta(G) \leq 6$. Since no two vertices share a common coordinate, this drawing defines X-, Y-, and Z-vertex orderings of G, representing the relative coordinates of the vertices. The assignment of ports to edge routes defines a (non-proper) 3-colouring of $A(G)$, where an arc $\overrightarrow{vw} \in A(G)$ is coloured $i \in \{X, Y, Z\}$ if the edge route vw uses an i-port at v. Clearly, for each vertex $v \in V(G)$, there are at most two arcs $\overrightarrow{vw} \in A(G)$ receiving the same colour. We therefore represent a general position 3-D orthogonal point-drawing of G by:

- A *(3-D general position) vertex layout*, consisting of X-, Y-, and Z-vertex orderings $(x_1, x_2, \ldots, x_n)$, $(y_1, y_2, \ldots, y_n)$ and $(z_1, z_2, \ldots, z_n)$ of G.

- A *(3-D) point-routing*, consisting of a 3-colouring of $A(G)$ such that for each vertex $v \in V(G)$, there are at most two arcs $\overrightarrow{vw} \in A(G)$ receiving the same colour; i.e., $\Delta\left(\overleftrightarrow{G}[i]\right) \leq 2$, for each colour $i \in \{X, Y, Z\}$.

In a general position vertex layout, for an edge vw to have a 2-bend edge route, it is necessary for the reversal arcs $\overrightarrow{vw}, \overrightarrow{wv} \in A(G)$ to be coloured differently. If for every edge $vw \in E(G)$, the reversal arcs $\overrightarrow{vw}, \overrightarrow{wv} \in A(G)$ are coloured differently, then we call the point-routing a *2-bend* point-routing.

As discussed in Section 3.4.4, algorithms for producing general position 3-D orthogonal drawings can be classified as layout-based or routing-based. Our layout-based algorithms determine a vertex layout initially, followed by the computation of a point-routing. Our routing-based algorithm determines the vertex layout with respect to a pre-determined point-routing.

The following algorithm forms the final step of all our algorithms. Given a vertex layout and a point-routing, it constructs a layout- and routing-preserving general position 3-D orthogonal point-drawing (possibly with crossings) in linear time. By a sequence of port assignment swaps, the algorithm then removes all edge route crossings from the drawing in quadratic time in the worst case.

Algorithm 5.1. General Position 3-D Point-Drawing

Input: • graph G with $\Delta(G) \leq 6$.
• general position 3-D vertex layout of $V(G)$.
• point-routing of G (with 3 colours).

Output: general position 3-D orthogonal point-drawing of G

---

1. For each vertex $v \in V(G)$,
   if $v = x_i = y_j = z_k$ then initially position v at $(3i, 3j, 3k)$.
2. Apply Algorithm 5.2 DETERMINE PORT ASSIGNMENT.
3. Apply Algorithm 5.3 CONSTRUCT EDGE ROUTES.
4. Apply Algorithm 5.4 POINT-DRAWING REMOVE EDGE CROSSINGS.
5. Delete each grid-plane not containing a vertex or a bend.

In what follows we describe the details of the components of Algorithm GENERAL POSITION 3-D POINT-DRAWING.

5.1.1 Edge Routes

As a first step in constructing edge routes for a given vertex layout and point-routing of a graph, we determine the assignment of ports to arcs. The following algorithm assigns ports to arcs so that, whenever possible, the port at a vertex v assigned to an arc $\overrightarrow{vw}$ points toward w. Recall that $A_G(v)[i]$ is the set of outgoing arcs at a vertex $v \in V(G)$ which are coloured $i \in \{X, Y, Z\}$.

Algorithm 5.2. DETERMINE PORT ASSIGNMENT

Input: • graph G with $\Delta(G) \leq 6$
• general position 3-D vertex layout of G
• point-routing of G (with 3 colours)

Output: routing-preserving assignment of ports to $A(G)$

---

for each vertex $v \in V(G)$, **for each** colour $i \in \{X, Y, Z\}$ **do**

  if $A_G(v)[i] = \{\overrightarrow{vw}\}$ **then**

    Assign to $\overrightarrow{vw}$ the i-port at v pointing towards w.

  else if $A_G(v)[i] = \{\overrightarrow{vu}, \overrightarrow{vw}\}$ $(u \neq w)$ **then**

    if v is between u and w in the i-ordering **then**

      Assign to $\overrightarrow{vu}$ and $\overrightarrow{vw}$ the i-ports at v pointing towards u and w.

    else if $\overrightarrow{uv} \in A_G(u)[i]$ **then**

      Assign to $\overrightarrow{vu}$ the i-port at v pointing away from u.

      Assign to $\overrightarrow{vw}$ the i-port at v pointing towards w.

    else if $\overrightarrow{wv} \in A_G(w)[i]$ **then**

      Assign to $\overrightarrow{vw}$ the i-port at v pointing away from w.

      Assign to $\overrightarrow{vu}$ the i-port at v pointing towards u.

    else

      Arbitrarily assign the i-ports at v to $\overrightarrow{vu}$ and $\overrightarrow{vw}$.

    end-if

  end-if

end-for

The following algorithm, for a given port assignment, determines each edge route with the minimum number of bends.

Algorithm 5.3. Construct Edge Routes

Input: • graph G with $\Delta(G) \leq 6$
- general position 3-D vertex layout of G
- port assignment for G

Output: general position 3-D point-drawing of G (possibly with 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 Figure 5.2.

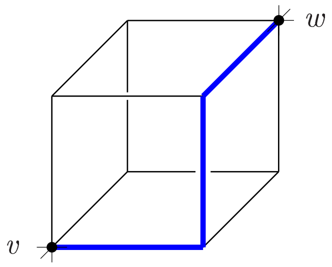
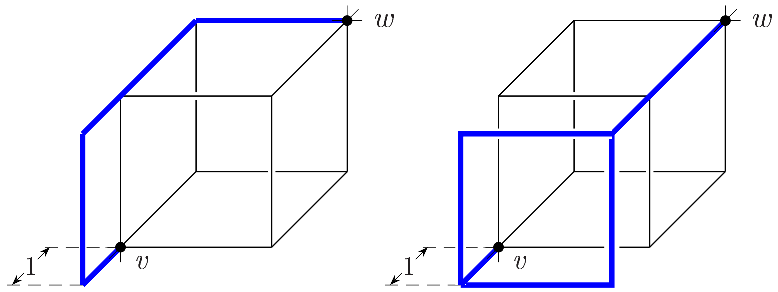


Figure 5.2: 2-bend edge route vw.

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 a 3-bend edge route for vw, said to be *anchored* at v, as illustrated in Figure 5.3.

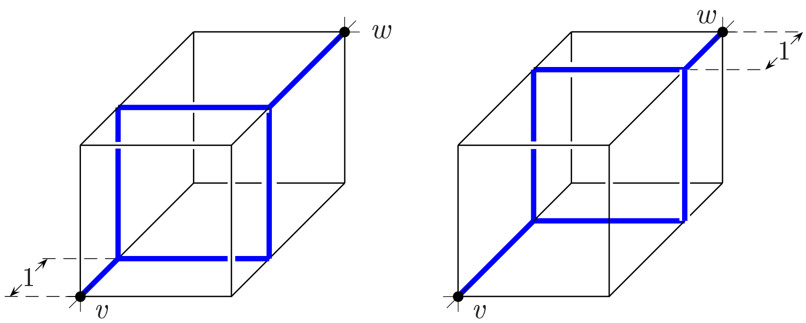


(a) perpendicular ports

(b) parallel ports

Figure 5.3: 3-bend edge routes vw anchored at v.

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, as in Figure 5.4, route vw with the 3-bend edge route said to be *anchored* at the chosen vertex.



(a) anchored at v

(b) anchored at w

Figure 5.4: 3-bend edge routes.

4. If port($\overrightarrow{vw}$) points away from w and port($\overrightarrow{wv}$) points away from v then use a 4-bend edge route for vw as in Figure 5.5. We say the edge route vw is *anchored* at v and at w.

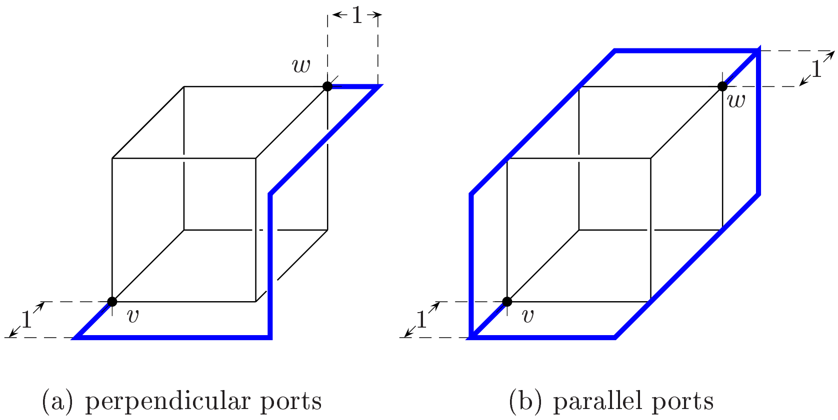


Figure 5.5: 4-bend edge routes vw anchored at v and at w.

For a given assignment of ports, each edge route uses the minimum number of bends, so in a general position 3-D orthogonal point-drawing the only edge routes needed are those described above (assuming that edge crossings are allowed). If the edge route vw is anchored at v then we say the arc $\overrightarrow{vw}$ has been *anchored*. Note that if for some edge vw, the arcs $\overrightarrow{vw}$ and $\overrightarrow{wv}$ are coloured the same, then the edge route vw needs at least three bends; i.e., at least one of $\overrightarrow{vw}$ and $\overrightarrow{wv}$ is anchored. The drawings produced have precisely $2m + k$ bends where k is the number of anchored arcs.

Lemma 5.1. *The algorithms* DETERMINE PORT ASSIGNMENT *and* CONSTRUCT EDGE ROUTES *construct a general position 3-D orthogonal point-drawing (possibly with edge crossings) with precisely one anchored arc for each instance of the following conditions (see Figure 5.6).*

- For some vertex v and colour $i \in \{X, Y, Z\}$,

  (a) $\overrightarrow{vu}, \overrightarrow{vw} \in A_G(v)[i]$ $(u \neq w)$, and (5.1)

  (b) v is not between u and w in the i-ordering.

- For some edge $vw \in E(G)$ and colour $i \in \{X, Y, Z\}$,

  (a) $\overrightarrow{vw}, \overrightarrow{vu} \in A_G(v)[i]$ $(w \neq u)$,

  (b) $\overrightarrow{wv}, \overrightarrow{wx} \in A_G(w)[i]$ $(v \neq x)$, (5.2)

  (c) v is between u and w in the i-ordering, and

  (d) w is between v and x in the i-ordering.

Proof. In Algorithm Construct Edge Routes, there is one anchored arc in Cases 2 and 3, and two anchored arcs in Case 4. Case 3 occurs precisely when (5.2) occurs. If Case 2 occurs there is one instance of (5.1), and if Case 4 occurs then there are two instances of (5.1). Hence there is one anchored arc for each instance of (5.1) and (5.2). □

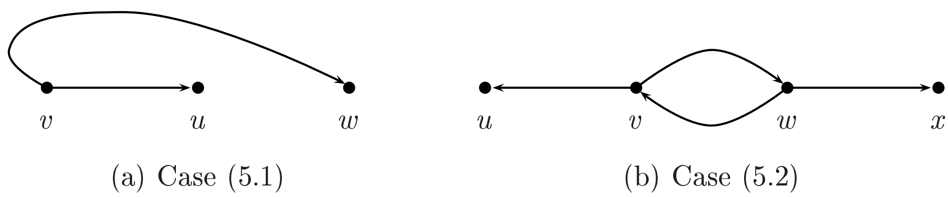


Figure 5.6: Cases with anchored arcs (with vertices in the i-ordering and arcs coloured i).

5.1.2 Removing Edge Crossings

We now characterise all possible intersections between edge routes constructed by the previous algorithm. As illustrated in Figure 5.7, 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 that the $(X = 3i - 1)$-plane, the $(X = 3i)$-plane and the $(X = 3i + 1)$-plane *belong* to v, and similarly for Y- and Z-coordinates. 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.

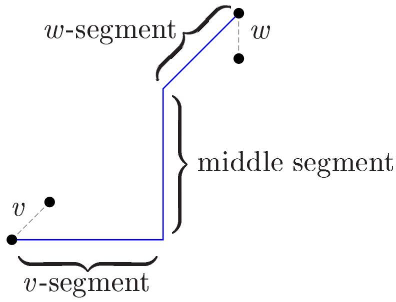


Figure 5.7: Segments of the 2-bend component of an edge route.

Suppose in a drawing produced by the algorithm CONSTRUCT EDGE ROUTES the edge routes vw and xy intersect. If vw and xy are non-adjacent 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.

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, so we need only consider intersections between non-unit-length segments.

Case 1 — The v-segments of vu and vw intersect: Clearly both of vu and vw must be anchored at v, and they must intersect as in Figure 5.8. Swapping the ports assigned to $\overrightarrow{vu}$ and $\overrightarrow{vw}$, and removing both anchors eliminates the edge crossing. Doing so introduces no new edge crossings.

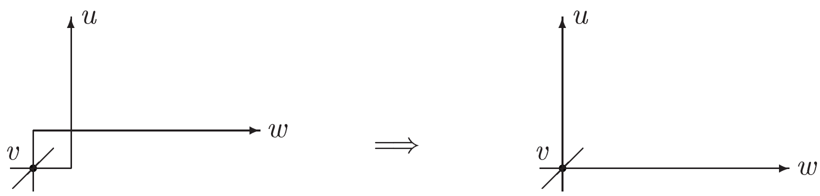


Figure 5.8: Case 1 — Rerouting intersecting v-segments (which must be anchored).

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

Case 2(a) — vw is not anchored: Clearly vu must be anchored. Since the middle segment of vu is parallel with the port assigned to $\overrightarrow{vu}$, the ports assigned to $\overrightarrow{vu}$ and $\overrightarrow{uv}$ must be perpendicular. As shown in Figure 5.9, by swapping the ports assigned to $\overrightarrow{vu}$ and $\overrightarrow{vw}$, anchoring $\overrightarrow{vw}$, and unanchoring $\overrightarrow{vu}$, the edge crossing is removed. Note that the new edge routes contain no new grid points belonging to u or w, so there are no new edge crossings introduced by this operation.

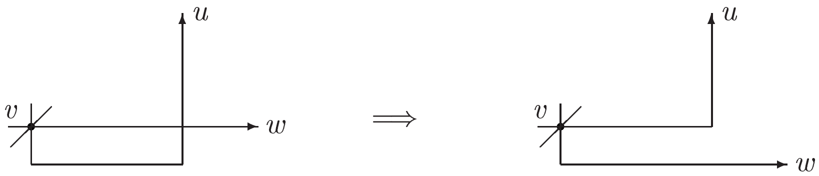


Figure 5.9: Case 2(a) — Rerouting intersecting v-segment of vw and middle segment of vu if vw is not anchored.

Case 2(b) — $\boldsymbol{vw}$ is anchored (see Figure 5.10): The edge route vu may be anchored at v, and if it is, then as in Case 2(a), the ports assigned to $\overrightarrow{vu}$ and $\overrightarrow{uv}$ must be perpendicular. By swapping the ports at v assigned to $\overrightarrow{vu}$ and $\overrightarrow{vw}$ the edge crossing is removed. The arc $\overrightarrow{vu}$ is now not anchored, if $\overrightarrow{vu}$ was anchored then $\overrightarrow{vw}$ is now anchored, and if $\overrightarrow{vu}$ was unanchored then $\overrightarrow{vw}$ is now unanchored. Hence an anchor, and thus a bend, is eliminated. Note that this operation may introduce new edge crossings between uv and some other edge incident to u.

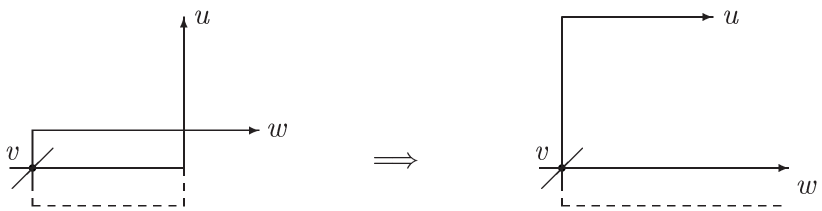


Figure 5.10: Case 2(b) — Rerouting intersecting v-segment of anchored vw and middle segment of vu.

Case 3 — The middle segments of $\boldsymbol{vu}$ and $\boldsymbol{vw}$ intersect (See Figure 5.11): Note that $\overrightarrow{vu}$ and $\overrightarrow{vw}$ may or may not be anchored. If $\overrightarrow{vu}$ is anchored then the edge route vu must use perpendicular ports at v and u, and similarly, if $\overrightarrow{vw}$ is anchored then the edge route vw must be assigned perpendicular ports at v and w. Swapping the ports assigned to $\overrightarrow{vu}$ and $\overrightarrow{vw}$, and swapping any anchors, removes the edge crossing.

Note that the sum of the lengths of the new middle segments of vu and vw is strictly

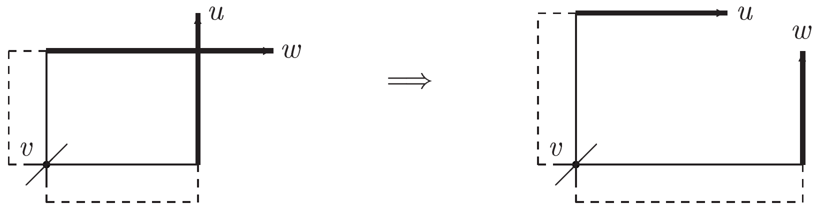


Figure 5.11: Rerouting intersecting middle-segments.

less than the previous sum. This operation may introduce new edge crossings between uv and some other edge at u, or between wv and some other edge at w.

The following algorithm summarises the crossing removal phase of our algorithm.

Algorithm 5.4. POINT-DRAWING REMOVE EDGE CROSSINGS

Input: • graph G with $\Delta(G) \leq 6$

• general position 3-D orthogonal point-drawing of G (possibly with crossings) generated by the CONSTRUCT EDGE ROUTES algorithm.

Output: general position 3-D orthogonal point-drawing of G (without crossings).

$V \leftarrow V(G)$

while $V \neq \emptyset$ **do**

  Choose $v \in V$, and set $V \leftarrow V \setminus \{v\}$.

  for each Case 2(b) or Case (3) crossing between edges vu and vw **do**

    Swap the ports at v assigned to $\overrightarrow{vu}$ and $\overrightarrow{vw}$.

    Reroute the edge routes vu and vw according

      to Algorithm 5.3 CONSTRUCT EDGE ROUTES.

    Set $V \leftarrow V \cup \{v, u, w\}$.

  end-for

end-while

for each vertex $v \in V(G)$ **do**

  for each Case (1) or Case 2(a) crossing between edges vu and vw **do**

    Swap the ports at v assigned to vu and vw.

  end-for

end-for

Lemma 5.2. *The algorithm* Point-Drawing Remove Edge Crossings *removes all crossings from the given 3-D orthogonal point-drawing in $O(n^2)$ time.*

Proof. In Case 3 and Case 2(b), but not Cases 2(a) and Case 1, swapping ports may create new edge route crossings between uv and some other edge route incident to u, or similarly at w. Therefore removing all Case 3 and Case 2(b) crossings in the first phase of the algorithm, and removing all Cases 2(a) and Case 1 edge crossings in the second phase of the algorithm, removes all crossings from the drawing.

In Case 3 the sum of the lengths of the middle segments of vu and vw is reduced (see the segments in bold). The length of each middle segment is $O(n)$ and there are at most $3n$ middle segments in total, so the sum of the lengths of the middle segment is $O(n^2)$.

In Case 2(b) (and also in Case 1) at least one anchored arc (and thus a bend) is eliminated. The number of anchored arcs is at most $6n$.

Hence the sum of the lengths of the middle segments plus the number of anchored arcs is $O(n^2)$, and every Case 3 or Case 2(b) port swap reduces this number by at least one. Therefore the algorithm executes $O(n^2)$ Case 3 or Case 2(b) port swaps. With each such port swap three vertices are added to V for re-checking. Hence, Case 2(b) and Case 3 needs to be checked for some vertex $O(n^2)$ times. To check Case 2(b) and Case 3 for a particular vertex v involves comparing the coordinates of a $O(1)$ number of pairs of edge routes incident to v. Hence the first phase of the algorithm takes $O(n^2)$ time.

Similarly, for a particular vertex, Case 1 and Case 2(a) can be checked in constant time. So the second phase of the algorithm takes $O(n)$ time, and the algorithm removes all edge crossings in $O(n^2)$ time. □

We can now prove the main result of this section.

Theorem 5.1. *Suppose G is a graph with $\Delta(G) \leq 6$, and we are given a general position vertex layout and point-routing of G with k instances of* (5.1) *and* (5.2)*. Then the algorithm* General Position 3-D Point-Drawing *will, in $O(n^2)$ time, construct a layout-preserving 3-D orthogonal point-drawing of G with at most four bends per edge route and at most $2m+k$ bends in total. The bounding box volume is at most $(n+k/3)^3$.*

Proof. As discussed earlier there is one anchored arc for each occurrence of (5.1) and (5.2). Clearly, a grid-plane not containing a vertex or a bend can be removed without affecting the drawing. 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 (i.e., coloured X) with its v-segment lying in this plane. Similarly for Y-planes and Z-planes. Therefore, after removing grid-planes not containing a vertex or a bend, 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, $i \in \{X, Y, Z\}$. It is well-known that of the boxes with fixed sum of side length the cube has maximum volume (see for example Kazarinoff [126]). So if k is the total number of anchored arcs then the bounding box volume is maximised when $k_X = k_Y = k_Z = k/3$, so the bounding box volume is at most $(n + k/3)^3$. □

5.2 Layout-Based Algorithms

We now describe our layout-based approach for producing general position 3-D orthogonal point-drawings. Here we are concerned with the following problem.

Problem 5.2. LAYOUT-BASED GENERAL POSITION 3-D POINT-DRAWING

Instance: A general position 3-D vertex layout of a graph G with $\Delta(G) \leq 6$.

Output: A layout-preserving 3-D orthogonal point-drawing of G with the minimum number of bends.

This problem amounts to finding a point-routing of G with the minimum number of instances of (5.1) and (5.2). We conjecture that it is NP-hard.

5.2.1 Diagonal General Position Vertex Layout

We initially consider layout-based algorithms with a diagonal layout of the vertices. A diagonal layout was first used for 3-D orthogonal point-drawing by the 3-BENDS algorithm of Eades *et al.* [86, 87]. Consider a diagonal layout of a maximum degree six graph G with corresponding vertex ordering $<$. A vertex $v \in V(G)$ has $\max\{\max\{s_<(v), p_<(v)\} - 3, 0\}$ arcs incident to v which must be assigned a port at v which point away from their destination. Such arcs must be anchored. Each edge route has at least two bends and each anchored arc contributes one further bend. Therefore the total number of bends in a diagonal layout 3-D orthogonal point-drawing is at least

$$2m \; + \sum_{v \in V(G)} \max\left\{\max\left\{s_<(v), p_<(v)\right\} - 3, 0\right\} \quad . \tag{5.3}$$

The following algorithm determines a diagonal layout 3-D orthogonal point-drawing with precisely this number of bends, thus solving the LAYOUT-BASED GENERAL POSITION 3-D POINT-DRAWING problem in the case of a diagonal layout.

Algorithm 5.5. Diagonal General Position 3-D Point-Drawing

Input: • graph G with $\Delta(G) \leq 6$
• vertex ordering $<$ of G

Output: diagonal layout 3-D point-drawing of G

---

1. Construct a graph H with $V(H) = A(G)$.
2. For each vertex $v \in V(G)$, add cliques $\left\{vv^A, vv^B, vv^C\right\}$ and $\left\{vv^D, vv^E, vv^F\right\}$ to $E(H)$, according to Table 5.1. (Refer to Section 4.1 for the relevant definitions. If $\deg(v) < 6$ then some of vv^A, vv^B, vv^C, vv^D, vv^E and vv^F will not be defined, so the above-mentioned cliques may be empty or consist of a single edge.)

Table 5.1: Definition of vv^A, vv^B, vv^C, vv^D, vv^E and vv^F

v	vv^A	vv^B	vv^C	vv^D	vv^E	vv^F
α-vertex ($\alpha \leq 3$)	vv^{-3}	vv^{-2}	vv^{-1}	vv^1	vv^2	vv^3
4-vertex	vv^{-2}	vv^{-1}	vv^1	vv^2	vv^3	vv^4
5-vertex	vv^{-1}	vv^1	vv^2	vv^3	vv^4	vv^5
6-vertex	vv^1	vv^2	vv^3	vv^4	vv^5	vv^6

3. For each edge $vw \in E(G)$, add the edge $\{\overrightarrow{vw}, \overrightarrow{wv}\}$ to $E(H)$ (called an 'r'-edge), as illustrated in Figure 5.12.
4. Determine a point-routing of G from a vertex 3-colouring of H.

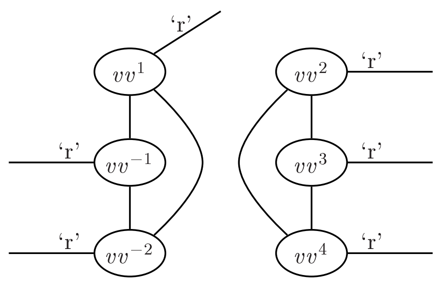


Figure 5.12: The subgraph of H corresponding to a 2-4 vertex v.

5. Apply Algorithm 5.1 General Position 3-D Point Drawing.

Lemma 5.3. *The algorithm* Diagonal General Position 3-D Point-Drawing *determines, in $O(n)$ time, a diagonal layout 3-D orthogonal point-drawing of G with*

$$2m + \sum_{v \in V(G)} \max\{\max\{s_<(v), p_<(v)\} - 3, 0\} \quad .$$

bends and at most four bends per edge route. The volume is

$$\left(n + \frac{1}{3} \sum_{v \in V(G)} \max\{\max\{s_<(v), p_<(v)\} - 3, 0\} \right)^3 \quad .$$

Proof. A vertex of H is incident with one 'r'-edge and at most two unlabelled edges, so the graph H has maximum degree $\Delta(H) \leq 3$, and is not K_4, so by Brooks' Theorem [47], H is vertex 3-colourable. The proof of Brook's Theorem due to Lovász [147] and simplified by Bryant [49] describes an algorithm for vertex 3-colouring H in $O(|E(H)|) = O(n)$ time. The 3-colouring of $V(H)$ determines a 3-colouring of $A(G)$. The unlabelled edges ensure that at most two arcs at a vertex v can receive the same colour, so the colouring is a point-routing of G.

Applying Theorem 5.1 with the given diagonal layout and this point-routing determines a 3-D orthogonal point-drawing with $2m + k$ bends where k is the number of instances of (5.1) and (5.2). Since all pairs of reversal arcs are coloured differently there are no instances of (5.2).

Suppose $\overrightarrow{vu}, \overrightarrow{vw} \in A_G(v)[i]$ ($u \neq w$) for some vertex v and colour $i \in \{X, Y, Z\}$. Then we can assume $\overrightarrow{vu} \in \{vv^A, vv^B, vv^C\}$ and $\overrightarrow{vw} \in \{vv^D, vv^E, vv^F\}$. An instance of

(5.1) occurs if v is not between u and w in the i-ordering. This occurs if and only if $vu = vv^i$ for some i, $1 \leq i \leq \max\{\max\{s_<(v), p_<(v)\} - 3, 0\}$. So there are

$$k = \sum_{v \in V(G)} \max\{\max\{s_<(v), p_<(v)\} - 3, 0\} \quad .$$

anchored arcs. By Theorem 5.1 the volume bound holds.

Consider the algorithm Point-Drawing Remove Edge Crossings applied with a diagonal layout. Clearly Case 3 cannot occur. If Case 2(b) occurs then $\overrightarrow{vw}$ must be anchored and port($\overrightarrow{vw}$) points towards w. However, in Algorithm Diagonal General Position 3-D Point-Drawing if an arc $\overrightarrow{vw}$ is anchored then port($\overrightarrow{vw}$) points away from w. Hence Cases 3 and 2(b) cannot occur when we apply Point-Drawing Remove Edge Crossings, so it takes $O(n)$ time. Therefore each step of Diagonal Layout 3-D Point-Drawing takes $O(n)$ time. The result follows. □

Combining (5.3) and Lemma 5.3 we obtain the following result.

Theorem 5.2. *The problem LAYOUT-BASED GENERAL POSITION 3-D POINT-DRAWING can be solved in $O(n)$ time in the case of a diagonal layout.* □

We now can characterise those 2-bend 3-D orthogonal point-drawings with a diagonal layout, a result first established by Wood [220].

Corollary 5.1. *A diagonal layout of a graph G admits a 2-bend 3-D orthogonal point-drawing if and only if every vertex v in the corresponding vertex ordering has $s(v) \leq 3$ and $p(v) \leq 3$.*

Proof. By Theorem 5.2, a diagonal layout admits a 2-bend point-drawing if and only if, for every vertex v, $\max\{\max\{s(v), p(v)\} - 3, 0\} = 0$; i.e., $\max\{s(v), p(v)\} \leq 3$; i.e., $s(v) \leq 3$ and $p(v) \leq 3$. □

If we apply algorithm Diagonal General Position 3-D Point-Drawing with a diagonal layout whose vertex ordering is determined using st-orderings (see Section 4.2) we obtain the following result.

Corollary 5.2. *If a graph G with maximum degree $\Delta(G) \leq 6$ has c connected components and k end-blocks, then there exists a diagonal layout 3-D orthogonal point-drawing* *of G, which can be determined in $O(n)$ time, with at most $3m - n + c + k$ bends and at most $((2n + m + c + k)/3)^3$ volume. If G is 6-regular and has a constant number of biconnected components then the number of bends is at most $8m/3 + O(1)$ and the volume is at most $(5n/3)^3 + O\left(n^{5/2}\right)$.*

Proof. Firstly, remove each vertex with degree one and its incident edge from G. Suppose the remaining graph, called G', has n' vertices, m' edges, c' connected components and k' end-blocks. Let n_i be the number of vertices $v \in V(G')$ with $\deg_{G'}(v) = i$. By Lemma 4.2, G' has a vertex ordering $<$ with $c' + k'$ vertices having zero predecessors or zero successors. For such a vertex v, $\max\{s(v), p(v)\} = \deg(v)$, so

$$\max\{\max\{s(v), p(v)\} - 3, 0\} = \begin{cases} 0, & \text{if } \deg_G(v) \leq 3; \\ 1, & \text{if } \deg_G(v) = 4; \\ 2, & \text{if } \deg_G(v) = 5; \\ 3, & \text{if } \deg_G(v) = 6. \end{cases}$$

For all other vertices v we have

$$\max\{\max\{s(v), p(v)\} - 3, 0\} \leq \begin{cases} 0, & \text{if } \deg_G(v) \leq 4; \\ 1, & \text{if } \deg_G(v) = 5; \\ 2, & \text{if } \deg_G(v) = 6. \end{cases}$$

Hence

$$\sum_{v \in V(G')} \max\{\max\{s(v), p(v)\} - 3, 0\} \leq n_5 + 2n_6 + c' + k' \ .$$

If we determine a 3-D orthogonal point-drawing of G' with Algorithm 5.5 Diagonal General Position 3-D Point-Drawing using the vertex ordering $<$, then by Lemma 5.3 there is at most

$$2m' + \sum_{v \in V(G')} \max\{\max\{s(v), p(v)\} - 3, 0\} \leq 2m' + n_5 + 2n_6 + c' + k'$$

bends. Now,

$$0 \leq n_3 + 2n_4 + n_5$$

$$\begin{aligned}2n_5 + 4n_6 &\leq n_3 + 2n_4 + 3n_5 + 4n_6 \\ 2n_5 + 4n_6 &\leq (2n_2 + 3n_3 + 4n_4 + 5n_5 + 6n_6) - 2n' \\ 2n_5 + 4n_6 &\leq 2m' - 2n' \\ n_5 + 2n_6 &\leq m' - n' \\ 2m' + n_5 + 2n_6 + c' + k' &\leq 3m' - n' + c' + k' \quad .\end{aligned}$$

So the number of bends in the drawing of G' is at most $3m' - n' + c' + k'$. It is easily seen that the vertices with degree one can be reinserted into the diagonal layout, and each incident edge routed with two bends. Hence the number of bends in the drawing of G is $3m' - n' + c' + k' + 2(m - m') = m' - n' + c' + k' + 2m$.

Now, $(n - n') = (c - c') + (k - k')$. So $(n - n') \leq (m - m') + (c - c') + (k - k')$, and hence $m' - n' + c' + k' \leq m - n + c + k$. So the number of bends in the drawing of G is at most $3m - n + c + k$.

The number of anchored arcs is at most $m - n + c + k$, so the volume of the drawing of G is at most $(n + (m - n + c + k)/3)^3 = ((2n + m + c + k)/3)^3$.

If G is 6-regular and has a constant number of biconnected components then the number of bends is $8m/3 + O(1)$ and the volume is $(5n/3)^3 + O\big(n^{5/2}\big)$.

By Lemmas 4.2 and 5.3, the st-orderings and the drawing itself can be determined in $O(n)$ time, respectively. ☐

If we use Algorithm 4.1 MEDIAN PLACEMENT ORDERING to determine the vertex ordering of a diagonal layout, we obtain the following result.

Corollary 5.3. *A graph G with maximum degree $\Delta(G) \leq 6$ has a diagonal layout 3-D orthogonal point-drawing, which can be determined in $O(n)$ time, with at most $5m/2 + n/4$ bends and at most $(m/6 + 13n/12)^3$ volume. For 6-regular graphs the number of bends is at most $31m/12$ and the volume is at most $(19n/12)^3$.*

Proof. Let $<$ be a vertex ordering of G determined by Algorithm 4.1 MEDIAN PLACEMENT ORDERING (with insertion ordering determined by Algorithm 4.2 INSERTION ORDERING). Suppose G has n_i vertices with degree i. Determine a diagonal layout 3-D point-drawing, with corresponding vertex ordering $<$, using the algorithm DIAGONAL GENERAL POSITION 3-D POINT-DRAWING. By Lemma 5.3, the number of anchors is

$$\sum_{v \in V(G)} \max\{\max\{s_<(v), p_<(v)\} - 3, 0\} \ .$$

A degree one or two vertex v has $\max\{s_<(v), p_<(v)\} \geq 1$, so

$$\max\{\max\{s_<(v), p_<(v)\} - 3, 0\} \leq (\max\{s_<(v), p_<(v)\} - 3) + 2 \ .$$

A degree three or four vertex v has $\max\{s_<(v), p_<(v)\} \geq 2$, so

$$\max\{\max\{s_<(v), p_<(v)\} - 3, 0\} \leq (\max\{s_<(v), p_<(v)\} - 3) + 1 \ .$$

A degree five or six vertex v has $\max\{s_<(v), p_<(v)\} \geq 3$, so

$$\max\{\max\{s_<(v), p_<(v)\} - 3, 0\} = \max\{s_<(v), p_<(v)\} - 3 \ .$$

Hence the number of anchored arcs is at most

$$\begin{aligned}
& \sum_{v \in V(G)} (\max\{s_<(v), p_<(v)\} - 3) + 2n_1 + 2n_2 + n_3 + n_4 \\
\leq \ & \frac{3m}{2} + \frac{n}{4} - 3n + 2n_1 + 2n_2 + n_3 + n_4 \text{ (by Theorem 4.2)} \\
\leq \ & \frac{m}{2} + \frac{1}{2}(n_1 + 2n_2 + 3n_3 + 4n_4 + 5n_5 + 6n_6) - \frac{11}{4}(n_1 + n_2 + n_3 + n_4 + n_5 + n_6) \\
& + 2n_1 + 2n_2 + n_3 + n_4 \\
= \ & \frac{m}{2} + \frac{1}{4}(-n_1 + n_2 - n_3 + n_4 - n_5 + n_6) \\
\leq \ & \frac{m}{2} + \frac{n}{4} \ .
\end{aligned}$$

By Lemma 5.3 the total number of bends is at most $5m/2 + n/4$, and volume is at most $(n + (m/2 + n/4)/3)^3 = (m/6 + 13n/12)^3$. For 6-regular graphs the number of bends is at most $31m/12$ and the volume is at most $(19n/12)^3$.

By Theorem 4.2 and Lemma 5.3, the vertex ordering and the drawing itself can be determined in $O(n)$ time, respectively. □

For graphs with average degree at least five, using the MEDIAN PLACEMENT ORDERING algorithm to determine the diagonal layout produces drawings with fewer bends and less volume than the algorithm based on st-orderings.

5.2.2 Arbitrary General Position Vertex layout

In this section we consider the layout-based approach for minimising the number of bends in 3-D orthogonal point-drawings given a fixed general position layout. Although the methods developed run in exponential time, they have proved to be effective in searching for 2-bend drawings of reasonably small graphs.

Maximum Clique Formulations

We now present a method for searching for solutions to LAYOUT-BASED GENERAL POSITION 3-D POINT-DRAWING using a maximum weight clique formulation. Consider the *edge route* graph R consisting of a vertex for every possible edge route. For each edge $vw \in E(G)$ there are 36 possible edge routes, one for each combination of ports at v and w. Vertices are adjacent in R if and only if their corresponding edge routes can co-exist in the drawing; i.e., vertices of R corresponding to edge routes for the same edge are non-adjacent, and vertices corresponding to edge routes which use the same port are non-adjacent. All other pairs of vertices in R are adjacent. A vertex is in a clique of R if and only if the corresponding edge route is in the drawing. The weight of the vertex corresponding to an edge route vw is $4 - \#\text{bends}\,(vw)$. So a maximum weight clique will define a bend-minimum drawing.

Lemma 5.4. *A general position vertex layout of a graph G has a layout-preserving 3-D orthogonal point-drawing with B bends if and only if the graph R has a clique of weight $4m - B$.* □

In Appendix C we review the existing clique finding algorithms and present a simple algorithm which performs well in comparison to the established methods. The graph R has $36m$ vertices and is quite dense, so even for relatively small graphs G, this method for solving LAYOUT-BASED GENERAL POSITION 3-D POINT-DRAWING is not practical. We shall now introduce a related problem whose maximum clique formulation can be solved for relatively small instances.

Problem 5.3. LAYOUT-BASED 2-BEND 3-D POINT-DRAWING
Instance: A general position vertex layout of a graph G with $\Delta(G) \leq 6$.

Output: A layout-preserving 2-bend 3-D orthogonal point-drawing of a subgraph $G_1 \subseteq G$ with the maximum number of edges.

We conjecture that this problem is also NP-hard. The problem LAYOUT-BASED 2-BEND 3-D POINT-DRAWING suggests an approach for producing 3-D orthogonal point-drawings where we find a partial 2-bend point-routing and then arbitrarily extend it to a point-routing of G. We shall describe two methods for the solution of LAYOUT-BASED 2-BEND 3-D POINT-DRAWING, the first in terms of a maximum clique formulation and the second involving hypergraph matching.

Consider the *arc route* graph R with vertex set $V(R) = A(G) \times \{X, Y, Z\}$. There is an edge in R between 'compatible' arc routes. We define the (complement of the) edge set of R as follows. Since each arc $\overrightarrow{vw} \in A(G)$ can be coloured at most once, for each pair of distinct colours $i, j \in \{X, Y, Z\}$, the edge $\{(\overrightarrow{vw}, i), (\overrightarrow{vw}, j)\} \notin E(R)$. For a 2-bend edge route vw, reversal arcs must be coloured differently, so for each colour $i \in \{X, Y, Z\}$, the edge $\{(\overrightarrow{vw}, i), (\overrightarrow{wv}, i)\} \notin E(R)$. Since different arcs must be assigned different ports, for each vertex $v \in V(G)$, for each pair of arcs $\overrightarrow{vu}, \overrightarrow{vw} \in A_G(v)$ and for each colour $i \in \{X, Y, Z\}$, if $v <_i u, w$ or $u, w <_i v$, then the edge $\{(\overrightarrow{vu}, i), (\overrightarrow{vw}, i)\} \notin E(R)$. All other pairs of vertices of R are adjacent. The next result follows immediately from the definition of R, where including a vertex $(\overrightarrow{vw}, i) \in V(R)$ in a clique of R corresponds to colouring the arc $\overrightarrow{vw}$ with colour i.

Lemma 5.5. *For a fixed general position vertex layout of a maximum degree six graph G, there is a layout-preserving 2-bend 3-D orthogonal point-drawing of a subgraph $G_1 \subseteq G$ if and only if R has a clique of size $2|E(G_1)|$.* □

Given a clique Q of R, to determine a point-routing of $G \setminus G_1$, colour those arcs $\overrightarrow{vw} \in A(G)$ without a corresponding vertex in Q, with whatever spare colour is available, so that there are at most two outgoing arcs at each vertex v receiving the same colour.

Clearly, the arc route graph can be used if a partial routing of the arcs is specified. Moreover, if we relax the general position model so that some vertices share a common coordinate, we can specify a partial routing of the edges by 2-bend non-planar edge routes, and use the arc route graph formulation to search for 2-bend general position point-drawings in the remainder of the graph. This approach was used to find some of the 2-bend point-drawings of the complete multi-partite graphs presented in Appendix B.

Hypergraph Matching Formulation

We now formulate the LAYOUT-BASED 2-BEND 3-D POINT-DRAWING problem as a hypergraph matching problem. Consider the hypergraph P with vertex set

$$V(P) = A(G) \cup \text{ports}(G) \cup (E(G) \times \{X, Y, Z\}) \quad ,$$

and edge set consisting of two edges each of size three, for each edge $\overrightarrow{vw} \in E(G)$ and colour $i \in \{X, Y, Z\}$. If $v <_i w$ then

$$(\overrightarrow{vw}, \text{port}(v, +i), (\{v, w\}, i)),\ (\overrightarrow{wv}, \text{port}(v, -i), (\{v, w\}, i)) \in E(P) \quad ,$$

and if $w <_i v$ then

$$(\overrightarrow{vw}, \text{port}(v, -i), (\{v, w\}, i)),\ (\overrightarrow{wv}, \text{port}(v, +i), (\{v, w\}, i)) \in E(P) \quad .$$

P is 3-uniform and 3-colourable. The vertex corresponding to an arc $\overrightarrow{vw} \in A(G)$ has degree three, the vertex corresponding to a positive (respectively, negative) i-port at a vertex $v \in V(G)$ has degree $s_i(v)$ $(p_i(v))$, and the vertex corresponding to a pair $(\overrightarrow{vw}, i)$ has degree two.

Lemma 5.6. *There is a layout-preserving 2-bend 3-D orthogonal point-drawing of a subgraph $G_1 \subseteq G$ if and only if P has a matching M with $|M| = 2|E(G_1)|$.*

Proof. Including an edge $(\overrightarrow{vw}, \text{port}(v, \pm i), (\{v, w\}, i))$ in a matching M of P corresponds to assigning the arc $\overrightarrow{vw} \in A(G)$ the colour i in a point-routing of G. By construction the arc vw will be assigned the i-port at v pointing towards w when edge routes are determined.

Given a matching M of P, for each arc $\overrightarrow{vw} \in A(G)$ there is at most one edge in M incident to the vertex corresponding to vw, so each arc is coloured at most once. For each $(i^{\pm})$-port at a vertex v there is at most one edge in M incident to the vertex corresponding to $\text{port}(v, \pm i)$, so each port is used at most once. Since the edges $(\overrightarrow{vw}, \text{port}(v, \pm i), (\{v, w\}, i))$ and $(\overrightarrow{wv}, \text{port}(v, \mp i), (\{v, w\}, i))$ have a common vertex, namely $(\{v, w\}, i)$, they cannot both be in M. So reversal arcs are coloured differently, and a 2-bend point-routing of G is determined. By the reverse argument, a layout-preserving 2-bend 3-D orthogonal point-drawing of a subgraph G_1 determines a matching of size $2|E(G_1)|$. □

A matching of the hypergraph P defines a matching in the graph P' formed from P by removing the vertices $(\{v, w\}, i)$ and their incident edges. Hall's marriage theorem [114] thus provides the following necessary condition for the existence of a matching in P, and thus a necessary condition for the LAYOUT-BASED 2-BEND 3-D POINT-DRAWING problem.

At each vertex $v \in V(H)$, for any set $S \subseteq A_G(G_1)\, v$, the number of ports at v which point toward a vertex w for some arc $\overrightarrow{vw} \in S$ is at least $|S|$. (5.4)

This implies that the number of neighbours of a vertex v in a single octant relative to v is at most three, in a single quadrant is at most four, in half-space must be at most five. The following example illustrates why (5.4) is not sufficent for our problem. Consider adjacent vertices v and w, such that $s_Z(v) = 5$, $p_Z(w) = 5$, and $w <_Z v$. Both vw and $\overrightarrow{wv}$ must be coloured Z, as in Figure 5.13.

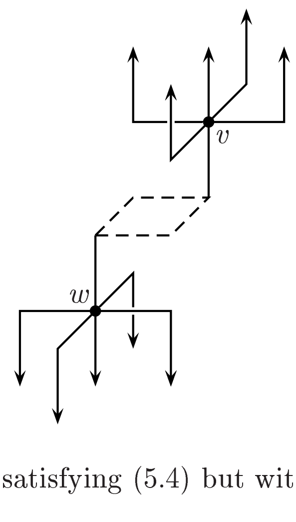


Figure 5.13: A layout satisfying (5.4) but without a 2-bend routing.

The Gallai-Edmonds matching structure theorem (see [148]) provides a mechanism describing all maximum matchings of any (bipartite or non-bipartite) graph. We can use this technique to evaluate all the maximum matchings of P' such that reversal arcs receive different colours, thus providing a method for the solution of LAYOUT-BASED 2-BEND 3-D POINT-DRAWING. Unfortunately there may be an exponential number of such matchings, so this algorithm is not polynomial.

5.3 Routing-Based Algorithm

We now describe a routing-based algorithm for producing general position 3-D orthogonal point-drawings. This method determines a general position vertex layout with respect to a pre-determined point-routing. Our aim is to produce drawings with as many 2-bend edge routes as possible. Hence the routing which is determined is a 2-bend point-routing. Initially we present two algorithms for determining a 2-bend point-routing of a given graph. The routing-based vertex-layout algorithm itself is described in Section 5.3.2.

5.3.1 2-Bend Routing Algorithms

Cycle Cover Decomposition

Our first method for determining a 2-bend point-routing is based on the algorithm for determining a disjoint cycle cover decomposition described in Section 2.5.

Algorithm 5.6. 2-BEND 3-D POINT-ROUTING

Input: graph G with $\Delta(G) \leq 6$

Output: 2-bend 3-D general position point-routing of G.

---

1. Determine a cycle cover decomposition of G with red, green and blue cycle covers.
2. For each edge vw in the red cycle cover, set $\text{col}(\overrightarrow{vw}) \leftarrow X$ and $\text{col}(\overrightarrow{wv}) \leftarrow Y$.
3. For each edge vw in the green cycle cover, set $\text{col}(\overrightarrow{vw}) \leftarrow Y$ and $\text{col}(\overrightarrow{wv}) \leftarrow Z$.
4. For each edge vw in the blue cycle cover, set $\text{col}(\overrightarrow{vw}) \leftarrow Z$ and $\text{col}(\overrightarrow{wv}) \leftarrow X$.

Lemma 5.7. *The algorithm* 2-BEND 3-D POINT-ROUTING *determines a 2-bend point-routing in $O(n)$ time.*

Proof. There are at most two arcs at each vertex v coloured $i \in \{X, Y, Z\}$ and reversal arcs are coloured differently, so the colouring is a 2-bend point-routing. By Theorem 2.1, the cycle cover decomposition and hence the 2-bend point-routing can be found in $O(n)$ time. □

Systems of Transitions

We now describe a second method for determining a 2-bend point-routing based on systems of transitions. Suppose G is an Eulerian graph. (A non-Eulerian graph of maximum degree six can be augmented to a 6-regular graph, as in Theorem 2.1.) A *transition* at a vertex v is a pair of distinct edges incident with v. A *system of transitions* at v is a partition of $\{vw \in E(G)\}$ into transitions at v. A *system of transitions* of G is a family $T_G = \{T_v : v \in V\}$ where T_v is a system of transitions at v [98, 121].

A k-colouring of the transitions in T_G such that transitions at a common vertex and transitions with a common edge receive different colours determines a k-colouring of $A(G)$ such that reversal arcs are coloured differently and $\Delta(\overleftrightarrow{G}[i]) = 2$ for each colour i; i.e., a point-routing. We therefore vertex-colour the graph $T(G)$ whose vertex set consists of all transitions in T_G, with vertices of $T(G)$ being adjacent if their corresponding transitions in G are (1) at a common vertex of G, or (2) contain a common edge of G.

These two types of edges decompose the graph $T(G)$ into (1) a collection of vertex-disjoint cliques $\{C_v : v \in V(G)\}$ where $|C_v| = \deg_G(v)/2$, and (2) a 2-regular spanning subgraph. If the system of transitions is determined by following an Eulerian tour of G, this 2-regular spanning subgraph is, in fact, a Hamiltonian cycle.

Hence, for a 6-regular graph G, if we determine the system of transitions by following an Eulerian tour of G, the graph $T(G)$ has an edge-decomposition into a Hamiltonian cycle and a set of edge-disjoint triangles. Each triangle represents a vertex of G and the edges around the Hamiltonian cycle correspond to the Eulerian tour of G.

That a 4-regular graph with such a 'cycle plus triangles' decomposition is vertex 3-colourable was conjectured by Erdős and first proved by Fleischner and Stiebitz [99] using a non-constructive and non-elementary colouring result of Alon and Tarsi [3]. Sachs [189] has since developed a constructive and elementary proof. So $T(G)$ is vertex 3-colourable, thus determining a 2-bend point-routing of G.

5.3.2 Determining a Layout

For a fixed routing of a graph G, in a general position 3-D orthogonal point-drawing with the minimum number of bends, each i-ordering, $i \in \{X, Y, Z\}$, is an optimal solution to the balanced ordering problem on the subgraph $\overleftrightarrow{G}[i]$. In the following algorithm, to determine each i-ordering, we use the local minimum approach for the balanced ordering problem developed in Chapter 4.

Algorithm 5.7. ROUTING-BASED GENERAL POSITION 3-D POINT-DRAWING

Input: graph G with $\Delta(G) \leq 6$

Output: general position 3-D orthogonal point-drawing of G.

---

1. Determine a 2-bend point-routing of G using Algorithm 5.6 2-BEND 3-D POINT-ROUTING.

2. For each $i \in \{X, Y, Z\}$, set the i-ordering to be a 2-balanced ordering of $\overleftrightarrow{G}[i]$ (see Theorem 4.5).

3. Apply Algorithm 5.1 GENERAL POSITION 3-D POINT-DRAWING.

Theorem 5.3. *The algorithm* ROUTING-BASED GENERAL POSITION 3-D POINT-DRAWING *determines, in* $O(n^2)$ *time, a 4-bend 3-D orthogonal point-drawing of* G *with at most* $2m + 3n/2$ *bends and at most* $(3n/2)^3$ *bounding box volume.*

Proof. In a 2-bend point-routing, reversal arcs are coloured differently, so $\overleftrightarrow{G}[i]$ has no 2-cycles, for each colour $i \in \{X, Y, Z\}$. $\overleftrightarrow{G}[i]$ has maximum outdegree two, so by Theorem 4.5, a 2-balanced vertex ordering of $\overleftrightarrow{G}[i]$ has total cost at most n. Applying Theorem 5.1, since reversal arcs are coloured differently, there will be no instances of (5.2), and in each i-ordering there will be at most $n/2$ instances of (5.1). Hence there will be at most $n/2$ anchored arcs coloured i, for each $i \in \{X, Y, Z\}$. In total there will be at most $3n/2$ anchored arcs, so the total number of bends is at most $2m + 3n/2$, and the bounding box volume is at most $(n + n/2)^3 = (3n/2)^3$. By Theorem 2.1 calculating the cycle covers and by Theorem 4.5 each vertex ordering takes $O(n)$ time. The final step of the algorithm, which by Theorem 5.1 takes $O(n^2)$ time, is the most time-consuming. So the overall algorithm takes $O(n^2)$ time. □

5.4 Diagonal Layout and Movement Algorithm

In this section we describe an algorithm for 3-D orthogonal point-drawing which, in some sense, combines the layout- and routing-based approaches. Initially the vertices are placed along the main diagonal of a cube, and a point-routing is determined. This routing also defines the movement of vertices away from the diagonal. This algorithm establishes the best known upper bound for the total number of bends in 3-D orthogonal point-drawings.

Algorithm 5.8. DIAGONAL LAYOUT AND MOVEMENT

Input: graph G with $\Delta(G) \leq 6$.

Output: general position 3-D orthogonal point-drawing of G.

---

1. Determine a vertex ordering $<$ of $V(G)$ using Algorithm 4.4 ALMOST 2-BALANCED VERTEX ORDERING. Call a vertex v *balanced* if $\max\{s(v), p(v)\} \leq 3$, and *unbalanced* otherwise.

2. Initialise the X-, Y- and Z-orderings of a general position vertex layout to be the vertex ordering $<$.

3. For each unbalanced vertex $v \in V(G)$, depending on the number of predecessors and successors of v in the vertex ordering $<$ (see Section 4.1), label arcs $\overrightarrow{vw} \in A(G)$ as *movement* or *special* arcs, according to Table 5.2.

4. Determine a point-routing of G with Algorithm 5.9 DLM — DETERMINE POINT-ROUTING, described in Section 5.4.2.

Table 5.2: Definition of movement and special arcs at an unbalanced vertex v.

v	(0,4)	(1,4)	(0,5)	(2,4)	(1,5)	(0,6)
vv^1	movement	movement	movement	special	movement	movement
vv^2	-	-	movement	-	special	movement
vv^3	-	-	-	-	-	special

5. For each movement arc vw coloured $i \in \{X, Y, Z\}$, move v to immediately past w in the i-ordering.

6. Apply Algorithm 5.1 General Position 3-D Point-Drawing

5.4.1 Movement of Vertices

The general stratgey of the Diagonal Layout and Movement algorithm is to anchor at most one arc $\overrightarrow{vw}$ at each vertex v. The port at a vertex v assigned to an unanchored arc $\overrightarrow{vw}$ must point toward w. 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. So, at a balanced vertex v (i.e., $\max\{s(v), p(v)\} \leq 3$), all of the arcs $\overrightarrow{vw}$ need not be anchored.

If $s(v) > 3$ (respectively, $p(v) > 3$) the positive (negative) ports can be assigned to at most three successor (predecessor) arcs of v. The remaining successor (predecessor) arcs $\overrightarrow{vw}$ must be assigned a negative (positive) port at v. These are precisely the *movement* and *special* arcs defined in Table 5.2. Note that there is one special arc $\overrightarrow{vw}$ at each unbalanced degree six vertex v. We shall prove that special arcs will become anchored when algorithm General Position 3-D Point-Drawing is applied.

If vw is a movement arc coloured i, then v is moved to immediately past w in the i-ordering (Step 5 of the algorithm), thus allowing vw to be assigned the port$(v, -i)$ for positive v and the port$(v, +i)$ for negative v. In Figure 5.14 we illustrate the movement and anchoring process in the case of a positive (0,6)-vertex.

For a vertex v with $\max\{s(v), p(v)\} > 3$, if $vw = vv^k$ is a movement or special

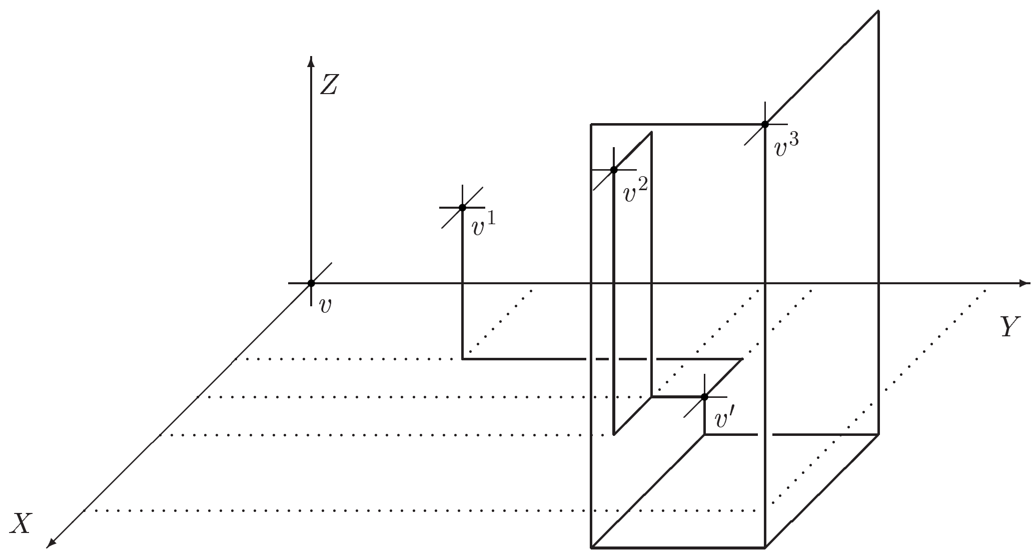


Figure 5.14: v is a positive (0,6)-vertex, vv^1 is a movement arc coloured X, vv^2 is a movement arc coloured Y, vv^3 is an special arc coloured Z; move v to v'.

arc then $k \leq \lfloor c_v/2 \rfloor$, so rule M1 is applicable. Therefore w cannot be opposite to v, and hence $\overrightarrow{wv}$ cannot also be a movement or special arc. (Consequently when edges are routed no 4-bend edge routes are constructed immediately. It is only through swapping ports to remove crossings that a 4-bend edge route can be introduced.) Furthermore, if vv^k is a movement arc then $k \leq \lfloor (c_v - 1)/2 \rfloor$, so by rules M2 and M3, if v and w are opposite unbalanced vertices then the movement arcs of v do not 'cross over' or have the same destination vertex as the movement arcs of w.

5.4.2 Determining a Point-Routing

To determine a point-routing we construct a graph H with vertex set $V(H) = A(G)$. Vertices are adjacent in H if the corresponding arcs must use perpendicular ports. A 3-vertex-colouring of H then determines a point-routing of $A(G)$.

Algorithm 5.9. DLM — DETERMINE POINT-ROUTING

Input: • graph G with $\Delta(G) \leq 6$.

• vertex ordering of G determined in Step 1 of Algorithm DIAGONAL LAYOUT AND MOVEMENT.

• classification of movement and special arcs from Step 3 of Algorithm Diagonal Layout and Movement.

Output: point-routing of $A(G)$.

---

1. Construct a graph H with vertex set $V(H) = A(G)$. We distinguish four types of edges of H as follows.

   (a) The first type of edge ensures that arcs which 'compete' for the same ports are coloured differently. In Table 5.3 the arcs vv^A, vv^B, vv^C, vv^D, vv^E and vv^F are defined for each type of vertex. (If v is a balanced or a positive (respectively, negative) unbalanced vertex then vv^A, vv^B and vv^C will be assigned the negative (positive) ports at v. The arcs vv^D, vv^E and vv^F will be assigned the positive (negative) ports at v.) For each vertex $v \in V(G)$, add a triangle $\{vv^A, vv^B, vv^C\}$ and $\{vv^D, vv^E, vv^F\}$ to $E(H)$.

Table 5.3: Definition of vv^A, vv^B, vv^C, vv^D, vv^E and vv^F

v	vv^A	vv^B	vv^C	vv^D	vv^E	vv^F
balanced	vv^{-3}	vv^{-2}	vv^{-1}	vv^1	vv^2	vv^3
(0,4)-vertex	vv^1	-	-	vv^2	vv^3	vv^4
(1,4)-vertex	vv^{-1}	vv^1	-	vv^2	vv^3	vv^4
(2,4)-vertex	vv^{-2}	vv^{-1}	vv^1	vv^2	vv^3	vv^4
(0,5)-vertex	vv^1	vv^2	-	vv^3	vv^4	vv^5
(1,5)-vertex	vv^{-1}	vv^1	vv^2	vv^3	vv^4	vv^5
(0,6)-vertex	vv^1	vv^2	vv^3	vv^4	vv^5	vv^6

   (b) If neither the arc $\overrightarrow{vw}$ not its reversal arc $\overrightarrow{wv}$ are special then add the edge $\{\overrightarrow{vw}, \overrightarrow{wv}\}$ (labelled 'r') to $E(H)$.

   (c) If $\overrightarrow{vw}$ and $\overrightarrow{wx}$ are both movement arcs for some vertices v, w and x, then add the edge $\{\overrightarrow{vw}, \overrightarrow{wx}\}$ (labelled '$*$') to $E(H)$. (This ensures that v and w do not move in the same ordering.)

   (d) If vv^2 is a movement arc coloured i then v will move past v^1 in the i-ordering. To ensure that v^1v does not use the incorrect i-port at v^1, add the edge $\{vv^2, v^1v\}$ (labelled '$**$') to $E(H)$. (Observe that in Figure 5.14, v^1v cannot use the port (v_1, Y^+).)

2. Repeatedly remove vertices of H with degree at most two, and merge non-adjacent vertices $v, w \in V(H)$ in a $K_4 \setminus vw$ subgraph (and replace any parallel edges by a single edge).

3. Determine a proper vertex-colouring of H with three colours.

4. Colour the removed vertices $v \in V(H)$ in reverse order of their removal, with a colour different from the (≤ 2) neighbours of v.

5. Determine a 3-colouring of $A(G)$ from the colouring of $V(H)$.

Lemma 5.8. *The graph H is vertex 3-colourable in $O(n)$ time.*

Proof. If $K_4 \setminus vw$ is a subgraph of H for some non-adjacent vertices v and w, then in any proper 3-colouring of $V(H)$, v and w must receive the same colour, so 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 $K_4 \setminus vw$ subgraph, H has maximum degree three, and is not K_4, so by Brooks' Theorem [47], is 3-colourable.

For an unbalanced vertex v, let H_v be the subgraph of H consisting of the vertices vv^A, vv^B and vv^C and their incident edges. We shall initially show that H_v 'reduces' to a maximum degree three subgraph.

For a degree six unbalanced vertex v, the vertex of H corresponding to the special arc vv^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 vv^A and vv^B as movement arcs, H_v is the same for a (0,6)-vertex v (after removing vv^C) and for a (0,5)-vertex v (see Figures 5.15 and 5.16). Similarly, for (1,5)- and (2,4)-vertices, H_v is the same as for (1,4)- and (2,3)- vertices respectively. We therefore need only consider (0,5)-, (1,4)- or (0,4)- unbalanced vertices. Thus the result for graphs with unbalanced degree six vertices in the vertex ordering reduces to the result for vertex orderings without such vertices.

Consider a (0,5)-vertex v. v^1 may be balanced or a (1,4)-vertex. If v^1 is balanced then, as in Figure 5.15, vv^1 has degree two and can be removed. In the remaining graph, vv^2 and v^1v have degree three.

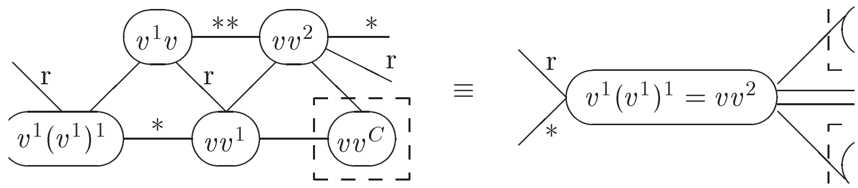


Figure 5.15: The subgraph H_v for a (0,5)-vertex or a (0,6)-vertex v with v^1 balanced.

Now, if v^1 is a (1,4)-vertex then, as in Figure 5.16, vv^2 and $v^1(v^1)^1$ are the non-adjacent vertices in a $K_4 \setminus \{e\}$ subgraph. If we merge these vertices then v^1v and vv^1 have degree two and can be removed. If v^2 is balanced then there is no edge $\{vv^2, v^2(v^2)^1\}$. If v^2 is unbalanced then v^2 must be a (1,4)-vertex, and therefore v^2v and the edge $\{vv^2, v^2v\}$ (labelled 'r') will be removed (see Figure 5.17). In either case vv^2 $(=v^1(v^1)^1)$ has degree three.

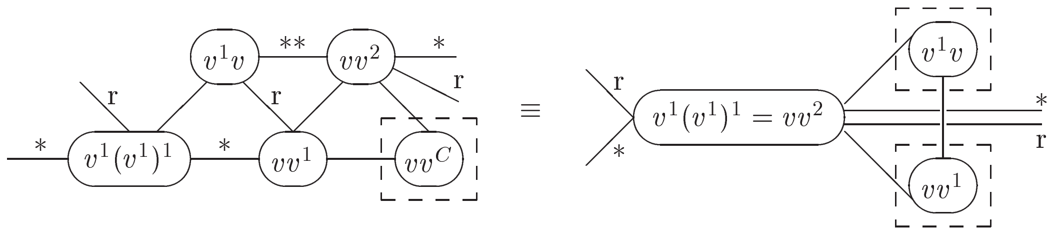


Figure 5.16: The subgraph H_v for a (0,5)-vertex or a (0,6)-vertex v with v^1 a (1,4)-vertex.

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 in Figure 5.17, the vertex vv^{-1} has degree two and can be removed. vv^1 now has degree at most three. For a (0,4)-vertex v, H_v simply consists of the degree one vertex vv^1, which can be removed.

Consider a vertex $vv^j \in V(H)$ for some $j \in \{D, E, F\}$, or $j \in \{A, B, C\}$ if v is balanced. vv^j is incident with at most two unlabelled edges and to at most one edge labelled 'r'. Unless v^j is a (0,5)- or (0,6)-vertex and $(v^j)^1 = v$ (in which case vv^j is incident with an edge labelled '$**$' and has already been considered), vv^j has degree at

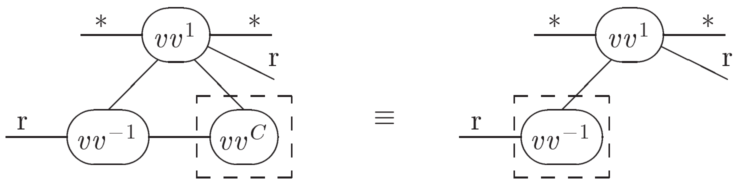


Figure 5.17: The subgraph H_v of H for a (1,4)-vertex or a (1,5)-vertex v.

most three.

We have shown that all remaining vertices in H have degree at most three, and it is easily seen that H is not K_4, so by Brooks' Theorem [47], H is 3-colourable. The proof of Brook's Theorem due to Lovász [147] and simplified by Bryant [49] describes an algorithm for finding a vertex 3-colouring of H in $O(|E(H)|) = O(n)$ time. □

The 3-vertex-colouring of H determines a 3-colouring of $A(G)$. The unlabelled edges in H ensure that at most two outgoing arcs at each vertex v receive the same colour. So the 3-colouring of H determines a point-routing of G (Step 4 of Algorithm Diagonal Layout and Movement), and hence Algorithm General Position 3-D Point-Drawing is applicable (Step 6 of Algorithm Diagonal Layout and Movement).

Theorem 5.4. *For a given graph G with maximum degree six, the* Diagonal Layout and Movement *algorithm will, in $O(n^2)$ time, determine a 4-bend 3-D orthogonal point-drawing of G with bounding box volume* $(4n/3)^3 = 2.37n^3$ *and at most* $7m/3$ *bends. If G has maximum degree five then the bounding box has volume n^3 and each edge route has two bends.*

Proof. We now calculate the number of bends and the volume of the drawing which will result when we apply algorithm General Position 3-D Point-Drawing. To do so, we count the number of instances of (5.1). Suppose the arcs $\overrightarrow{vu}, \overrightarrow{vw} \in A_G(v)[i]$ $(u \neq w)$ for some vertex v and colour $i \in \{X, Y, Z\}$. We can assume that $\overrightarrow{vu} \in \{vv^A, vv^B, vv^C\}$ and $\overrightarrow{vw} \in \{vv^D, vv^E, vv^F\}$.

Suppose $\overrightarrow{vu}$ is a movement arc. Then u is not between v and w in the initial ordering. v moves past u in the i-ordering, and since the movement arcs originating at w (if any) do not cross over u, w cannot move past u in any ordering. Therefore v is between u and w in the final i-ordering.

Suppose $\overrightarrow{vu}$ is neither a movement arc nor a special arc. Then v is between u and w in the initial ordering, and v does not move past u or w in any ordering. If u moves past v then it does so in the same ordering as the colour assigned to the movement arc $\overrightarrow{uv}$. Since $\{\overrightarrow{vu}, \overrightarrow{uv}\} \in E(H)$ in this case, $\overrightarrow{uv}$ is not coloured i, so u does not move in the i-ordering. Similarly w does not move in the i-ordering, and hence, v is between u and w in the i-ordering.

So, the only case where v is not between u and w in the i-ordering is if $\overrightarrow{vu}$ or $\overrightarrow{vw}$ is special. Since every vertex is incident to at most one special arc, every instance of (5.1) corresponds to a unique special arc. Hence there are at most k instances of (5.1) where k is the number of special arcs, which is precisely the number of unbalanced degree six vertices.

Now suppose there is an instance of (5.2); i.e., there is a pair of reversal arcs $\overrightarrow{vw}, \overrightarrow{wv} \in A(G)$ receiving the same colour i, $\overrightarrow{vu} \in A_G(v)[i]$ $(w \neq u)$, $\overrightarrow{wx} \in A_G(w)[i]$ $(v \neq x)$, v is between u and w in the i-ordering, and w is between v and x in the i-ordering. The 'r' edges in H ensure that one of $\overrightarrow{vw}$ and $\overrightarrow{wv}$, say $\overrightarrow{vw}$, must be special. However, in this case v will not be between u and w in the i-ordering. So there are no instances of (5.2).

If k is the number of special arcs then Theorem 5.1 asserts G has a 4-bend 3-D orthogonal point-drawing with bounding box volume $(n + k/3)^3$ and $2m + k$ bends. Since $k \leq n$ the bounding box volume is at most $(n + n/3)^3 = (4n/3)^3$. If d is the average degree of those vertices without special arcs then $6k + d(n - k) = 2m$ and the number of bends is $2m + k = 2m + (2m - d(n - k))/6 = 7m/3 - d(n - k)/6$. Since $n \geq k$ the drawing has at most $7m/3$ total bends.

For maximum degree five graphs, no special arcs are introduced by the algorithm and reversal arcs are coloured differently, so the point-routing is a 2-bend point-routing. By the same argument as above, if $\overrightarrow{vu}, \overrightarrow{vw} \in A_G(v)[i]$ $(u \neq w)$ then v is between u and w in the i-ordering. Hence, the conditions (5.1) and (5.2) do not occur. So there are no anchored arcs in the point-drawing produced. With no anchored edge routes, no new anchors can be introduced by the edge crossing removal stage. So the crossing-free drawing has two bends per edge route and bounding box volume n^3.

The 3-colouring of H takes $O(|E(H)|)$=$O(n)$ time, and by Theorem 5.1, algorithm

General Position 3-D Point-Drawing takes $O(n^2)$ time, so the algorithm Diagonal Layout and Movement takes $O(n^2)$ time. □

Corollary 5.4. *The algorithm* Diagonal Layout and Movement *is a 7/6-approximation algorithm for the* BEND-MINIMUM GENERAL POSITION 3-D POINT-DRAWING *problem.*

Proof. Since every general position 3-D orthogonal point-drawing has at least $2m$ bends, and the Diagonal Layout and Movement algorithm determines a general position 3-D orthogonal point-drawing with at most $7m/3$ bends, the approximation factor is at most $(7m/3)/(2m) = 7/6$. □

5.5 3-Bend Algorithms

We now consider the problem of minimising the maximum number of bends on any edge route in 3-D orthogonal point-drawings. As discussed in Section 3.5.1, K_5 provides a lower bound of two for the maximum number of bends per edge route in 3-D orthogonal point-drawings. Eades *et al.* [86, 87] first established that every maximum degree six graph has an orthogonal point-drawing with a maximum of three bends per edge route. Their 3-Bends algorithm is based on an arbitrary diagonal layout of the vertices, and a cycle cover decomposition of the edges. As stated in their paper the drawings produced have $27n^3$ volume; by simply deleting grid-planes not containing a vertex or a bend the volume is easily seen to be at most $8n^3$.

The Incremental algorithm of Papakostas and Tollis [166, 168], using an ad-hoc vertex layout and edge routing strategy, also produces orthogonal point-drawings with at most three bends per edge. The volume of the drawings produced is at most $4.63n^3$. This algorithm has the advantage of supporting the on-line insertion of vertices in constant time.

In this section we describe an algorithm, which given an arbitrary 3-D general position vertex layout of graph, determines a 3-bend layout-preserving orthogonal point-drawing. We then present an algorithm, which is a modification of the 3-Bends algorithm of Eades *et al.* [86, 87], for producing 3-D orthogonal point-drawings with

$n^3 + O(n^{5/2})$ volume and at most three bends per edge. This is the best known upper bound for the volume of 3-bend 3-D orthogonal point-drawings.

5.5.1 Edge Routes

In this section we employ a modified version of Algorithm General Position 3-D Point-Drawing as the basis for our main algorithms. Given a maximum degree six graph G, a general position vertex layout and a point-routing of G we position the vertices as in Algorithm General Position 3-D Point-Drawing, however our algorithms directly specify the port assignment. We again employ Algorithm Construct Edge Routes, although we only use 2-bend edge routes (see Figure 5.2) and 3-bend edge routes with parallel ports (see Figures 5.3(b) and 5.4). Furthermore, 3-bend edge routes using ports pointing in the same direction are constructed somewhat differently, as we now describe.

The minimal box containing all vertices is called the *inner box*. For each direction $d \in \{X^{\pm}, Y^{\pm}, Z^{\pm}\}$, the box extending out from the d-face of the inner box is called the *d-outer box*, as shown in Figure 5.18.

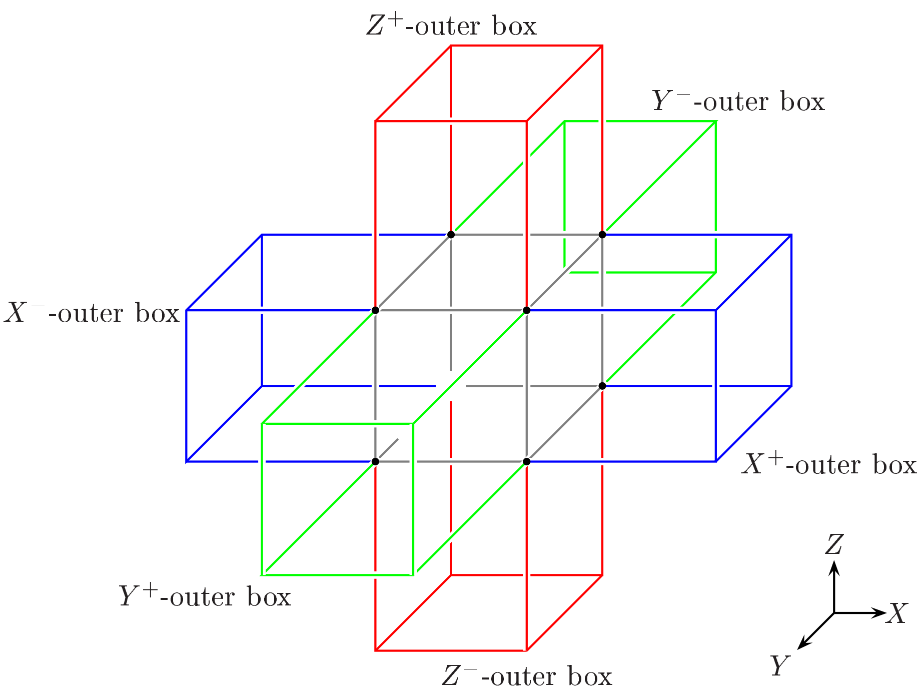


Figure 5.18: Inner and Outer Boxes.

2-bend edge routes and 3-bend edge routes vw using opposite ports at v and w are routed entirely within the inner box exactly as was the case previously. We call these edge routes *inner*. If, for some direction d, an edge is assigned d-ports at both end-vertices, instead of the edge route shown in Figure 5.3(b), we use the edge route shown in Figure 5.19, which is routed to a *height* $h(vw)$ in the d-outer box. The algorithms to follow specify the value of $h(vw)$.

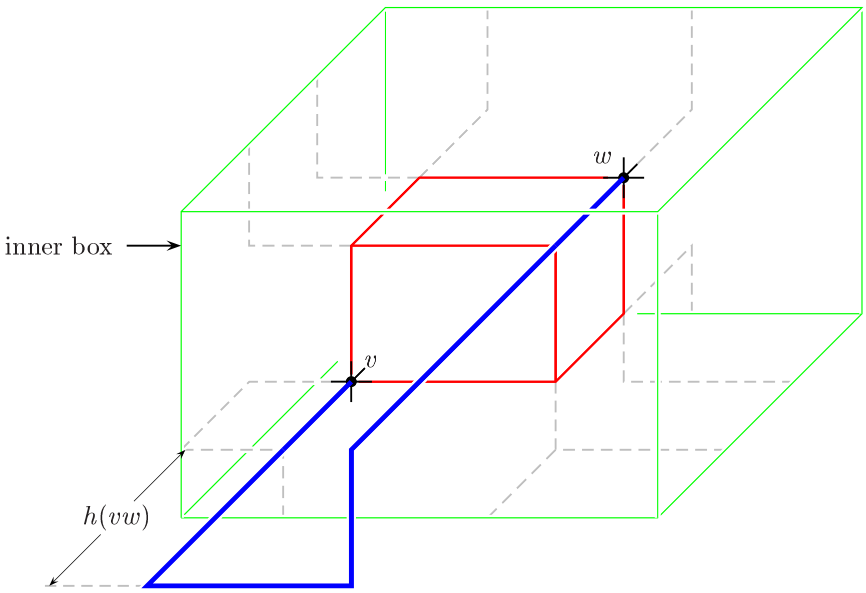


Figure 5.19: Outer 3-bend edge route.

This approach has the advantage that some edges routed in a particular outer box can have the same height, thus reducing the volume. Also, given a drawing only using the above-mentioned edge routes, we shall prove that the Algorithm 5.4 Point-Drawing Remove Edge Crossings will not introduce any 4-bend edge routes. A disadvantage of this approach is that the edge routes are longer.

5.5.2 Arbitrary Layout 3-Bend Algorithm

The following algorithm for producing 3-bend 3-D orthogonal point-drawings which preserve a given general position vertex layout, is based on a cycle cover decomposition of the graph. Edges in the cycle cover C_i, $i \in \{X, Y, Z\}$, are routed using i-ports at both end-vertices. All edges are outer 3-bend edge routes except in the case of an odd cycle where one edge of the cycle is an inner 3-bend edge route. Edges in a particular outer box are routed with unique height.

Algorithm 5.10. General Position 3-Bend 3-D Point-Drawing

Input: • multigraph G with $\Delta(G) \leq 6$
• general position 3-D vertex layout of $V(G)$

Output: layout-preserving 3-bend 3-D orthogonal point-drawing of G

---

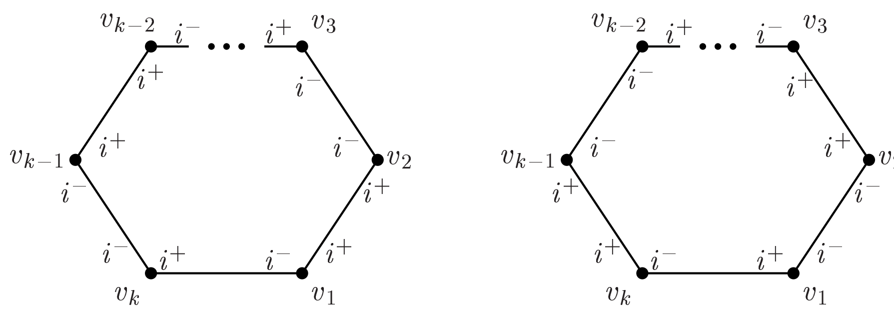
1. Suppose the X-, Y- and Z-vertex orderings are $(x_1, x_2, \ldots, x_n)$, $(y_1, y_2, \ldots, y_n)$ and $(z_1, z_2, \ldots, z_n)$, respectively.

2. For each vertex $v \in V(G)$, if $v = x_i = y_j = z_k$ then position v at $(3i, 3j, 3k)$.

3. Determine a cycle cover decomposition C_X, C_Y, C_Z of G (see Section 2.5).

4. For each $i \in \{X, Y, Z\}$, and for each cycle $C = (v_1, v_2, \ldots, v_k)$ of C_i:

   - If k is even, then traverse the cycle and assign to each edge alternately the i^+/i^- ports at both end-vertices.
   - If k is odd, then assign to the edge $v_k v_1$ the i-ports at v_k and v_1 which point toward each other. Traverse the remainder of the cycle and assign to each edge alternately i^+/i^- ports at both end-vertices, as shown in Figure 5.20.



(a) $v_k <_i v_1$ (b) $v_1 <_i v_k$

Figure 5.20: Port assignment for an odd cycle in C_i.

5. For each $d \in \{X^\pm, Y^\pm, Z^\pm\}$, for each edge vw assigned d-ports at v and w, assign to vw a unique *height* $h(vw) \geq 1$.

6. For each edge $vw \in E(G)$ assigned d-ports, for some direction d, at both v and w, route vw with the 3-bend edge route shown in Figure 5.19 in the d-outer box. Route edges assigned opposite ports as in Figure 5.4.

7. Apply Algorithm 5.4 Point-Drawing Remove Edge Crossings.

8. Remove each grid-plane not containing a vertex or a bend.

Theorem 5.5. *The algorithm* General Position 3-Bend 3-D Point-Drawing *determines, in $O(n^2)$ time, a layout-preserving 3-D orthogonal point-drawing of G with $8n^3$ bounding box volume and three bends per edge route.*

Proof. By construction, each edge is assigned unique ports at its end-vertices, and only 3-bend edge routes are used. We now prove that given a general position 3-D orthogonal point-drawing only using 2-bend edge routes and 3-bend edge routes with parallel ports (routed as described above), the algorithm Point-Drawing Remove Edge Crossings will not introduce a 4-bend edge route.

For the edge route shown in Figure 5.19, both of the segments in the outer box are called *middle* segments. The segment of such an edge route incident to the end-vertex v is called a *v-segment*.

Since middle segments on outer edge routes have unique height, they cannot intersect. A v-segment parallel to the i-axis has an i-coordinate belonging to v and no other vertex, so v-segments can only intersect as in Case 1 of Algorithm 5.4 Point-Drawing Remove Edge Crossings. Swapping ports, in this case, does not introduce any new edge route crossings, so cannot introduce a 4-bend edge route. Therefore the only possible intersection is between the middle segments of 2-bend edge routes (Case 3 of Algorithm 5.4 Point-Drawing Remove Edge Crossings). Swapping ports removes the crossing, and both edge routes remain two bend edge routes.

The inner box is initially $3n \times 3n \times 3n$. Every edge in cycle cover C_i either adds one i-plane in the outer box or occupies one of the i-planes belonging to one of its end-vertices. Since there are at most $m/3$ edges in each cycle cover, after removing grid-planes not containing a vertex or a bend, the bounding box volume is at most

$(n + m/3)^3 \leq 8n^3$. The most time-consuming step of the algorithm is the removal of edge crossings which takes $O(n^2)$ time. □

We now describe a heuristic for determining sets of edge routes in the same outer box which can be routed with the same height, thus reducing the volume of the drawing. Construct a graph H with vertex set corresponding to the edges of G routed in a particular outer box, with edges between vertices of H corresponding to edge routes which will intersect if routed with same height. Then if we determine the heights of the edge routes from a vertex-colouring of H, then we obtain an intersection-free drawing. In general, this method does not provide improved worst case volume bounds. In the next section we describe an algorithm which does provide improved volume bounds, by allowing certain edges routed in a particular outer box to have the same height.

5.5.3 Diagonal Layout 3-Bend Algorithm

We now describe a modification to the 3-BENDS algorithm of Eades *et al.* [86, 87], which provides the best known upper bound for the volume of 3-bend 3-D orthogonal point-drawings.

Algorithm 5.11. DIAGONAL GENERAL POSITION 3-BEND POINT-DRAWING

Input: multigraph G with $\Delta(G) \leq 6$

Output: 3-bend 3-D orthogonal point-drawing of G

---

1. Determine a book-embedding of G using the algorithm of Malitz [151] (See Section 1.3). Suppose $(v_1, v_2, \ldots, v_n)$ is the spine ordering, and $p : E(G) \to \{1, 2, \ldots, P\}$ is the page numbering where $P = O(\sqrt{n})$.

2. Apply the 3-BENDS algorithm of Eades *et al.* [86, 87] using $(v_1, v_2, \ldots, v_n)$ as the ordering of the vertices along the diagonal, and route each 3-bend edge route vw as shown in Figure 5.19 with $h(vw) = p(vw)$.

3. Remove each grid-plane not containing a vertex or a bend.

Theorem 5.6. *The algorithm* DIAGONAL GENERAL POSITION 3-BEND POINT-DRAWING *determines a 3-D orthogonal point-drawing of G with $n^3 + O(n^{5/2})$ bounding box volume and three bends per edge route.*

Proof. Note that the only types of edge routes used in the 3-BENDS algorithm of Eades *et al.* [86, 87] are 2-bend edge routes and 3-bend edge routes with both ports pointing in the same direction. So, by the proof of Theorem 5.5, edge routes can only intersect if they are routed with the same height in the same outer box; i.e., they are in the same page of the book embedding. However, if edges routed at the same height intersect in the outer box, then they would also intersect in the book embedding (see Figure 5.21). Hence there are no edge route crossings.

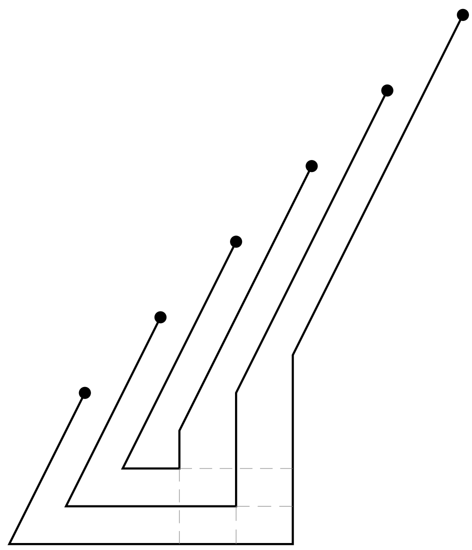


Figure 5.21: Edges in the same page and routed in the same outer box.

The bounding box is $(P + n + P) \times (P + n + P) \times (P + n + P)$. By Malitz [151], $P = O(\sqrt{m}) = O(\sqrt{n})$, so the volume is $(n + O(\sqrt{n}))^3 = n^3 + O(n^{5/2})$. $\square$

5.6 Lower Bounds

Since every edge route in a general position 3-D orthogonal drawing has at least two bends, there is an obvious lower bound of $2m$ for the BEND-MINIMUM GENERAL POSITION 3-D POINT-DRAWING problem. We now present infinite families of graphs which require more than two bends per edge in any general position 3-D orthogonal point-drawing. Our results are based on the observation that if an edge is routed using the X^+ port at the vertex x_n, then this edge route must be anchored, and similarly for other 'extreme' ports, as in Figure 5.22.

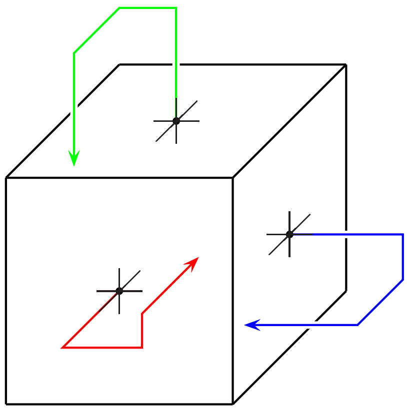


Figure 5.22: Edge routes using 'extreme' ports are necessarily anchored.

For 6-regular graphs all ports must be used, so such a graph requires at least $2m+6$ bends in a general position 3-D orthogonal point-drawing. Hence the graph consisting of some number of disjoint copies of K_7 provides the following lower bound. Note that general position 3-D orthogonal point-drawings of K_7 with $2m+6$ bends do exist.

Lemma 5.9. *There exists an infinite family of graphs, each with at least $2m+6n/7$ bends in any general position 3-D orthogonal point-drawing.* □

Note that this lower bound differs from our upper bound of $7m/3$ (see Theorem 5.4) by only $n/7$. For biconnected graphs we have the following lower bound[1].

Lemma 5.10. *There exists an infinite family of biconnected graphs, each with at least $2m+4n/7$ bends in any general position 3-D orthogonal point-drawing.*

Proof. Consider the 6-regular graph G_a ($a \geq 2$) formed from a copies of $K_7 \setminus e$ (for some edge e) with a cycle added between the copies, as illustrated in Figure 5.23.

Clearly G_a is biconnected. Removing an edge from K_7 can save at most two anchored arcs, so a general position 3-D orthogonal point-drawing of $K_7 \setminus e$ has at least

[1]This result was discovered in conjunction with Therese Biedl.

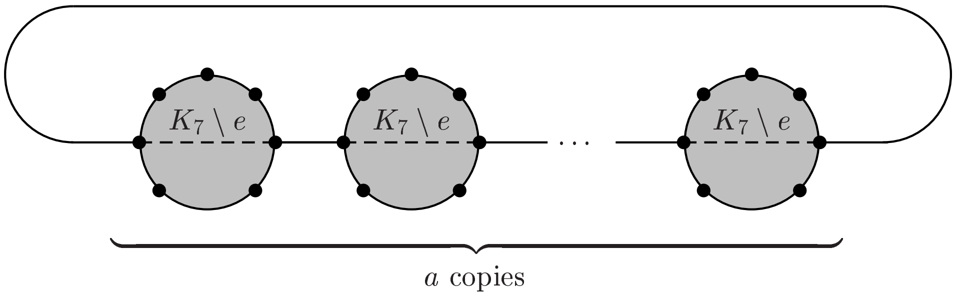


Figure 5.23: The graph G_a.

$2|E(K_7 \setminus e)| + 4$ bends. The 'cycle' edges of G_a each have at least two bends, so G_a has at least $2m + 4n/7$ bends. □

Lemma 5.11. *There exists an infinite family of 4-connected graphs, each with at least $2m + 2n/7$ bends in any general position 3-D orthogonal point-drawing.*

Proof. Consider the 6-regular graph $G_{a,b}$ $(a, b \geq 2)$ formed from the $a \times b$ 4-regular 'torus grid' graph replacing each vertex by $K_7 \setminus \{e_1, e_2\}$ (for some non-incident edges e_1, e_2), as shown in Figure 5.24.

Removing any three vertices from $G_{a,b}$ cannot disconnect the graph, but removing four vertices can, so $G_{a,b}$ is 4-connected. Removing two edges from K_7 can save at most four anchored arcs, so a general position 3-D orthogonal point-drawing of $K_7 \setminus \{e_1, e_2\}$ has at least $2|E(K_7 \setminus \{e_1, e_2\})| + 2$ bends. Edges not in a $K_7 \setminus \{e_1, e_2\}$ have at least two bends, so $G_{a,b}$ has at least $2m + 2n/7$ bends. □

This sequence of lower bounds suggests the following open problem.

Open Problem 5.1. Does every 6-connected 6-regular graph have a general position 3-D orthogonal point-drawing with at most $2m + 6$ bends?

5.6.1 2-Bends Problem

We now look at the ramifications of the above lower bounds for the 2-bends problem discussed in Section 3.5.1. Edge routes with at most two bends can be classified as 0-bend, 1-bend, 2-bend planar or 2-bend non-planar, as illustrated in Figure 5.25.

Lemma 5.12. *Suppose in a given 2-bend 3-D orthogonal point-drawing of an m-edge graph G the number of 0-bend edge routes is k_0 and the number of 2-bend planar edge*

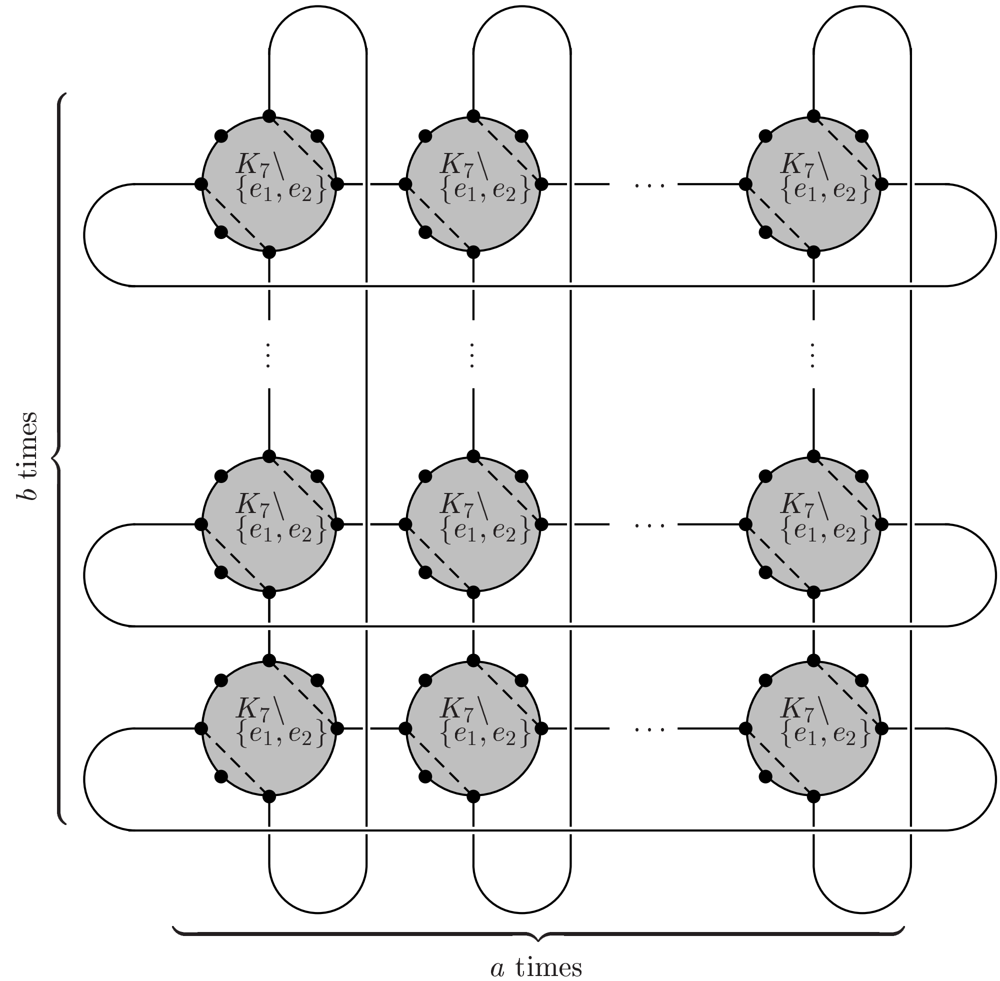


Figure 5.24: The graph $G_{a,b}$.

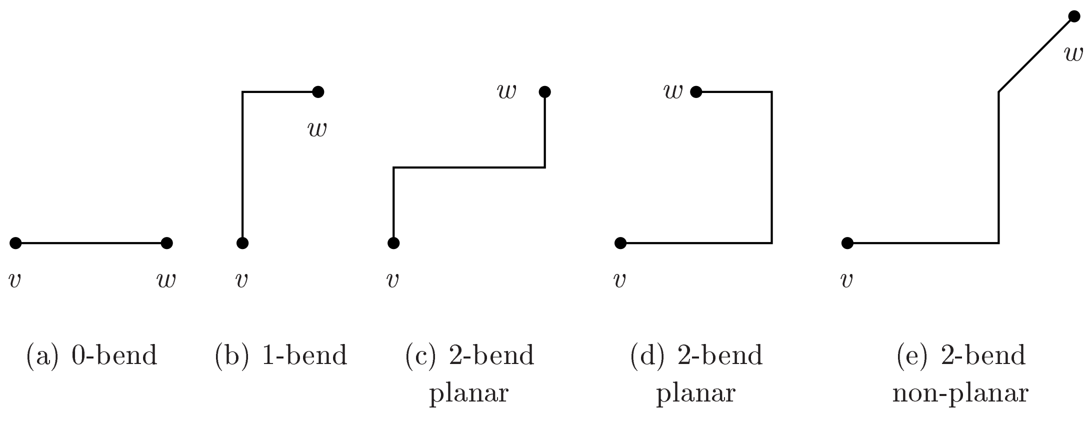


Figure 5.25: Edge routes vw with at most two bends.

routes is k_2. Then there exists a general position 3-D orthogonal point-drawing of G with $2m + k_0 + k_2$ bends.

Proof. [2] We now show that by inserting planes and adding bends to the edge routes that the given 2-bend drawing can be transformed into a drawing with a general position vertex layout and the stated number of bends.

Consider a grid-plane P containing k vertices ($k > 1$). As illustrated in Figure 5.26, replace the plane by k adjacent planes, and position each of the k vertices in a unique plane.

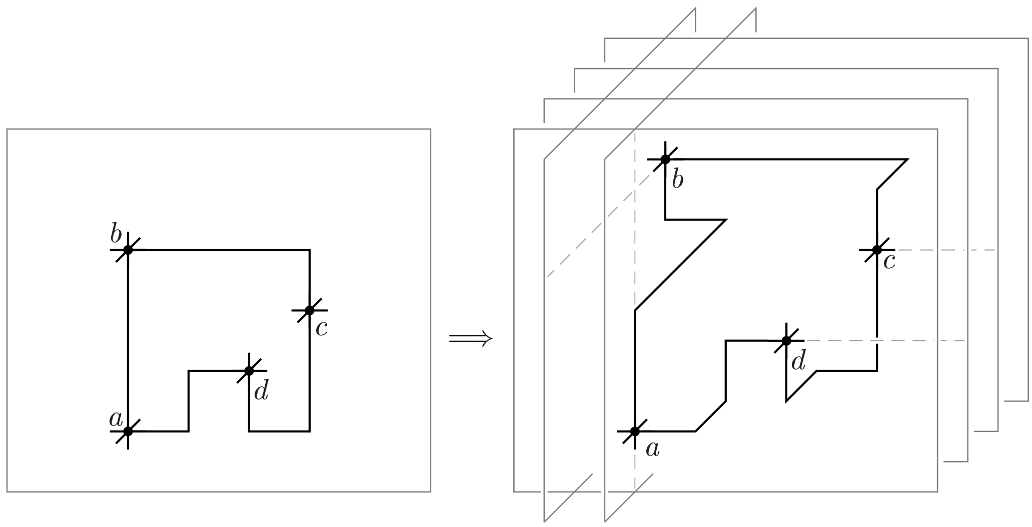


Figure 5.26: Removing a plane containing many vertices.

A 0-bend edge route is split in the middle and replaced by the 2-bend planar edge route shown in Figure 5.25(c). (If the 0-bend edge has length one then an extra plane perpendicular to the original plane is also inserted.)

Edge segments from an edge with at least one bend and incident to a vertex v are routed in the plane containing v. For a 1-bend edge route vw in the original plane, an extra segment is inserted perpendicular to P, running between the planes containing v and w. Hence vw is replaced by a 2-bend non-planar edge route.

For a 2-bend edge route vw in the original plane, the middle segment of vw is routed arbitrarily in the plane containing v or w, and a third segment is inserted perpendicular

[2]This proof was developed in conjunction with Antonios Symvonis.

to P, running between the planes containing v and w. Hence vw is replaced by a 3-bend non-planar edge route.

For a 2-bend non-planar edge route vw incident to one of the k vertices, the segment of vw perpendicular to P is extended in the obvious manner. Similarly, an edge passing through the original plane and not incident to any of the k vertices, is extended so that it passes through all k planes.

This process is continued until there are no grid-planes containing more than one vertex. Note that a 0-bend edge route will firstly be replaced by a 2-bend planar edge, and in a second transformation will be replaced by a 3-bend edge route (as shown in Figure 5.26 for edge ab). The resulting drawing has no crossings, has a general position vertex layout, and every edge has two bends except for the 0-bend and 2-bend planar edge routes in the original drawing, which now have three bends. Hence the new drawing has $2m + k_0 + k_2$ bends. □

Corollary 5.5. *There exists an infinite family of 6-regular n-vertex graphs, such that in any 2-bend 3-D orthogonal point-drawing of any one of the graphs, $k_0 + k_2 \geq 6n/7$.*

Proof. By Lemma 5.9, there exists an infinite family of graphs, each with at least $2m + 6n/7$ bends in any general position 3-D orthogonal point-drawing. If there is a 2-bend point-drawing of such a graph, then by Lemma 5.12 there is exists a general position point-drawing with $2m + k_0 + k_2$ bends. Hence $2m + k_0 + k_2 \geq 2m + 6n/7$, so $k_0 + k_2 \geq 6n/7$. □

The following two results are obtained using the same argument applied with Lemmas 5.10 and 5.11, respectively.

Corollary 5.6. *There exists an infinite family of 6-regular biconnected n-vertex graphs, such that in any 2-bend 3-D orthogonal point-drawing of any one of the graphs, $k_0+k_2 \geq 4n/7$.* □

Note that a 1-factor has $n/2$ edges, and $n/2 < 4n/7$, so there exists biconnected graphs for which any 2-bend 3-D orthogonal point-drawing has more than a 1-factor of 0-bend and 2-bend planar edge routes.

Corollary 5.7. *There exists an infinite family of 6-regular 4-connected n-vertex graphs, such that in any 2-bend 3-D orthogonal point-drawing of any one of the graphs, $k_0+k_2 \geq 2n/7$.* □

Chapter 6

The General Position Model for Two-Dimensional Orthogonal Box-Drawing

In this chapter we present algorithms for producing 2-D orthogonal box-drawings which establish improved degree-restriction results compared to existing algorithms. The methods and results presented in this chapter were published in Wood [221].

A 2-D orthogonal graph drawing is said to be in the *general position* model if no two vertices are intersected by a single grid-line. We call such a drawing a *general position* 2-D orthogonal drawing. This chapter, which describes algorithms for producing general position 2-D orthogonal drawings, is organised as follows. In Section 6.1 we present a framework for the main algorithms to follow. As discussed in Section 3.4.4 we classify such algorithms as *layout-* or *routing-based*. Section 6.2 describes our layout-based algorithm. The vertex layout algorithm is based on methods developed in Chapter 4 for the balanced vertex ordering problem. The arc-routing algorithm, which can be applied to an arbitrary general position 2-D vertex layout, constructs and colours the vertices of a certain graph. The drawings produced have the smallest known degree-restriction bound for bounded aspect ratio drawings. This strategy is generalised to a multi-dimensional setting in Chapter 7. Routing-based approaches to 2-D general position box-drawing are given by Papakostas and Tollis [164, 169] and Biedl and Kaufmann [30].

6.1 Representation

Consider a general position 2-D orthogonal box-drawing of a graph G. Since no two vertices share a common coordinate, this drawing induces X- and Y-vertex orderings of G, representing the relative coordinates of the vertices. The assignment of ports to edge routes induces a (non-proper) 2-colouring of $A(G)$, where an arc $\overrightarrow{vw} \in A(G)$ is coloured $i \in \{X, Y\}$ if the edge route vw uses an i-port at v.

Since each pair of vertices differ in both coordinates, an edge route has at least one bend. Our algorithms use exactly one bend per edge route. The ports used by a 1-bend edge route must be perpendicular and point toward the other vertex (see Figure 6.1); i.e., reversal arcs are coloured differently. We therefore represent a general position 2-D orthogonal box-drawing of G by:

- A (*2-D general position*) *vertex layout* consisting of vertex orderings $(<_X, <_Y)$ of G, which represent the relative coordinates of the vertices in each dimension.
- A (*2-D general position*) *arc-routing* of G consisting of a 2-colouring of $A(G)$ such that for every edge $vw \in E(G)$, the reversal arcs $\overrightarrow{vw} \in A(G)$ and $\overrightarrow{wv} \in A(G)$ are coloured differently[1].

In the X-ordering a predecessor (respectively, successor) arc of a vertex v is called a *X-predecessor* (*X-successor*) *arc* of v. We denote the number of predecessor and successor arcs of v in the X-ordering by $p_X(v)$ and $s_X(v)$ respectively. The cost of a vertex $v \in V(G)$ in the X-ordering, defined in Section 4.1 to be $|s_X(v) - p_X(v)|$, is denoted by $c_X(v)$. Similarly definitions are made for the Y-ordering.

For each vertex $v \in V(G)$ and direction $d \in \{\pm X, \pm Y\}$, the set of outgoing arcs $\overrightarrow{vw} \in A(G)$ with w in direction d from v, is denoted by $A_G(v)\langle d \rangle$. We represent a

[1] A 2-D arc-routing can simply be represented by an orientation of the edges. For an edge vw oriented from v to w, the arcs $\overrightarrow{vw}$ and $\overrightarrow{wv}$ are coloured X and Y, respectively. This is the approach taken by Biedl and Kaufmann [30]. We use the 2-colouring representation for consistency with our representation for multi-dimensional arc-routings used in Chapters 5 and 7.

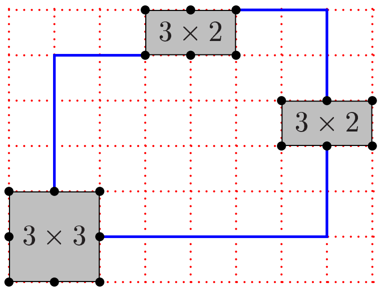


Figure 6.1: 2-D 1-bend edge routes

quadrant relative to v by the corresponding pair of non-opposite directions. The set of arcs $\overrightarrow{vw} \in A(G)$ with w in some quadrant Q relative to v is denoted by $A_G(v)\langle Q\rangle$; i.e.,

$$A_G(v)\langle Q\rangle = \bigcap_{d\in Q} A_G(v)\langle d\rangle \quad .$$

Using the notation introduced in Section 2.1, for some dimension $i \in \{X, Y\}$, $A_G(v)\langle i^{\pm}\rangle\, i$ refers to the arcs in $A_G(v)\langle i^{\pm}\rangle$ coloured i. If an arc $vw \in A_G(v)\langle X^{-}\rangle\, X$, for example, then the edge route vw will leave v on the left. A vertex v clearly must have width at least

$$M_X(v) = \max\left\{\left|A_G(v)\left\langle Y^{+}\right\rangle[Y]\right|, \left|A_G(v)\left\langle Y^{-}\right\rangle[Y]\right|\right\} \quad ,$$

and height

$$M_Y(v) = \max\left\{\left|A_G(v)\left\langle X^{+}\right\rangle[X]\right|, \left|A_G(v)\left\langle X^{-}\right\rangle[X]\right|\right\} \quad .$$

We now present an algorithm, which given a 2-D general position vertex layout and arc-routing of a graph G, determines a general position 2-D orthogonal box-drawing of G. This algorithm will form the final step in our graph drawing algorithms to follow.

Algorithm 6.1. General Position 2-D Box-Drawing

Input: • graph G

• 2-D general position vertex layout of $V(G)$

• 2-D general position arc-routing of $A(G)$

Output: general position 2-D box-drawing of G

---

1. Represent each vertex $v \in V(G)$ by a $M_X(v) \times M_Y(v)$ rectangle with maximum corner at

$$\left(\sum_{w \leq_X v} M_X(w), \sum_{w \leq_Y v} M_Y(w) \right) .$$

2. For each vertex $v \in V(G)$ and $i \in \{X, Y\}$, assign ports on the $(\pm i)$-face of v to the arcs $\overrightarrow{vw} \in A_G(v)\langle i^{\pm}\rangle[i]$. To reduce the number of crossings we assign particular ports on v to these arcs in order of the distance from v to w in the i-ordering, as illustrated in Figure 6.2.

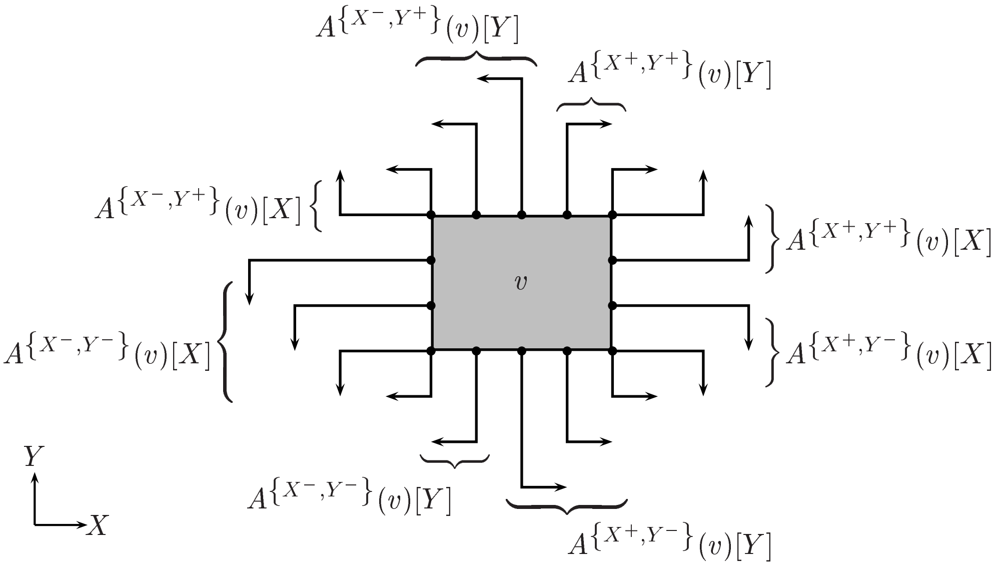


Figure 6.2: Port assignments at a vertex v.

3. For each edge $vw \in E(G)$, if the arcs $\overrightarrow{vw}$ and $\overrightarrow{wv}$ have been assigned an X-port and a Y-port at v and at w with coordinates of (x_v, y_v) and (x_w, y_w) respectively, then the edge vw is routed from v to w with one bend as follows.

$$(x_v, y_v) \rightarrow (x_w, y_v) \rightarrow (x_w, y_w)$$

The next result follows immediately from the above construction.

Lemma 6.1. *The algorithm* General Position 2-D Box-Drawing *determines a* *general position 2-D orthogonal box-drawing of G with bounding box*

$$\left(\sum_v M_X(v)\right) \times \left(\sum_v M_Y(v)\right) \quad .$$

Each vertex v has surface

$$2\left(M_X(v) + M_Y(v)\right) \quad .$$

□

6.2 Layout-Based Approach

In a 2-D general position vertex layout of a graph G, the cost of a vertex $v \in V(G)$ is defined to be the average[2] cost of v over the X- and Y-orderings; i.e.,

$$c(v) = \frac{1}{2}\left(c_X(v) + c_Y(v)\right) \quad .$$

We are interested in the following problem.

Problem 6.1. 2-D GENERAL POSITION VERTEX LAYOUT
Instance : Graph G, integer $K \geq 0$.
Question : Does G have a 2-D general position vertex layout with $\max_v c(v) \leq K$?

We conjecture that this problem is NP-complete. In Section 6.2.3, we provide an algorithm which determines a vertex layout with a tight bound on $\max_v c(v)$.

6.2.1 Arc-Routing Algorithm

The following algorithm, given an arbitrary 2-D general position vertex layout of a graph G, determines a 2-D general position arc-routing of G. To represent the colouring of $A(G)$ we vertex-colour a graph H with vertex set $V(H) = A(G)$.

Algorithm 6.2. 2-D GENERAL POSITION ARC-ROUTING

Input: 2-D general position vertex layout of a graph G.
Output: 2-D general position arc-routing of $A(G)$.

---

[2] We use the 'average' here rather than the 'sum' since this definition will be extended to a multi-dimensional setting in Chapter 7.

1. For each edge $vw \in E(G)$, add the edge $\{vw, wv\}$ to $E(H)$ (called an *r-edge*).

2. For each vertex $v \in V(G)$ and for each quadrant Q relative to v,

   (a) Arbitrarily partition the arcs in $A_G(v)\langle Q\rangle$ into pairs $\{\overrightarrow{vu_1}, \overrightarrow{vw_1}\}, \ldots, \{\overrightarrow{vu_k}, \overrightarrow{vw_k}\}$, with at most one *leftover* arc in $A_G(v)\langle Q\rangle$ not included in a pair.

   (b) Add an edge (called a *q-edge*) to $E(H)$ between the vertices corresponding to the arcs $\overrightarrow{vu_j}$ and $\overrightarrow{vw_j}$, $1 \leq j \leq k$.

3. Split those vertices in $v \in V(G)$ with at least three leftover arcs in $A_G(v)$ into two groups V_X and V_Y of equal size (or differing by one).

4. For each vertex $v \in V(G)$:

   (a) If there are exactly two leftover arcs $\overrightarrow{vu}, \overrightarrow{vw} \in A_G(v)$ then add an edge (called an l-edge) between the vertices in H corresponding to $\overrightarrow{vu}$ and $\overrightarrow{vw}$.

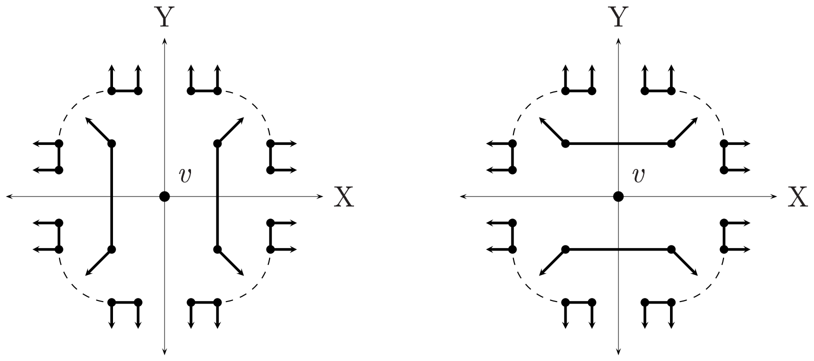
   (b) If $v \in V_i$ ($i \in \{X, Y\}$) has exactly three leftover arcs then add an edge, called an l-edge, between the vertices of H corresponding to the two leftover arcs at v which are both i-successor arcs or both i-predecessor arcs (see Figure 6.3).

   (c) If $v \in V_i$ ($i \in \{X, Y\}$) has four leftover arcs then add edges (called l-edges) between the vertices of H corresponding to the two leftover i-successor arcs of v, and between the vertices of H corresponding to the two leftover i-predecessor arcs of v (see Figure 6.3).

5. Determine a 2-colouring of $A(G)$ from a vertex-colouring of H with two colours.

Lemma 6.2. *The algorithm* 2-D General Position Arc-Routing *determines a 2-D general position arc-routing of G in $O(m+n)$ time such that for each vertex v,*

$$2(M_X(v) + M_Y(v)) \;\leq\; \deg(v) + c(v) + 4 \;\;,$$



(a) $v \in V_X$ (b) $v \in V_Y$

Figure 6.3: Connecting leftover arcs at v.

and for each $i \in \{X, Y\}$,

$$\sum_v M_i(v) \;\leq\; \frac{m}{2} \;+\; \frac{1}{4}\left(3n + 1 + \sum_v c_i(v)\right) \quad .$$

Proof. A cycle in H consists of alternating r- and (q- or l-) edges and is therefore of even length. So H is bipartite, and a 2-colouring of H can be computed in $O(|E(H)|) = O(m)$ time, thus determining a 2-colouring of $A(G)$. Since the vertices corresponding to reversal arcs $\overrightarrow{vw}$ and $\overrightarrow{wv}$ are adjacent in H, this 2-colouring of $A(G)$ is a 2-D arc-routing of $A(G)$.

For each quadrant q relative to a vertex v and in each pair of the partition of $A_G(v)\langle Q\rangle$, the arcs $\overrightarrow{vu_i}$ and $\overrightarrow{vw_i}$ are coloured differently, so we have the following bounds on, for example, the number of X-successor arcs $\overrightarrow{vw}$ coloured X.

$$\left\lfloor \frac{|A_G(v)\langle\{X^+, Y^+\}\rangle\,|}{2} \right\rfloor + \left\lfloor \frac{|A_G(v)\langle\{X^+, Y^-\}\rangle\,|}{2} \right\rfloor$$
$$\leq\; \left|A_G(v)\left\langle X^+ \right\rangle X\right| \;\leq$$
$$\left\lceil \frac{|A_G(v)\langle\{X^+, Y^+\}\rangle\,|}{2} \right\rceil + \left\lceil \frac{|A_G(v)\langle\{X^+, Y^-\}\rangle\,|}{2} \right\rceil \quad .$$

So,

$$\frac{s_X(v)}{2} - 1 \;\leq\; \left|A_G(v)\left\langle X^+ \right\rangle X\right| \;\leq\; \frac{s_X(v)}{2} + 1 \quad .$$

Similarly, we have the following bound on the number of X-predecessor arcs coloured

X.

$$\frac{p_X(v)}{2} - 1 \ \le\ \left|A_G(v)\langle X^-\rangle\, X\right| \ \le\ \frac{p_X(v)}{2} + 1\ .$$

Recall that $M_Y(v) \;=\; \max\{|A_G(v)\langle X^+\rangle\, X|, |A_G(v)\langle X^-\rangle\, X|\}$. So

$$\begin{aligned}\frac{1}{2}\max\{s_X(v), p_X(v)\} - 1 \ &\le M_Y(v) \ \le\ \frac{1}{2}\max\{s_X(v), p_X(v)\} + 1 \\ \frac{1}{4}(\deg(v) + c_X(v)) - 1 \ &\le M_Y(v) \ \le\ \frac{1}{4}(\deg(v) + c_X(v)) + 1 \qquad \text{(by (4.1))}\end{aligned}$$

Using the same argument for the number of Y-successors and Y-predecessors coloured Y, for each $i, j \in \{X, Y\}$ $(i \neq j)$,

$$\frac{1}{4}(\deg(v) + c_j(v)) - 1 \ \le\ M_i(v) \ \le\ \frac{1}{4}(\deg(v) + c_j(v)) + 1\ . \tag{6.1}$$

So

$$\begin{aligned}2\,(M_X(v) + M_Y(v)) \ &\le\ 2\left(\frac{1}{4}(2\deg(v) + c_X(v) + c_Y(v)) + 2\right) \\ &=\ \deg(v) + \frac{c_X(v) + c_Y(v)}{2} + 4 \\ &=\ \deg(v) + c(v) + 4\ .\end{aligned}$$

Now, in each quadrant relative to a vertex v, there is at most one leftover arc at v. A vertex v with at most two leftover arcs has, for each $i \in \{X, Y\}$,

$$M_i(v) \ \le\ \left\lceil \frac{\max\{s_i(v), p_i(v)\}}{2} \right\rceil\ .$$

A vertex $v \in V_i$ with at least three leftover arcs has

$$\begin{aligned}M_i(v) \ &\le\ \left\lceil \frac{\max\{s_i(v), p_i(v)\}}{2} \right\rceil, \text{ and} \\ M_j(v) \ &\le\ \frac{\max\{s_j(v), p_j(v)\}}{2} + 1 \quad (j \neq i, j \in \{X, Y\})\ .\end{aligned}$$

So, for each $i \in \{X, Y\}$,

$$\begin{aligned}\sum_v M_i(v) \ &=\ \sum_{v \notin V_j} M_i(v) \ +\ \sum_{v \in V_j} M_i(v) \\ &\le\ \sum_{v \notin V_j} \frac{\max\{s_i(v), p_i(v)\} + 1}{2} \ +\ \sum_{v \in V_j}\left(\frac{\max\{s_j(v), p_j(v)\}}{2} + 1\right) \\ &=\ \frac{n}{2} \ +\ \frac{|V_j|}{2} \ +\ \sum_v \frac{\max\{s_i(v), p_i(v)\}}{2}\end{aligned}$$

$$
\begin{aligned}
&= \frac{n}{2} + \frac{\lceil n/2 \rceil}{2} + \sum_v \frac{\deg(v) + c_i(v)}{4} \quad \text{(by (4.1))} \\
&\leq \frac{n}{2} + \frac{n+1}{4} + \frac{m}{2} + \sum_v \frac{c_i(v)}{4} \\
&\leq \frac{m}{2} + \frac{1}{4}\left(3n + 1 + \sum_v c_i(v)\right) \quad . \qquad \square
\end{aligned}
$$

6.2.2 Fixed Vertex Layout Drawings

We now derive results for a fixed general position vertex layout.

Algorithm 6.3. FIXED GENERAL POSITION 2-D BOX-DRAWING

Input: • graph G

• 2-D general position vertex layout of $V(G)$

Output: layout-preserving 2-D orthogonal box-drawing of G.

---

1. Determine an arc-routing with Algorithm 6.2 2-D GENERAL POSITION ARC-ROUTING.
2. Apply Algorithm 6.1 GENERAL POSITION 2-D BOX-DRAWING.

Theorem 6.1. *For an arbitrary 2-D general position vertex layout, Algorithm* FIXED GENERAL POSITION 2-D BOX-DRAWING *determines a 2-D orthogonal box-drawing of G in $O(m+n)$ time such that:*

- *Each edge route has 1 bend.*
- *Each vertex is 2-degree-restricted.*
- *The aspect ratio of a vertex v is at most $2 + o(\deg(v))$.*
- *The bounding box is at most*

$$\left(m + \frac{3n+1}{4}\right) \times \left(m + \frac{3n+1}{4}\right) \quad .$$

Proof. By Lemma 6.2, for every vertex vertex v, surface $(v) \leq \deg(v) + c(v) + 4$. Since $c(v) \leq \deg(v)$, v is 2-degree-restricted.

For each $i \in \{X, Y\}$, $0 \leq c_i(v) \leq \deg(v)$, so by (6.1),

$$\frac{1}{4}\deg(v) - 1 \;\leq\; M_i(v) \;\leq\; \frac{1}{2}\deg(v) + 1 \;. \tag{6.2}$$

Hence,

$$\max\left\{\frac{M_X(v)}{M_Y(v)}, \frac{M_Y(v)}{M_X(v)}\right\} \;\leq\; \frac{\deg(v)/2 + 1}{\deg(v)/4 - 1} \;=\; 2 + o\left(\deg(v)\right)$$

So v has aspect ratio at most $2 + o\left(\deg(v)\right)$. The bounding box is

$$\left(\sum_v M_X(v)\right) \times \left(\sum_v M_Y(v)\right)$$

Since $c_i(v) \leq \deg(v)$ and by Lemma 6.2, the bounding box is

$$\begin{aligned} &\left(\frac{m}{2} + \frac{1}{4}\left(3n + 1 + \sum_v \deg(v)\right)\right) \;\times\; \left(\frac{m}{2} + \frac{1}{4}\left(3n + 1 + \sum_v \deg(v)\right)\right) \\ = &\left(m + \frac{3n+1}{4}\right) \;\times\; \left(m + \frac{3n+1}{4}\right) \;. \qquad \square \end{aligned}$$

6.2.3 Balanced Vertex Layout Drawings

We now describe how the methods developed for the balanced ordering problem in Section 4.3 can be applied to find ‘balanced’ 2-D general position vertex layouts. By balanced we mean that there is an upper bound on the cost $c(v)$ for each vertex v. The following algorithm, which is similar to the vertex layout technique of Biedl and Kaufmann [30], is illustrated in Figure 6.4.

Algorithm 6.4. Balanced 2-D General Position Vertex Layout

Input: graph G.

Output: 2-D general position vertex layout of G.

---

1. Determine an arbitrary vertex ordering $(v_1, v_2, \ldots, v_n)$ of G.

2. Determine the X-ordering using Algorithm 4.1 Median Placement Ordering with insertion ordering $(v_1, v_2, \ldots, v_n)$.

3. Determine the Y-ordering using Algorithm 4.1 Median Placement Ordering with insertion ordering $(v_n, v_{n-1}, \ldots, v_1)$.

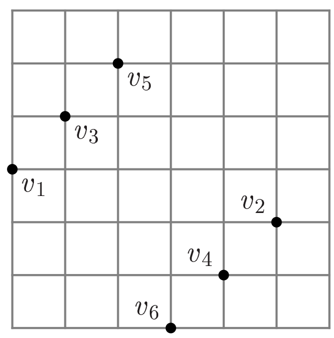


Figure 6.4: Balanced 2-D vertex layout of K_6.

Theorem 6.2. *The algorithm* Balanced 2-D General Position Vertex Layout *determines a 2-D general position vertex layout of G in $O(m + n)$ time such that for each vertex v,*

$$c(v) \leq 1 + \frac{1}{2}\deg(v) \quad .$$

Proof. For each vertex v, by Lemma 4.3 concerning the performance of the algorithm Median Placement Ordering with arbitrary insertion orderings, $c_X(v) \leq s(v) + 1$ and $c_Y(v) \leq p(v)+1$, where $s(v)$ and $p(v)$ are the number of successors and predecessors of v respectively in the vertex ordering $(v_1, v_2, \ldots, v_n)$. So $c(v) \leq (s(v) + p(v) + 2)/2 = \deg(v)/2 + 1$. $\square$

Note that the above bound is tight up to the additive constant, since an extremal vertex in the X-ordering has $c_X(v) = \deg(v)$, so $c(v) \geq \deg(v)/2$. We now present our algorithm for 2-D orthogonal box-drawing using a balance general position vertex layout.

Algorithm 6.5. BALANCED GENERAL POSITION 2-D BOX-DRAWING

Input: graph G.

Output: 2-D orthogonal box-drawing of G.

---

1. Determine a general position vertex layout with Algorithm 6.4 BALANCED 2-D GENERAL POSITION VERTEX LAYOUT.
2. Determine an arc-routing with Algorithm 6.2 2-D GENERAL POSITION ARC-ROUTING.
3. Apply Algorithm 6.1 GENERAL POSITION 2-D BOX-DRAWING.

Theorem 6.3. *The algorithm* BALANCED GENERAL POSITION 2-D BOX-DRAWING *determines a 2-D orthogonal box-drawing of G in $O(m+n)$ time such that:*

- *Each edge route has* 1 *bend.*
- *Each vertex is $\frac{3}{2}$-degree-restricted.*
- *The aspect ratio of a vertex v is $2 + o(\deg(v))$.*
- *The bounding box area is $\left(\frac{3m+4n+2}{4}\right) \times \left(\frac{3m+4n+2}{4}\right)$.*

Proof. For any vertex v, by Lemma 6.2, $\text{surface}(v) = 2(M_X(v) + M_Y(v)) \leq \deg(v) + c(v) + 4$. By Theorem 6.2, in a 2-D balanced vertex layout, for every vertex $v \in V(G)$, $c(v) \leq 1 + \deg(v)/2$. So $\text{surface}(v) \leq \frac{3}{2}\deg(v) + 5$, and each vertex is 3/2-degree-restricted. By Lemma 6.2, the bounding box is

$$\left(\sum_v M_X(v)\right) \times \left(\sum_v M_Y(v)\right)$$

$$\leq \left(\frac{1}{4}\sum_v c_X(v) + \frac{m}{2} + \frac{3n+1}{4}\right) \times \left(\frac{1}{4}\sum_v c_Y(v) + \frac{m}{2} + \frac{3n+1}{4}\right) \quad .$$

The X- and Y-orderings are determined by algorithm MEDIAN PLACEMENT ORDERING, so by Corollary 4.1, the bounding box is at most

$$\left(\frac{m+n}{4} + \frac{m}{2} + \frac{3n+1}{4}\right) \times \left(\frac{m+n}{4} + \frac{m}{2} + \frac{3n+1}{4}\right)$$
$$= \left(\frac{3m+4n+2}{4}\right) \times \left(\frac{3m+4n+2}{4}\right) \quad . \qquad \square$$

6.2.4 Diagonal Vertex Layout Drawings

We now present an algorithm for producing 2-D orthogonal square-drawings using a diagonal layout.

Algorithm 6.6. DIAGONAL GENERAL POSITION 2-D SQUARE-DRAWING

Input: graph G.

Output: 2-D orthogonal square-drawing of G.

---

1. Determine a 2-D diagonal layout of G with the corresponding vertex ordering determined by Algorithm 4.1 MEDIAN PLACEMENT ORDERING (with insertion ordering determined by Algorithm 4.2 INSERTION ORDERING).

2. Determine a 2-D arc-routing with Algorithm 6.2 2-D GENERAL POSITION ARC-ROUTING.

3. Apply Algorithm 6.1 GENERAL POSITION 2-D BOX-DRAWING.

Theorem 6.4. *The algorithm* DIAGONAL GENERAL POSITION 2-D SQUARE-DRAWING *determines a diagonal layout 2-D square-drawing in $O(m+n)$ time such that:*

- *Each edge route has one bend.*

- *Each vertex is 2-degree-restricted.*

- *The bounding box volume is*

$$\left(\frac{3m}{4}+\frac{5n}{8}\right)\times\left(\frac{3m}{4}+\frac{5n}{8}\right)$$

Proof. We represent a vertex v by the $\max\{M_X(v), M_Y(v)\} \times \max\{M_X(v), M_Y(v)\}$ square. Algorithm 2-D General Position Arc-Routing determines a 2-D arc-routing such that,

$$M_X(v), M_Y(v) \ \leq\ \left\lceil \frac{\max\{s(v), p(v)\}}{2} \right\rceil \ .$$

Hence

$$\begin{aligned} \text{surface}\,(v) &= 4\left\lceil \frac{\max\{s(v), p(v)\}}{2} \right\rceil \\ &\leq 2\left(\max\{s(v), p(v)\}+1\right) \\ &\leq 2\deg(v)+2 \ . \end{aligned}$$

So each vertex v is 2-degree-restricted. The bounding box side length is at most

$$\begin{aligned} &\sum_v \left\lceil \frac{\max\{s(v), p(v)\}}{2} \right\rceil \\ &\leq \sum_v \left(\frac{1}{2}\left(\max\{s(v), p(v)\}+1\right)\right) \\ &\leq \frac{1}{2}\left(\frac{3m}{2}+\frac{n}{4}+n\right) \qquad\qquad \text{(by Theorem 4.2)} \\ &\leq \frac{3m}{4}+\frac{5n}{8} \ . \end{aligned}$$

The bounding box volume bound follows. □

Chapter 7

The General Position Model for Multi-Dimensional Orthogonal Box-Drawing

In this chapter we present and analyse algorithms for producing general position D-dimensional orthogonal box-drawings ($D \geq 3$) of arbitrary degree graphs. For $D = 3$, our results establish improved bounds for the degree-restriction of vertices. This chapter was published in Wood [222].

A D-dimensional orthogonal drawing is in the *general position model*, called a *general position* orthogonal drawing, if no two vertices are intersected by a single $(D-1)$-dimensional grid-hyperplane. This chapter presents algorithms for determining general position D-dimensional orthogonal drawings, for some constant $D \geq 3$. These algorithms generalise those for general position 2-D orthogonal box-drawing presented in Chapter 6.

This chapter is organised as follows. Section 7.1 provides a framework for the development of the main algorithms to follow. As discussed in Section 3.4.4, algorithms for general position orthogonal graph drawing can be classified as layout- or routing-based. We present layout-based algorithms in Section 7.2 and a routing-based algorithm for general position 3-D drawing in Section 7.3.

7.1 Framework

Consider a general position D-dimensional orthogonal drawing of a graph G. Since no two vertices share a common coordinate, this drawing induces D vertex orderings of G, representing the relative coordinates of the vertices in each dimension. The assignment of ports to edge routes, induces a (non-proper) D-colouring of $A(G)$, where an arc $\overrightarrow{vw} \in A(G)$ is coloured $i \in \{1, 2, \ldots, D\}$ if the edge route vw uses an i-port at v. Since each pair of vertices differ in all D coordinates, each edge route has at least $D-1$ bends. The ports used by a $(D-1)$-bend edge route must be perpendicular and point toward the other vertex, as in Figure 7.1, so for each edge vw the reversal arcs $\overrightarrow{vw}, \overrightarrow{wv} \in A(G)$ are coloured differently.

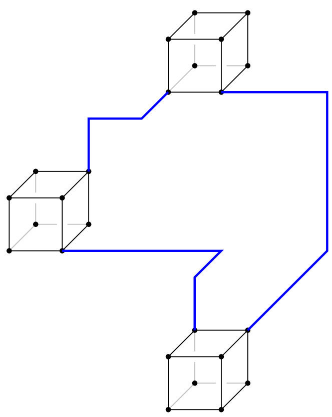


Figure 7.1: $(D-1)$-bend edge routes in $D = 3$ dimensions.

We therefore represent a general position D-dimensional orthogonal drawing of G by:

- A *(D-dimensional general position) vertex layout* of $V(G)$, consisting of D vertex orderings $(<_1, <_2, \ldots, <_D)$ of G. We call $<_i$, $1 \leq i \leq D$, the *i-ordering* of the layout, and for $D = 3$ we will refer to the 1-, 2-, and 3-orderings as the X-, Y- and Z-orderings.

- A *(D-dimensional general position) arc-routing* of $A(G)$, consisting of a D-colouring of $A(G)$ such that for each edge $vw \in E(G)$ the reversal arcs $\overrightarrow{vw}, \overrightarrow{wv} \in A(G)$ are coloured differently.

Consider a D-dimensional general position vertex layout of a graph G. In each i-ordering, $1 \leq i \leq D$, a predecessor (respectively, successor) arc of a vertex v is called an *i-predecessor* (*i-successor*) *arc* of v (see Section 4.1). We denote the number of predecessor and successor arcs of v in the i-ordering by $p_i(v)$ and $s_i(v)$, respectively. The cost of a vertex $v \in V(G)$ in the i-ordering, defined in Chapter 4 to be $|s_i(v) - p_i(v)|$, is denoted $c_i(v)$. The *cost* of v is defined to be the average cost of v over the D orderings; i.e.,

$$c(v) = \frac{1}{D} \sum_{1 \leq i \leq D} c_i(v)$$

The following problem is of interest.

Problem 7.1. **D-DIMENSIONAL GENERAL POSITION VERTEX LAYOUT**
Instance : graph G, integer $K \geq 0$.
Question : Does G have a D-dimensional general position vertex layout with $c(v) \leq K$ for every vertex $v \in V(G)$?

We conjecture that this problem is NP-complete. In Section 7.2.3 we provide lower and upper bounds for this problem. The methods to be described in this section are summarised in the following algorithm.

Algorithm 7.1. D-DIMENSIONAL GENERAL POSITION BOX-DRAWING

Input:
- graph G
- D-dimensional general position vertex layout of $V(G)$
- D-dimensional general position arc-routing of $A(G)$

Output: general position D-dimensional box-drawing of G

---

1. For each vertex $v \in V(G)$, determine the size $\alpha_1(v) \times \alpha_2(v) \times \cdots \times \alpha_D(v)$ of the box representing v (see Section 7.1.1).

2. Position each vertex $v \in V(G)$ at the grid-point with maximum i-coordinate of

$$\sum_{w \leq v} \alpha_i(w) \quad .$$

Note that the bounding box has size

$$\left(\sum_v \alpha_1(v)\right) \times \left(\sum_v \alpha_2(v)\right) \times \cdots \times \left(\sum_v \alpha_D(v)\right) \quad .$$

3. Assign ports to edges, as described in Section 7.1.2. (An arc $\overrightarrow{vw} \in A_G(v)[i]$ will be assigned a port on the i-face of v pointing towards w.)

4. For each edge $vw \in E(G)$ construct a $(D-1)$-bend edge route as follows. Suppose the arc $\overrightarrow{vw} \in A(G)$ is coloured $i \in \{1, 2, \ldots, D\}$ and its reversal arc $\overrightarrow{wv}$ is coloured $j > i$. The edge route vw consists of D contiguous grid-line segments which traverse the sides of the hypercube with corners at port$(\overrightarrow{vw})$ and port$(\overrightarrow{wv})$. These segments are respectively parallel to the $i, (i-1), \ldots, 1, (i+1), (i+2), \ldots, (j-1), D, (D-1), \ldots, j$ axes.

5. Remove edge crossings using Algorithm 7.2 Box-Drawing Remove Edge Crossings.

For a given general position vertex layout, $A_G(v)\langle d\rangle$ denotes the set of outgoing arcs at some vertex $v \in V(G)$ in the direction d; i.e.,

$$A_G(v)\langle d\rangle = \begin{cases} \{\overrightarrow{vw} \in A_G(v) : v <_d w\}, & \text{if } d > 0; \\ \{\overrightarrow{vw} \in A_G(v) : w <_{-d} v\}, & \text{if } d < 0. \end{cases}$$

For each direction $d \in \{1, 2, \ldots, D\}$ and vertex $v \in V(G)$, the set of arcs in $A_G(v)\langle d\rangle$, which are coloured i is denoted $A_G(v)\langle d\rangle[i]$. If an arc $\overrightarrow{vw} \in A_G(v)\langle i^{\pm}\rangle[i]$ then the edge route vw uses an $(i^{\pm})$-port at v. The maximum of the number of edges routed on the (i^+)-face and (i^-)-face of v is denoted $M_i(v)$; i.e.,

$$M_i(v) = \max\left\{\left|A_G(v)\langle i\rangle[i]\right|, \left|A_G(v)\langle i^-\rangle[i]\right|\right\}.$$

Clearly surface$_i(v)$ must be at least $M_i(v)$.

7.1.1 Determining Vertex Size

We now describe how to determine the size of the grid-box representing a vertex v given the number of edges routed on each face of v. For each vertex v we wish to determine positive integers $\alpha_i(v)$, $1 \leq i \leq D$, such that surface$_i(v)$ is at least $M_i(v)$; i.e.,

$$\text{determine } \alpha_i(v), 1 \leq i \leq D \text{ such that } \forall i \prod_{\substack{1 \leq j \leq D \\ j \neq i}} \alpha_j(v) \geq M_i(v). \tag{7.1}$$

Our aim is to minimise the surface(v) such that (7.1) is satisfied. For each i with $M_i(v) = 0$ we replace $M_i(v)$ by 1. A solution to the new problem with

$$\text{surface}(v) \leq k \left(\sum_i 2M_i(v) \right) + k'$$

is a solution to the original problem with

$$\text{surface}(v) \leq k \left(\sum_i 2M_i(v) \right) + \big(k' + D\big) \quad .$$

So we now assume that $M_i(v) \geq 1$.

We define $M_*(v)$ to be the geometric mean of $\{M_i(v) : i = 1, 2, \ldots, D\}$; i.e.,

$$M_*(v) = \left(\prod_i M_i(v) \right)^{1/D} \quad .$$

Lemma 7.1. *A real-valued exact solution to* (7.1) *can be obtained with* $\alpha_i(v) = r_i(v)$ *where we define*

$$r_i(v) \quad = \quad \frac{M_*(v)^{D/(D-1)}}{M_i(v)} \tag{7.2}$$

Proof. For each i, $1 \leq i \leq D$,

$$\begin{aligned}
\text{surface}_i(v) &= \prod_{\substack{1 \leq k \leq D \\ k \neq i}} r_k(v) \\
&= \prod_{\substack{1 \leq k \leq D \\ k \neq i}} \Big(M_*(v)^{D/(D-1)} \Big/ M_k(v) \Big) \\
&= \prod_{\substack{1 \leq k \leq D \\ k \neq i}} \left(\left(\prod_{1 \leq j \leq D} M_j(v) \right)^{1/(D-1)} \Bigg/ M_k(v) \right) \\
&= \prod_{\substack{1 \leq k \leq D \\ k \neq i}} \left(\left(\prod_{\substack{1 \leq j \leq D \\ j \neq k}} M_j(v) \right) \Bigg/ M_k(v)^{D-2} \right)^{1/(D-1)}
\end{aligned}$$

$$
\begin{aligned}
&= \left(\prod_{\substack{1 \le k \le D \\ k \ne i}} \left(\left(\prod_{\substack{1 \le j \le D \\ j \ne k}} M_j(v) \right) \Big/ M_k(v)^{D-2} \right) \right)^{1/(D-1)} \\
&= \left(\left(M_i(v)^{D-1} \prod_{\substack{1 \le j \le D \\ j \ne i}} M_j(v)^{D-2} \right) \Big/ \left(\prod_{\substack{1 \le k \le D \\ k \ne i}} M_k(v)^{D-2} \right) \right)^{1/(D-1)} \\
&= \left(M_i(v)^{D-1} \right)^{1/(D-1)} \\
&= M_i(v) \qquad \square
\end{aligned}
$$

This result suggests to obtain an integer-valued solution to (7.1), set $\alpha_i(v) = \lceil r_i(v) \rceil$ for each i. We now present a technical lemma which will be applied in the analysis of the algorithms to follow. It essentially says that if the ratios among $\{M_1, M_2, \ldots, M_D\}$ are bounded then surface(v) is asymptotically $2\sum_i M_i(v)$, which is the obvious lower bound.

Theorem 7.1. *If for each i, j, $1 \le i, j \le D$, $M_i(v)/M_j(v) \le f(v)$, for some function $f : V(G) \to \mathbb{R}$, then setting $\alpha_i(v) = \lceil r_i(v) \rceil$,*

$$
\mathit{surface}(v) \ \le\ 2\sum_i M_i(v) \ +\ O\big(f(v)^{D-2}\big) \left(\sum_i M_i(v) \right)^{(D-2)/(D-1)}
$$

Proof. We initially show that $M_i(v)/M_j(v) \le f(v)$ implies $r_i(v)/r_j(v) \le f(v)$ for all i, j, $1 \le i, j \le D$.

$$
\begin{aligned}
\max_i r_i(v) \,/\, \min_j r_j(v) \le\ & \left(\max_i \frac{M_*(v)^{D/(D-1)}}{M_i(v)} \right) \Big/ \left(\min_i \frac{M_*(v)^{D/(D-1)}}{M_i(v)} \right) \\
=\ & M_j(v)/M_i(v) \ ,
\end{aligned}
$$

where $M_i(v)$ and $M_j(v)$ are maximum and minimum of $\{M_1(v), M_2(v), \ldots, M_D(v)\}$. Hence $r_i(v)/r_j(v) \le f(v)$ for all i, j, $1 \le i, j \le D$.

For each i, $1 \le i \le D$,

$$
\text{surface}_i(v) \ =\ \prod_{j \ne i} \alpha_j(v) \ <\ \prod_{j \ne i} (r_j(v) + 1)
$$

Since $\prod_{j \neq i} r_j(v) = M_i(v)$, our aim to is to show that adding one to $r_j(v)$ does not increase surface$_i(v)$ by too much. To this end, we establish the following result, whose proof we defer until Lemma 7.2.

If $x_1, x_2, \ldots, x_n > 0$ with $x_i, x_j \leq \Lambda(\geq 1)$, for all i, j, $1 \leq i, j \leq D$, then

$$\prod_{i=1}^{n}(x_i + 1) \;=\; \prod_{i=1}^{n} x_i \;+\; O\big(\Lambda^{n-1}\big)\left(\prod_{i=1}^{n} x_i\right)^{(n-1)/n} \tag{7.3}$$

Applying (7.3), with $\{x_1, x_2, \ldots, x_n\} = \{r_j : j \neq i\}$ and $\Lambda = f(v)$, we obtain

$$\prod_{j\neq i}(r_j + 1) \;\leq\; \prod_{j\neq i} r_j \;+\; O\big(f(v)^{D-2}\big)\left(\prod_{\neq i} r_j\right)^{(D-2)/(D-1)} .$$

Hence

$$\text{surface}_i(v) \;\leq\; M_i(v) \;+\; O\big(f(v)^{D-2}\big)\, M_i(v)^{(D-2)/(D-1)}$$

Therefore

$$\begin{aligned}
&\text{surface}\,(v)\\
&\leq\; 2\sum_i M_i(v) \;+\; \sum_i O\big(f(v)^{D-2}\big)\, M_i(v)^{(D-2)/(D-1)}\\
&\leq\; 2\sum_i M_i(v) \;+\; O\big(f(v)^{D-2}\big)\sum_i M_i(v)^{(D-2)/(D-1)}\\
&\leq\; 2\sum_i M_i(v) \;+\; O\big(f(v)^{D-2}\big)\left(D^{1/(D-1)}\sum_i M_i(v)\right)^{(D-2)/(D-1)} \qquad \text{(by Cauchy-Schwarz)}\\
&\leq\; 2\sum_i M_i(v) \;+\; O\big(f(v)^{D-2}\big)\left(\sum_i M_i(v)\right)^{(D-2)/(D-1)} \qquad \square
\end{aligned}$$

Lemma 7.2. *If $x_1, x_2, \ldots, x_n > 0$ $(n \geq 2)$ with $x_i/x_j \leq \Lambda$, for all i, j, $1 \leq i, j \leq D$, then*

$$\prod_{i=1}^{n}(x_i + 1) \;\leq\; \prod_{i=1}^{n} x_i \;+\; O\big(\Lambda^{n-1}\big)\left(\prod_{i=1}^{n} x_i\right)^{(n-1)/n} \tag{7.4}$$

Proof. Suppose $x_1 \geq x_2 \geq \cdots \geq x_n$. Denote $\prod_{i=1}^{k} x_i$ by P_k. We proceed by induction on k with the following induction hypothesis.

$$\prod_{i=1}^{k}(x_i + 1) \;\leq\; P_k \;+\; O\Big(\Lambda^{k-1}\Big)\,(P_k)^{(k-1)/k} \tag{7.5}$$

Consider the case of $k = 2$. Since $x_1/x_2 \le \Lambda$, we have $x_2 \ge x_1/\Lambda$ and $x_1x_2 \ge x_1^2/\Lambda$. So $x_1 \le \sqrt{x_1x_2\Lambda}$. Similarly $x_2 \le \sqrt{x_1x_2\Lambda}$. So $(x_1+1)(x_2+1) = x_1x_2 + x_1 + x_2 + 1 \le x_1x_2 + 2\sqrt{x_1x_2\Lambda} + 1 \le x_1x_2 + O(\Lambda)\sqrt{x_1x_2}$. So the induction hypothesis holds for $n = 2$.

Suppose the induction hypothesis holds for all $k_0 < k$. Then

$$\begin{aligned}
&\prod_{i=1}^{k}(x_i+1)\\
&= (x_k+1)\prod_{i=1}^{k-1}(x_i+1)\\
&\le (x_k+1)\left(P_{k-1} + O\left(\Lambda^{k-2}\right)(P_{k-1})^{(k-2)/(k-1)}\right) \text{ (by the induction hypothesis)}\\
&\le P_k + x_k\,O\left(\Lambda^{k-2}\right)(P_{k-1})^{(k-2)/(k-1)} + P_{k-1} + O\left(\Lambda^{k-2}\right)(P_{k-1})^{(k-2)/(k-1)} \quad (7.6)
\end{aligned}$$

We now determine upper bounds in terms of Λ and P_k for each component of (7.6). Since $x_k \le x_j$, for j, $1 \le j \le k-1$, we have $x_k^{k-1} \le P_{k-1}$. So

$$x_k \le (P_{k-1})^{1/(k-1)} . \quad (7.7)$$

Now, for all j, $1 \le j \le k-1$, we have $x_j/x_k \le \Lambda$. So $x_k \ge x_j/\Lambda$, and hence

$$\begin{aligned}
x_k^{k-1} &\ge P_{k-1}/\Lambda^{k-1}\\
x_k^{k} &\ge P_k/\Lambda^{k-1}\\
x_k &\ge (P_k/\Lambda^{k-1})^{1/k}\\
x_k^{-1} &\le (\Lambda^{k-1}/P_k)^{1/k}\\
x_k^{-1} &\le \Lambda^{(k-1)/k}P_k^{-1/k}\\
P_{k-1} &\le \Lambda^{(k-1)/k}P_k^{1-1/k}\\
P_{k-1} &\le (\Lambda P_k)^{(k-1)/k} . \quad (7.8)
\end{aligned}$$

Now,

$$\begin{aligned}
&x_k\,O\left(\Lambda^{k-2}\right)(P_{k-1})^{(k-2)/(k-1)}\\
&\le (P_{k-1})^{1/(k-1)}\,O\left(\Lambda^{k-2}\right)(P_{k-1})^{(k-2)/(k-1)} \text{ (by (7.7))}\\
&\le O\left(\Lambda^{k-2}\right)P_{k-1}
\end{aligned}$$

$$\begin{aligned}
&\le O\left(\Lambda^{k-2}\right)(\Lambda P_k)^{(k-1)/k}(\text{by (7.8)})\\
&\le O\left(\Lambda^{k-1}\right)(P_k)^{(k-1)/k} \quad . && (7.9)
\end{aligned}$$

Now,

$$\begin{aligned}
&O\left(\Lambda^{k-2}\right)(P_{k-1})^{(k-2)/(k-1)}\\
&\le O\left(\Lambda^{k-2}\right)\left((\Lambda P_k)^{(k-1)/k}\right)^{(k-2)/(k-1)} (\text{by (7.8)})\\
&\le O\left(\Lambda^{k-2}\right)(\Lambda P_k)^{(k-2)/k} (\text{by (7.8)})\\
&\le O\left(\Lambda^{k-1}\right)(P_k)^{(k-2)/k} \quad . && (7.10)
\end{aligned}$$

Substituting (7.8), (7.9) and (7.10) into (7.6) we obtain,

$$\begin{aligned}
&\prod_{i=1}^{k}(x_i+1)\\
&\le P_k \;+\; O\left(\Lambda^{k-1}\right)(P_k)^{(k-1)/k} \;+\; (\Lambda P_k)^{(k-1)/k} \;+\; O\left(\Lambda^{k-1}\right)(P_k)^{(k-2)/k}\\
&\le P_k \;+\; O\left(\Lambda^{k-1}\right)(P_k)^{(k-1)/k} \quad .
\end{aligned}$$

Hence the induction hypothesis holds for k, and by the induction principle the result holds. □

In $D = 3$ dimensions we have the following bound for the surface (v) regardless of whether $M_X(v)$, $M_Y(v)$ and $M_Z(v)$ have bounded ratios.

Lemma 7.3. *For every $M_X(v)$, $M_Y(v)$ and $M_Z(v)$ there is a solution to (7.1) with*

$$\mathit{surface}\,(v) \;\le\; 4\,(M_X(v) + M_Y(v) + M_Z(v)) + O(1)$$

Proof. In what follows $\{i, j, k\} = \{X, Y, Z\}$, and we omit the '$(v)$' from $M_i(v)$, $r_i(v)$, etc. Note that for $D = 3$, problem (7.1) becomes

$$\text{determine } \alpha_i, \alpha_j, \alpha_k \text{ such that } \alpha_i\alpha_j \ge M_k,\ \alpha_i\alpha_k \ge M_j \text{ and } \alpha_j\alpha_k \ge M_i \quad . \tag{7.11}$$

We wish to minimise the surface $(v) = 2\,(\alpha_i\alpha_j + \alpha_i\alpha_k + \alpha_j\alpha_k)$. For each $i \in \{X, Y, Z\}$ the real-valued exact solution to (7.11) is given by

$$(\alpha_i =)\; r_i \;=\; \sqrt{\frac{M_j M_k}{M_i}} \quad .$$

Suppose without loss of generality that $M_i \geq M_j$ and $M_i \geq M_k$. Then $r_i \leq r_j$ and $r_i \leq r_k$. We initially consider three special cases for small values of r_i.

Case 1: $r_i \leq 1$ (i.e., $M_i \geq M_j M_k$).

We set $\alpha_i \leftarrow 1$, $\alpha_j \leftarrow M_k$ and $\alpha_k \leftarrow \lceil M_i/M_k \rceil$. Hence

$$\alpha_i\alpha_j = M_k,\ \alpha_i\alpha_k \geq M_i/M_k \geq M_j \text{ and } \alpha_j\alpha_k \geq M_k(M_i/M_k) = M_i \ \ .$$

So a valid solution to (7.11) is determined. We have the following upper bound.

$$\begin{aligned}
\text{surface}\,(v) &= 2(\alpha_i\alpha_j + \alpha_i\alpha_k + \alpha_j\alpha_k) \\
&= 2(M_k + \lceil M_i/M_k \rceil + M_k \lceil M_i/M_k \rceil) \\
&< 2(M_k + M_i/M_k + 1 + M_k(M_i/M_k + 1)) \\
&= 2(2M_k + M_i/M_k + M_i + 1) \\
&\leq 2(2M_k + 2M_i + 1) \ \ .
\end{aligned}$$

So, in this case the result stands.

Case 2: $1 < r_i \leq \sqrt{2}$ (i.e., $M_j M_k/2 \leq M_i < M_j M_k$).

We set $\alpha_i \leftarrow 1$, $\alpha_j \leftarrow M_k$ and $\alpha_k \leftarrow M_j$. Hence

$$\alpha_i\alpha_j = M_k,\ \alpha_i\alpha_k = M_j \text{ and } \ \alpha_j\alpha_k = M_k M_j > M_i \ \ .$$

So a valid solution to (7.11) is determined. We have the following upper bound.

$$\begin{aligned}
\text{surface}\,(v) &= 2(\alpha_i\alpha_j + \alpha_i\alpha_k + \alpha_j\alpha_k) \\
&= 2(M_k + M_j + M_k M_j) \\
&\leq 2(M_k + M_j + 2M_i) \ \ .
\end{aligned}$$

So, in this case the result stands.

Case 3: $\sqrt{2} < r_i \leq 2$ (i.e., $M_j M_k/4 \leq M_i < M_j M_k/2$).

Set $\alpha_i \leftarrow 2$. Assume without loss of generality that $M_j \leq M_k$, and set $\alpha_j \leftarrow \lceil M_k/2 \rceil$ and $\alpha_k \leftarrow M_j$. Hence

$$\alpha_i\alpha_j = \ 2\,\lceil M_k/2 \rceil \geq M_k,$$

$$\begin{aligned}\alpha_i\alpha_k &= 2M_j > M_j, \text{ and}\\ \alpha_j\alpha_k &= \lceil M_k/2\rceil\, M_j \geq M_jM_k/2 > M_i\ .\end{aligned}$$

So a valid solution to (7.11) is determined. We have the following upper bound.

$$\begin{aligned}\text{surface}\,(v) &= 2(\alpha_i\alpha_j + \alpha_i\alpha_k + \alpha_j\alpha_k)\\ &= 2(2\lceil M_k/2\rceil + 2M_j + \lceil M_k/2\rceil\, M_j)\\ &\leq 2(M_k + 1 + 2M_j + M_jM_k/2 + M_j/2)\\ &\leq 2(M_k + 1 + 2M_j + 2M_i + M_k/2)\\ &= 2(2M_i + 2M_j + 3M_k/2 + 1)\end{aligned}$$

So, in this case the result stands.

Case 4: $r_i > 2$ for every $i \in \{1,2,3\}$.

Set $\alpha_i \leftarrow \lceil r_i \rceil$, $\alpha_j \leftarrow \lceil r_j \rceil$, and $\alpha_k \leftarrow \lceil r_k \rceil$. Obviously this is a valid solution to (7.11) and we have the following upper bound.

$$\begin{aligned}\text{surface}\,(v) &= 2(\alpha_i\alpha_j + \alpha_i\alpha_k + \alpha_j\alpha_k)\\ &< 2((r_i+1)(r_j+1) + (r_i+1)(r_k+1) + (r_j+1)(r_k+1))\\ &= 2((r_ir_j + r_i + r_j + 1) + (r_ir_k + r_i + r_k + 1) + (r_jr_k + r_j + r_k + 1))\\ &= 2((M_k + r_i + r_j) + (M_j + r_i + r_k) + (M_i + r_j + r_k) + 3)\end{aligned}$$

It is well-known that $x + y \leq xy$ for any two real numbers $x, y \geq 2$. So $r_i + r_j \leq r_ir_j = M_k$, $r_i + r_k \leq r_ir_k = M_j$ and $r_j + r_k \leq r_jr_k = M_i$. Hence

$$\text{surface}\,(v) \leq 2(2M_i + 2M_j + 2M_k + 3)\ ,$$

and, in this case the result stands. □

7.1.2 Determining Port Assignments

Given a general position vertex layout and arc-routing, we now describe how to assign ports on a vertex v to the arcs incident to v such that an arc $\overrightarrow{vw} \in A(v)[i]$ is assigned an i-port on v pointing toward w. Suppose the k^{th} *segment of an arc* $\overrightarrow{vw}$, $1 \leq k \leq D$, refers to the k^{th} segment of the edge route vw starting at v.

We firstly assign ports to arcs so that no two edges routed on the same face can intersect. This algorithm improves on the algorithm of Biedl [27] for $D = 3$, in that we potentially use all the ports on a face. This is possible due to the second stage of our port assignment method which eliminates all subsequent edge route crossings.

We now describe how to assign the ports on the (i^+)-face of a vertex v (for some dimension i), to the arcs in $A_G(v)\langle i^+\rangle[i]$; i.e. arcs $\overrightarrow{vw}$ coloured i with w in direction i^+ from v. Assigning ports on the (i^-)-face to the arcs in $A_G(v)\langle i^-\rangle[i]$ is analogous.

We group the arcs in $A_G(v)\langle i^+\rangle[i]$ according to the direction of their second segment, which by the routing of edges described in Algorithm 7.1 D-Dimensional General Position Box-Drawing is one of $(i+1)^+$, $(i+1)^-$, $(i-1)^+$ and $(i-1)^-$. For these cases we say an arc in $A_G(v)\langle i\rangle[i]$ is either an *up*, *down*, *right* or *left* arc, respectively. Ports are assigned so that the ports 'underneath' the second segment of an arc are assigned to arcs within the same grouping.

Firstly, as illustrated in Figure 7.2(a), we partition the face into two regions, the first with enough ports for the Down and Right arcs, and the second with enough ports for the Up and Left arcs. Within the first region we determine the ports to be used by the Right arcs by numbering the ports starting at the top-right corner in a right-to-left row-by-row fashion, as in Figure 7.2(a). Similarly, we determine the ports of the second region to be used by the Left arcs by numbering the ports starting at the bottom-left corner of the second region in a left-to-right row-by-row fashion. The remaining ports in the first region are assigned to the Down arcs and the remaining ports in the second region are assigned to the Up arcs, as in Figure 7.2(b).

We assign ports to the arcs in each grouping in turn, and within a grouping we assign ports to the arcs in increasing order of the length of the first segment of the arc. Since our graphs are simple this length is unique. For each arc we choose an unused port within its grouping so that the second segment of the produced edge route has minimum possible length, as in Figure 7.2(b). Clearly no two edges routed on the same face can intersect.

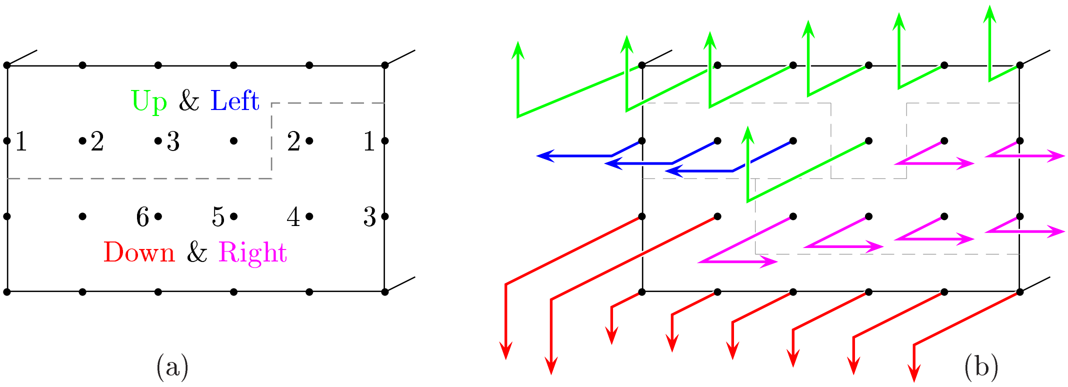


Figure 7.2: Determining port assignments on a face.

7.1.3 Removing Edge Crossings

We now show how to remove edge route crossings in general position D-dimensional orthogonal box-drawings ($D \geq 3$). The method is a generalisation of the crossing elimination rule for 3-D orthogonal point-drawings shown in Figure 5.11.

Suppose the edge routes vw and xy intersect at some grid-point, and the intersecting segments of vw and xy are a- and b-segments, respectively (for some dimensions a and b). Label the endpoints of these segments r, s, p and q as shown in Figure 7.3(a).

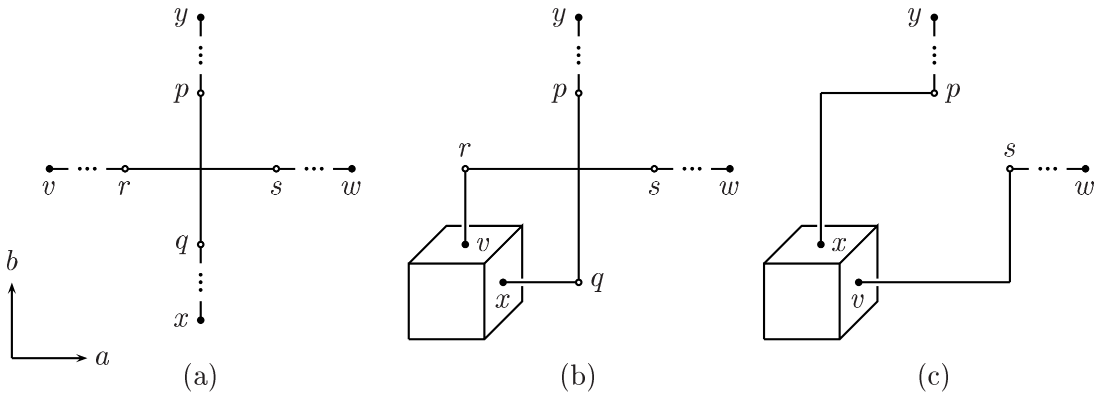


Figure 7.3: Removing edge crossings in general position.

In what follows we describe a sequence of segments contained in an edge route as a *path*. Since the graph is simple, we can assume without loss of generality that $y \neq w$. Therefore port$(\overrightarrow{wv})$ and port$(\overrightarrow{yx})$ differ in every coordinate. It follows that for every dimension i except for a and b, there is an i-segment on the paths from w to s, and

y to p. This implies there is at most one segment on the paths r to v, and if there is such a segment then it is a b-segment. Similarly, there is at most one segment on the path q to x, and if there is such a segment then it is an a-segment. This implies that v and x are coplanar, and since the vertices are in general position, $v = x$. By the construction used in the previous section, edge routes are assigned unique ports on a face, and no two edge routes on the same face can intersect. So the paths from r to v $(= x)$ and from q to x $(= v)$ have exactly one segment, and the edge crossing occurs between the second segments of arcs incident to a common vertex, as shown in Figure 7.3(b). Each such crossing can be removed by swapping the ports assigned to these arcs, and rerouting the corresponding edge routes as shown in Figure 7.3(c). We have the following algorithm for removing edge route crossings in general position orthogonal drawings.

Algorithm 7.2. Box-Drawing Remove Edge Crossings

Input: D-dimensional general position orthogonal drawing of a graph G (possibly with crossings)

Output: D-dimensional general position orthogonal drawing of G (without crossings).

$A \leftarrow A(G)$

while $A \neq \emptyset$ **do**

  Choose $\overrightarrow{vw} \in A$.

  Set $A \leftarrow A \setminus \{\overrightarrow{vw}\}$.

  if $\overrightarrow{vw}$ intersects some other arc $\overrightarrow{vu}$ **then**

    Swap the ports at v assigned to $\overrightarrow{vw}$ and $\overrightarrow{vu}$.

    Reroute the edge routes vu and vw as described above.

    Set $A \leftarrow A \cup \{\overrightarrow{vu}, \overrightarrow{vw}\}$.

    if $D = 3$ **then**

      Set $A \leftarrow A \cup \{\overrightarrow{uv}, \overrightarrow{wv}\}$.

    end-if

  end-if

end-while

Lemma 7.4. *The algorithm* Box-Drawing Remove Edge Crossings *removes all crossings from the given general position orthogonal box-drawing in $O(mn\Delta)$ time.*

Proof. We shall prove that at all times the set A contains all arcs which possibly intersect some other arc. Initially this is true since $A = A(G)$. As proved above, an arc $\overrightarrow{vw}$ can only intersect another arc incident to v. Hence, if $\overrightarrow{vw}$ does not intersect some other arc $\overrightarrow{vu}$, then $\overrightarrow{vw}$ does not intersect any arc, and $\overrightarrow{vw}$ can be removed from A.

Suppose that $\overrightarrow{vw}$ intersects some other arc $\overrightarrow{vu}$. After swapping the ports assigned to $\overrightarrow{vu}$ and $\overrightarrow{vw}$ all new edge crossings must involve $\overrightarrow{vu}$ or $\overrightarrow{vw}$ (or $\overrightarrow{uv}$ or $\overrightarrow{wv}$ if $D = 3$). By adding $\overrightarrow{vu}$ and $\overrightarrow{vw}$ (and $\overrightarrow{uv}$ and $\overrightarrow{wv}$ if $D = 3$) to A for re-checking, we maintain the condition that A contains all arcs which possibly intersect some other arc. The algorithm continues until $A = \emptyset$, at which point the drawing must be crossing-free.

For an arc $\overrightarrow{vw}$ whose second segment is parallel to the i-axis, let $l(\overrightarrow{vw}) = |p - q|$, where $(u_1, u_2, \ldots, u_n)$ is the i-ordering of the vertices and $v = u_p$ and $w = u_q$.

Now $l(\overrightarrow{vw}) = O(n)$, so $\sum_{\overrightarrow{vw}} l(\overrightarrow{vw}) = O(mn)$. Each port swap between arcs $\overrightarrow{vw}$ and $\overrightarrow{vu}$ reduces $l(\overrightarrow{vw}) + l(\overrightarrow{vu})$. Hence there will be $O(mn)$ port swaps. Therefore $O(mn)$ arcs are added to A, so $O(mn)$ arcs are checked for crossings. To test if an arc intersects some other arc takes $O(\Delta)$ time, so the algorithm takes $O(mn\Delta)$ time. □

The effect of a number of port swaps, all in the same plane, is shown in Figure 7.4.

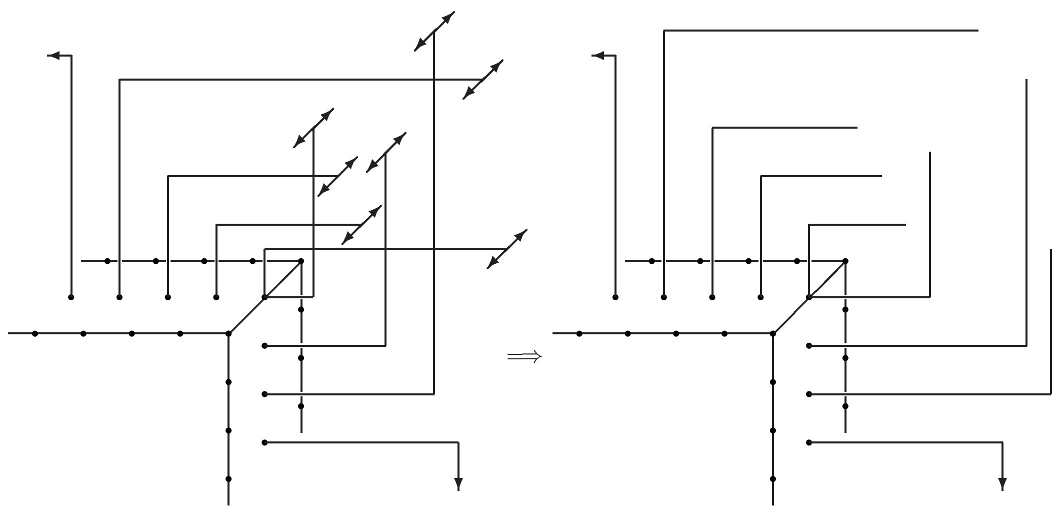


Figure 7.4: Rerouting crossing edge routes.

Based on the above methods for port assignment and the elimination of edge crossings we have the following result.

Lemma 7.5. *Given a D-dimensional general position vertex layout, D-dimensional general position arc-routing and the size of each vertex of a graph G, if for every vertex $v \in V(G)$ the surface$_i(v) \geq M_i(v)$ for each i, $1 \leq i \leq D$, then a crossing-free assignment of the ports on each vertex v to the arcs $A_G(v)$ can be determined in $O(mn\Delta)$ time.* □

7.1.4 Upper Bounds

We now establish upper bounds for the surface and volume of the bounding box of a general position D-dimensional orthogonal box-drawing in terms of the size and shape of the vertices. For each vertex v we denote the arithmetic, geometric and harmonic means of $\alpha_1(v), \alpha_2(v), \ldots, \alpha_D(v)$ by $\alpha^+(v)$, $\alpha^*(v)$ and $\alpha^-(v)$ respectively; i.e.,

$$\alpha^+(v) = \frac{1}{D} \sum_{1 \leq i \leq D} \alpha_i(v), \quad \alpha^*(v) = \left(\prod_{1 \leq i \leq D} \alpha_i(v) \right)^{1/D}, \quad \alpha^-(v) = D \left(\sum_{1 \leq i \leq D} \frac{1}{\alpha_i(v)} \right)^{-1}.$$

Obviously volume $(v) = \alpha^*(v)^D$, and also,

$$\begin{aligned} \text{surface}\,(v) &= 2 \sum_{1 \leq i \leq D} \prod_{\substack{1 \leq j \leq D \\ j \neq i}} \alpha_j(v) \\ &= 2 \sum_{1 \leq i \leq D} \frac{\alpha^*(v)^D}{\alpha_i(v)} \\ &= 2\alpha^*(v)^D \sum_{1 \leq i \leq D} \frac{1}{\alpha_i(v)} \\ &= \frac{2D\alpha^*(v)^D}{\alpha^-(v)} . \end{aligned} \tag{7.12}$$

The arithmetic, geometric and harmonic means of the dimensions of the bounding box are denoted by β^+, β^* and β^- respectively. As in (7.12) we have,

$$\text{surface}\,(\text{ bounding box }) = \frac{2D(\beta^*)^D}{\beta^-}. \tag{7.13}$$

It is well-known that, of the D-dimensional hyperboxes with fixed sum of side lengths, the D-dimensional hypercube has maximum volume and maximum surface area (see for example Kazarinoff [126]). Given a D-dimensional vertex v, consider the hypercube C with the same sum of side lengths as v; i.e., C has side length $\alpha^+(v)$. We define the *surface aspect*(v) to be the ratio of the surface(C) to the surface(v), and the *volume aspect*(v) to be the ratio of the volume(C) to the volume(v). Clearly surface aspect and volume aspect are both at least one. By (7.12) we have,

$$\text{surface aspect}\,(v) = \frac{\text{surface}\,(C)}{\text{surface}\,(v)} = \frac{\alpha^+(v)^{D-1}\alpha^-(v)}{\alpha^*(v)^D}\ , \tag{7.14}$$

$$\text{volume aspect}\,(v) = \frac{\text{volume}\,(C)}{\text{volume}\,(v)} = \left(\frac{\alpha^+(v)}{\alpha^*(v)}\right)^D\ . \tag{7.15}$$

Lemma 7.6. *For a general position D-dimensional orthogonal box-drawing,*

$$\text{surface}\,(\text{ bounding box }) \ \leq\ n^{D-2}\sum_v \text{surface aspect}\,(v) \times \text{surface}\,(v)$$

Proof. Since the surface aspect of the bounding box is at least one, by (7.14) applied to the bounding box,

$$(\beta^*)^D/\beta^- \leq (\beta^+)^{D-1}\ ,$$

so by (7.13),

$$\text{surface}\,(\text{ bounding box }) \ =\ \frac{2D(\beta^*)^D}{\beta^-} \ \leq\ 2D\left(\beta^+\right)^{D-1}\ .$$

The average side 'length' of the bounding box is

$$\beta^+ = \frac{1}{D}\sum_i\sum_v \alpha_i(v)\ .$$

So the surface of the bounding box is

$$2D\left(\frac{1}{D}\sum_i\sum_v \alpha_i(v)\right)^{D-1} \ =\ 2D\left(\sum_v\frac{1}{D}\sum_i \alpha_i(v)\right)^{D-1} \ =\ 2D\left(\sum_v \alpha^+(v)\right)^{D-1}\ .$$

By the Cauchy-Schwarz inequality,

$$2D\left(\sum_v \alpha^+(v)\right)^{D-1} \ \leq\ 2Dn^{D-2}\sum_v \alpha^+(v)^{D-1}$$

$$= n^{D-2} \sum_v \left(\frac{\alpha^+(v)^{D-1} \alpha^-(v)}{\alpha^*(v)^D} \right) \times \left(\frac{2D\alpha^*(v)^D}{\alpha^-(v)} \right)$$
$$= n^{D-2} \sum_v \text{surface aspect}\,(v) \times \text{surface}\,(v) \quad . \qquad \square$$

It is easily seen that this bound is tight for $D = 2$, for same-sized hypercube drawings and in the case of $n = D$ pairwise perpendicular lines. The proof of the following bound on the volume of the bounding box is similar to that of Lemma 7.6.

Lemma 7.7. *For a general position D-dimensional orthogonal box-drawing,*

$$\text{volume}\,(\text{ bounding box }) \ \le\ n^{D-1} \sum_v \text{volume aspect}\,(v) \times \left(\frac{\text{surface}\,(v)}{2D} \right)^{D/(D-1)}$$

Proof. The volume of the bounding box is

$$\begin{aligned}
(\beta^*)^D \le\ & (\beta^+)^D \\
=\ & \left(\sum_{1\le i\le D} \sum_v \frac{\alpha_i(v)}{D} \right)^D \\
=\ & \left(\sum_v \sum_{1\le i\le D} \frac{\alpha_i(v)}{D} \right)^D \\
=\ & \left(\sum_v \alpha^+(v) \right)^D \quad .
\end{aligned}$$

By the Cauchy-Schwarz inequality,

$$\begin{aligned}
\left(\sum_v \alpha^+(v) \right)^D & \le n^{D-1} \sum_v \alpha^+(v)^D \\
& = n^{D-1} \sum_v \left(\frac{\alpha^+(v)}{\alpha^*(v)} \right)^D \alpha^*(v)^D \\
& = n^{D-1} \sum_v \text{volume aspect}\,(v) \times \text{volume}\,(v)
\end{aligned}$$

Of the D-dimensional hyperboxes with fixed surface S, the cube with side length $(S/2D)^{1/(D-1)}$ has maximum volume [126]. So

$$\text{volume}\,(\text{ bounding box }) \le n^{D-1} \sum_v \text{volume aspect}\,(v) \times \left(\frac{\text{surface}\,(v)}{2D} \right)^{D/(D-1)} . \qquad \square$$

Corollary 7.1. *For a general position D-dimensional orthogonal box-drawing,*

$$\text{surface}\,(\text{ bounding box }) \le n^{D-2} \sum_v \text{aspect ratio}\,(v) \times \text{surface}\,(v)$$

$$\text{volume}\,(\text{ bounding box }) \le n^{D-1} \sum_v \text{aspect ratio}\,(v) \times \left(\frac{\text{surface}\,(v)}{2D}\right)^{D/(D-1)} .$$

Proof. Of the D-dimensional hyperboxes with fixed sum of side lengths, the line has minimum surface and minimum volume, so the surface aspect and volume aspect of a line is maximum (for the D-dimensional hyperboxes with fixed sum of side lengths). The surface aspect and volume aspect of a line are no more than its aspect ratio. The result follows from Lemma 7.6 and Lemma 7.7. □

The next result will be used to establish a bound on the bounding box volume for the orthogonal graph drawing algorithms presented in Sections 7.2 and 7.3.

Theorem 7.2. *A d-degree-restricted general position D-dimensional orthogonal box-drawing with each vertex having aspect ratio at most a has*

$$\text{volume}\,(\text{ bounding box }) \le a\left(n^{D-2}\left(\frac{dm}{D} + o\,(m)\right)\right)^{D/(D-1)}$$

Proof. By Corollary 7.1,

$$\text{volume}\,(\text{ bounding box }) \;\le\; n^{D-1} \sum_v a \left(\frac{\text{surface}\,(v)}{2D}\right)^{D/(D-1)} .$$

By the Cauchy-Schwarz inequality,

$$\begin{aligned}
\text{volume}\,(\text{ bounding box }) &\le a\,n^{D-1}\left(n^{(D-1)/D-1} \sum_v \frac{\text{surface}\,(v)}{2D}\right)^{D/(D-1)} \\
&= a\,n^{D-1} n^{-1/(D-1)} \left(\sum_v \frac{\text{surface}\,(v)}{2D}\right)^{D/(D-1)} \\
&= a\,n^{D(D-2)/(D-1)} \left(\sum_v \frac{\text{surface}\,(v)}{2D}\right)^{D/(D-1)} \\
&= a\left(n^{D-2} \sum_v \frac{\text{surface}\,(v)}{2D}\right)^{D/(D-1)}
\end{aligned}$$

Since the drawing is d-degree-restricted,

$$\text{volume}\,(\text{ bounding box }) \le a\left(n^{D-2} \sum_v \frac{d \cdot \deg(v) + o\,(\deg(v))}{2D}\right)^{D/(D-1)}$$

$$= a \left(n^{D-2} \left(\frac{dm}{D} + o(m) \right) \right)^{D/(D-1)} . \qquad \square$$

7.2 Layout-Based Algorithms

In this section we describe our layout-based approach for determining general position D-dimensional orthogonal drawings, for some constant $D \geq 3$. In Section 7.2.1 we present an algorithm for determining an arc-routing given an arbitrary general position vertex layout. We derive algorithms using fixed, balanced and diagonal vertex layouts in Sections 7.2.2, 7.2.3 and 7.2.4.

7.2.1 Arc-Routing Algorithm

We now present an algorithm for determining an arc-routing of $A(G)$ with respect to a given general position vertex layout of a graph G. To represent the colouring of $A(G)$ we vertex-colour a graph H with vertex set $V(H) = A(G)$. We represent a D-dimensional orthant by the corresponding set of D pairwise non-opposite directions. For a given vertex v and direction d, the set of orthants $\{T : d \in T\}$ in direction d from v is denoted $\Phi_d^D(v)$. We denote the set of arcs $\overrightarrow{vw}$ at a vertex v with w in orthant T by $A_G(v)\langle T\rangle$; i.e.,

$$A_G(v)\langle T\rangle = \bigcap_{d \in T} A_G(v)\langle d\rangle \quad .$$

Algorithm 7.3. D-Dimensional General Position Arc-Routing

Input: • graph G

• D-dimensional general position vertex layout of $V(G)$

Output: D-dimensional general position arc-routing of $A(G)$

---

1. For each edge $vw \in E(G)$, insert the edge $\{\overrightarrow{vw}, \overrightarrow{wv}\}$ to $E(H)$ (called an *'r'-edge*).

2. For each vertex $v \in V(G)$ and for each orthant T relative to v,

(a) Partition the arcs in $A_G(v)\langle T\rangle$ into sets $Q_1, Q_2, \ldots, Q_k$, so that $|Q_j| = D$, $1 \leq j < k$ (see Figure 7.5).

(b) Add a clique (called *'c'-edges*) to $E(H)$ between the vertices of H corresponding to the arcs in Q_j, $1 \leq j \leq k$.

3. Determine a D-colouring of $A(G)$ from a vertex-colouring of H with D colours.

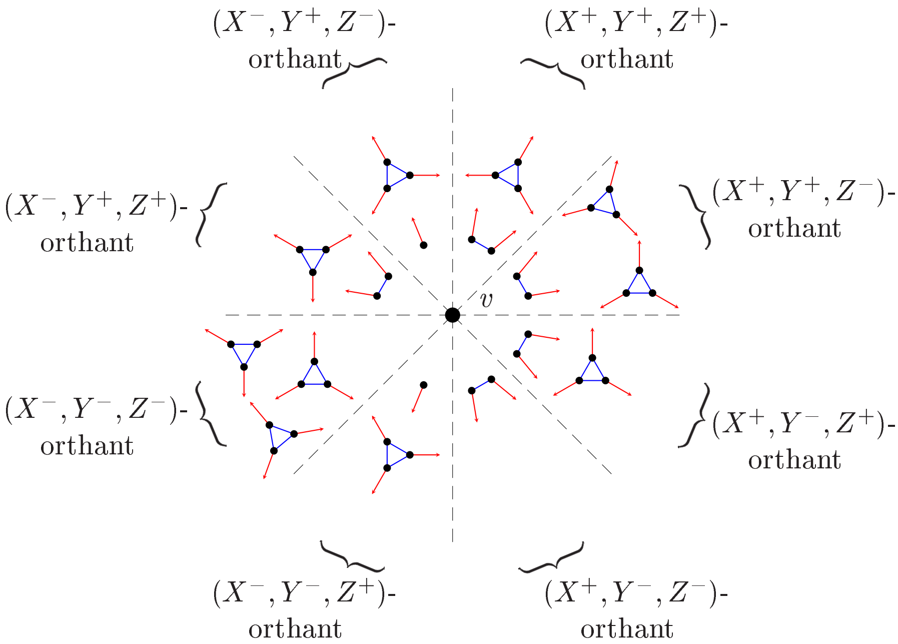


Figure 7.5: Partitioning of $A_G(v)$ and construction of H for $D = 3$.

Lemma 7.8. *The algorithm* D-DIMENSIONAL GENERAL POSITION ARC-ROUTING *determines an arc-routing of* $A(G)$ *in* $O(D(m+n))$ *time such that for each vertex* $v \in V(G)$,

$$2\sum_i M_i(v) \;\leq\; \deg(v) + c(v) + (D-1)2^D \;.$$

Proof. A vertex of H is incident with one 'r' edge and at most $D-1$ 'c' edges. So the maximum degree $\Delta(H) \leq D$, and since the complete graph $K_{D+1} \not\subseteq H$, by Brooks' Theorem [47], H is vertex D-colourable. The proof of Brook's Theorem due to Lovász [147] and simplified by Bryant [49] describes an algorithm for finding a vertex-colouring of H with at most $\Delta(H)$ colours in $O(|E(H)|) = O(Dm)$ time. The vertex-colouring of H determines a D-dimensional routing of $A(G)$. Since $\{\overrightarrow{vw}, \overrightarrow{wv}\} \in E(H)$, reversal arcs are coloured differently, so the routing is an arc-routing.

For each orthant T relative to a vertex v and in each partition of $A_G(v)\langle T\rangle$, there is at most one arc $\overrightarrow{vw}$ coloured i, $1 \le i \le D$. Therefore, for each dimension i, $1 \le i \le D$, we have the following bounds on the number of arcs $\overrightarrow{vw}$ coloured i with w in direction i^+ from v.

$$\sum_{T\in\Phi_i^D(v)} \left\lfloor \frac{|A_G(v)\langle T\rangle\,|}{D} \right\rfloor \;\le\; \left|A_G(v)\langle i\rangle[i]\,\right| \;\le\; \sum_{T\in\Phi_i^D(v)} \left\lceil \frac{|A_G(v)\langle T\rangle\,|}{D} \right\rceil \quad .$$

So,

$$\frac{1}{D}\left(\left(\sum_{T\in\Phi_i^D(v)} \left|A_G(v)\langle T\rangle\,\right|\right) - (D-1)|\Phi_i^D(v)|\right)$$
$$\le \left|A_G(v)\langle i\rangle[i]\,\right| \le$$
$$\frac{1}{D}\left(\left(\sum_{T\in\Phi_i^D(v)} \left|A_G(v)\langle T\rangle\,\right|\right) + (D-1)|\Phi_i^D(v)|\right) \quad .$$

It follows that

$$\frac{1}{D}\left(s_i(v) - (D-1)2^{D-1}\right) \;\le \left|A_G(v)\langle i\rangle[i]\,\right| \le\; \frac{1}{D}\left(s_i(v) + (D-1)2^{D-1}\right) \quad .$$

Similarly,

$$\frac{1}{D}\left(p_i(v) - (D-1)2^{D-1}\right) \;\le\; \left|A_G(v)\langle i^-\rangle[i]\,\right| \;\le\; \frac{1}{D}\left(p_i(v) + (D-1)2^{D-1}\right) \quad .$$

Since $M_i(v) = \max\{|A_G(v)\langle i\rangle[i]\,|, |A_G(v)\langle i^-\rangle[i]\,|\}$,

$$\frac{1}{D}\left(\max\{s_i(v), p_i(v)\} - (D-1)2^{D-1}\right)$$
$$\le\; M_i(v) \le$$
$$\frac{1}{D}\left(\max\{s_i(v), p_i(v)\} + (D-1)2^{D-1}\right) \quad .$$

By (4.1),

$$\frac{1}{D}\left(\frac{1}{2}\left(\deg(v) + c_i(v)\right) - (D-1)2^{D-1}\right)$$

$$\leq \; M_i(v) \leq$$
$$\frac{1}{D}\left(\frac{1}{2}\left(\deg(v)+c_i(v)\right)+(D-1)2^{D-1}\right) \quad . \qquad (7.16)$$

Summing over all dimensions, we obtain,

$$\begin{aligned}
\sum_i M_i(v) &\leq \sum_i \frac{1}{D}\left(\frac{1}{2}\left(\deg(v)+c_i(v)\right) \;+\; (D-1)2^{D-1}\right) \\
2\sum_i M_i(v) &\leq \sum_i \frac{1}{D}\left(\deg(v)+c_i(v) \;+\; (D-1)2^{D}\right) \\
2\sum_i M_i(v) &\leq \deg(v)+c(v)+(D-1)2^{D} \quad . \qquad \square
\end{aligned}$$

7.2.2 Fixed Vertex Layout Drawings

We now derive an algorithm for a fixed general position vertex layout.

Algorithm 7.4. FIXED GENERAL POSITION D-DIMENSIONAL BOX-DRAWING

Input: • graph G.
• D-dimensional general position vertex layout of $V(G)$.

Output: layout-preserving D-dimensional orthogonal box-drawing of G.

---

1. Determine an arc-routing with Algorithm 7.3 D-DIMENSIONAL GENERAL POSITION ARC-ROUTING.
2. Apply Algorithm 7.1 D-DIMENSIONAL GENERAL POSITION BOX-DRAWING.

Theorem 7.3. *The algorithm* FIXED GENERAL POSITION D-DIMENSIONAL BOX-DRAWING *determines a layout-preserving D-dimensional orthogonal box-drawing of G in $O(mn\Delta)$ time such that:*

- *Each edge route has $D-1$ bends.*
- *Each vertex is 2-degree-restricted*
- *The aspect ratio of each vertex v is at most $2+o(\deg(v))$.*

- *The bounding box volume is* $O\Big(\big(n^{D-2}m\big)^{D/(D-1)}\Big)$.

Proof. By (7.16) and since $c_i(v) \le \deg(v)$,

$$\frac{1}{D}\left(\frac{1}{2}\deg(v) - (D-1)2^{D-1}\right) \;\le\; M_i(v) \le \;\frac{1}{D}\left(\deg(v) + (D-1)2^{D-1}\right) \quad .$$

So for all i, j, $1 \le i, j, \le D$,

$$\begin{aligned}
\frac{M_i(v)}{M_j(v)} &\le \frac{\deg(v) + (D-1)2^{D-1}}{\frac{1}{2}\deg(v) - (D-1)2^{D-1}} \\
&\le \frac{2\deg(v) + (D-1)2^{D}}{\deg(v) - (D-1)2^{D}} \\
&\le \frac{2\left(\deg(v) - (D-1)2^{D}\right) \;+\; 3(D-1)2^{D}}{\deg(v) - (D-1)2^{D}} \\
&\le\; 2 + \frac{3(D-1)2^{D}}{\deg(v) - (D-1)2^{D}} \quad .
\end{aligned}$$

It follows from Theorem 7.1 with

$$f(v) \;=\; 2 + \frac{3(D-1)2^{D}}{\deg(v) - (D-1)2^{D}}$$

that

$$\text{surface}\,(v) \;\le\; 2\sum_i M_i \;+\; \left(2 + \frac{3(D-1)2^{D}}{\deg(v) - (D-1)2^{D}}\right)^{D-2} \left(\sum_i M_i\right)^{(D-2)/(D-1)} \quad .$$

For constant D we have

$$\text{surface}\,(v) \;\le\; 2\sum_i M_i \;+\; \Big(2 + O\big(\deg(v)^{-1}\big)\Big)^{D-2} \left(\sum_i M_i\right)^{(D-2)/(D-1)} \quad . \tag{7.17}$$

By Lemma 7.8 and since $c(v) \le \deg(v)$ with D a constant we have

$$2\sum_i M_i(v) \;\le\; 2\deg(v) + O(1) \quad .$$

Hence,

$$\begin{aligned}
\text{surface}\,(v) &\le\; 2\deg(v) + O(1)\;\; \Big(2 + O\big(\deg(v)^{-1}\big)\Big)^{D-2} \Big(\deg(v) + O(1)\Big)^{(D-2)/(D-1)} \\
&\le\; 2\deg(v) + O\Big(\deg(v)^{(D-2)/(D-1)}\Big) \\
&\le\; 2\deg(v) + o\left(\deg(v)\right) \quad .
\end{aligned}$$

So v is 2-degree-restricted. Suppose $\alpha_i(v)$ and $\alpha_j(v)$ are the maximum and minimum of $\{\alpha_1(v), \alpha_2(v), \ldots, \alpha_D(v)\}$, respectively. Then

$$\begin{aligned}
\text{aspect ratio}\,(v) &= \alpha_i(v)/\alpha_j(v) \\
&= \left(\alpha_i(v)\prod_{k\neq i,j}\alpha_k(v)\right) \Big/ \left(\alpha_j(v)\prod_{k\neq i,j}\alpha_k(v)\right) \\
&= \left(\prod_{k\neq j}\alpha_k(v)\right) \Big/ \left(\prod_{k\neq i}\alpha_k(v)\right) \\
&= \frac{\text{surface}_j(v)}{\text{surface}_i(v)} \ .
\end{aligned}$$

Now,

$$\begin{aligned}
\text{surface}_j(v) &\leq M_j(v) \ + \ O\big(f(v)^{D-2}\big)\, M_j(v)^{(D-2)/(D-1)} \\
&\leq M_j(v) \ + \ \big(2 + O\big(\deg(v)^{-1}\big)\big)^{D-2} M_j(v)^{(D-2)/(D-1)} \ .
\end{aligned}$$

Since $M_j(v) \leq \frac{1}{D}\left(\deg(v) + (D-1)2^{D-1}\right)$, for constant D we have

$$\text{surface}_j(v) \leq \frac{1}{D}\deg(v) + O\Big(\deg(v)^{(D-2)/(D-1)}\Big) \ .$$

Now $\text{surface}_i(v) \ \geq \ \deg(v)/2D$, so

$$\begin{aligned}
\text{aspect ratio}\,(v) &\leq \frac{\frac{1}{D}\deg(v) + O\big(\deg(v)^{(D-2)/(D-1)}\big)}{\frac{1}{2D}\deg(v)} \\
&\leq \frac{2\deg(v) + O\big(\deg(v)^{(D-2)/(D-1)}\big)}{\deg(v)} \\
&\leq 2 + o\,(\deg(v)) \ .
\end{aligned}$$

Hence the aspect ratio of v is $2 + o\,(\deg(v))$. The volume bound follows immediately from Theorem 7.2.

Applying Algorithm D-Dimensional General Position Box-Drawing, which takes $O(mn\Delta)$ time, is the most time-consuming step of the algorithm. So Algorithm Fixed General Position D-Dimensional Box-Drawing takes $O(mn\Delta)$ time. □

7.2.3 Balanced Vertex Layout Drawings

We initially show that the complete graph provides a lower bound for the problem D-DIMENSIONAL GENERAL POSITION VERTEX LAYOUT.

Lemma 7.9. *In any D-dimensional general position vertex layout of K_n there is a vertex v with*

$$c(v) \geq \frac{\deg(v)}{2} \quad .$$

Proof. By Lemma 4.1, the total cost of a D-dimensional layout of K_n is

$$\sum_v \frac{1}{D} \sum_i c_i(v) \;=\; \frac{1}{D} \sum_i \sum_v c_i(v) \;=\; \frac{1}{D} \sum_i \lfloor n^2/2 \rfloor \;=\; \left\lfloor \frac{n^2}{2} \right\rfloor \quad .$$

So even if each vertex has the same cost, there exists a vertex v with

$$\begin{aligned} c(v) &\geq \frac{\lfloor n^2/2 \rfloor}{n} \\ &= \begin{cases} n/2, & \text{if } n \text{ is even;} \\ (n^2-1)/2n, & \text{if } n \text{ is odd.} \end{cases} \\ &> \frac{n-1}{2} \\ &= \frac{\deg(v)}{2} \quad . \end{aligned}$$

□

The following algorithm provides a tight upper bound for the problem D-DIMENSIONAL GENERAL POSITION VERTEX LAYOUT. It is based on the algorithm for determining balanced 2-D general position vertex layouts presented in Chapter 6.

Algorithm 7.5. BALANCED D-DIMENSIONAL GENERAL POSITION VERTEX LAYOUT

Input: graph G and positive integer D.

Output: D-dimensional general position vertex layout of G.

---

1. Determine a 2-D general position vertex layout, represented by X- and Y- vertex orderings, with Algorithm 6.4 BALANCED 2-D GENERAL POSITION VERTEX LAYOUT.

2. Set the i-ordering of the vertex layout to be the X-ordering for odd i, $1 \leq i \leq D$.

3. Set the i-ordering of the vertex layout to be the Y-ordering for even i, $1 \leq i \leq D$.

Theorem 7.4. *The algorithm* BALANCED D-DIMENSIONAL GENERAL POSITION VERTEX LAYOUT *determines a D-dimensional general position vertex layout of G in $O(D(m+n))$ time such that for each vertex v,*

$$c(v) \leq 1 + \frac{\lceil D/2 \rceil}{D} \deg(v) \ .$$

Proof. For each vertex v and each ordering i, $1 \leq i \leq D$, the cost $c_i(v) \leq s(v)+1$ if i is odd, and $c_i(v) \leq p(v)+1$ if i is even. So

$$\begin{aligned} c(v) &\leq \frac{1}{D}\left(\left\lceil \frac{D}{2} \right\rceil (s(v)+1) + \left\lfloor \frac{D}{2} \right\rfloor (p(v)+1)\right) \\ &= \frac{1}{D}\left(\left\lfloor \frac{D}{2} \right\rfloor \deg(v) + \left(\left\lceil \frac{D}{2} \right\rceil - \left\lfloor \frac{D}{2} \right\rfloor\right) s(v) + D\right) \\ &\leq \frac{1}{D}\left(\left\lceil \frac{D}{2} \right\rceil \deg(v) + D\right) \\ &= 1 + \frac{\lceil D/2 \rceil}{D} \deg(v) \ . \end{aligned}$$

By Theorem 6.2, a balanced 2-D vertex layout can be determined in $O(m+n)$ time, so algorithm BALANCED D-DIMENSIONAL GENERAL POSITION VERTEX LAYOUT takes $O(D(m+n))$ time. □

For a D-dimensional general position vertex layout of K_n the upper bound provided by Theorem 7.4 is

$$c(v) \ \leq \ 1 + \frac{\lceil D/2 \rceil}{D} \deg(v) \ = \ \begin{cases} (n+1)/2, & \text{if } D \text{ is even;} \\ 1 + (n-1)(D+1)/2D, & \text{if } D \text{ is odd.} \end{cases}$$

For even D, the difference between this upper bound and the lower bound of Lemma 7.9 is at most 1. For odd D, the difference between the upper and lower bounds is at least $n/2D$. It is an open problem to establish tight bounds on $\max_v c(v)$ in the case of odd D. We now derive results for general position orthogonal graph drawing based on a balanced vertex layout.

Algorithm 7.6. BALANCED GENERAL POSITION D-DIMENSIONAL BOX-DRAWING

Input: graph G.

Output: D-dimensional orthogonal box-drawing of G.

---

1. Determine a general position vertex layout with the BALANCED D-DIMENSIONAL GENERAL POSITION VERTEX LAYOUT algorithm.

2. Determine an arc-routing with Algorithm 7.3 D-DIMENSIONAL GENERAL POSITION ARC-ROUTING.

3. Apply Algorithm 7.1 D-DIMENSIONAL GENERAL POSITION BOX-DRAWING.

Theorem 7.5. *The algorithm* BALANCED GENERAL POSITION D-DIMENSIONAL BOX-DRAWING *determines a D-dimensional orthogonal box-drawing of G in $O(mn\Delta)$ time such that:*

- *Each edge route has $D-1$ bends.*
- *Each vertex is $3/2$-degree-restricted if D is even, and $(3/2+1/2D)$-degree-restricted if D is odd.*
- *The aspect ratio of each vertex v is $2+o\left(\deg(v)\right)$.*
- *The bounding box volume is $O\Big(\left(n^{D-2}m\right)^{D/(D-1)}\Big)$.*

Proof. By Lemma 7.8, and since in a D-dimensional balanced vertex layout (Theorem 7.4), for every vertex v, $c(v) \leq 1+\frac{\lceil D/2\rceil}{D}\deg(v)$, it follows that

$$2\sum_i M_i(v) \;\leq\; \left(1+\frac{\lceil D/2\rceil}{D}\right)\deg(v)+O(1) \quad .$$

By (7.17),

$$\mathrm{surface}\,(v) \;\leq\; 2\sum_i M_i \;+\; \left(2+\frac{O(1)}{\deg(v)}\right)^{2(D-2)}\left(\sum_i M_i\right)^{(D-2)/(D-1)} \quad .$$

So surface (v) is at most

$$\begin{aligned}
&\left(1+\frac{\lceil D/2\rceil}{D}\right)\deg(v) \;+\; \left(2+\frac{O(1)}{\deg(v)}\right)^{2(D-2)}\left(\left(1+\frac{\lceil D/2\rceil}{D}\right)\deg(v)\right)^{(D-2)/(D-1)}\\
\leq\; &\left(1+\frac{\lceil D/2\rceil}{D}\right)\deg(v) \;+\; O\Big(\deg(v)^{(D-2)/(D-1)}\Big)\\
=\; &\left(1+\frac{\lceil D/2\rceil}{D}\right)\deg(v) \;+\; o\left(\deg(v)\right) \quad .
\end{aligned}$$

So v is $3/2$-degree-restricted if D is even, and $3/2+1/2D$-degree-restricted if D is odd. The bounding box volume, aspect ratio and time bounds follow from Theorem 7.3. ☐

7.2.4 Diagonal Vertex Layout Drawings

We now present two algorithms for producing orthogonal box-drawings with a diagonal general position vertex layout.

Algorithm 7.7. DIAGONAL GENERAL POSITION D-DIMENSIONAL CUBE-DRAWING

Input: graph G.

Output: D-dimensional orthogonal hypercube-drawing of G.

---

1. Determine a D-dimensional diagonal vertex layout of G with corresponding vertex ordering determined by Algorithm 4.1 MEDIAN PLACEMENT ORDERING (with insertion ordering determined by the Algorithm 4.2 INSERTION ORDERING).

2. Determine an arc-routing with Algorithm 7.3 D-DIMENSIONAL GENERAL POSITION ARC-ROUTING.

3. Apply Algorithm 7.1 D-DIMENSIONAL GENERAL POSITION BOX-DRAWING.

Theorem 7.6. *The algorithm* DIAGONAL GENERAL POSITION D-DIMENSIONAL CUBE-DRAWING *determines a D-dimensional hypercube-drawing in $O(D(m+n))$ time such that:*

- *Each edge route has $D-1$ bends.*
- *Each vertex is 2-degree-restricted.*
- *The bounding box volume is at most*

$$\left(n + \left(\frac{n^{D-2}}{2D}\left(3m + \frac{n}{2}\right)\right)^{1/(D-1)}\right)^{D}$$

Proof. By Theorem 7.3 for arbitrary D-dimensional general position vertex layouts, each vertex is 2-degree-restricted.

For each vertex v and dimension i, $1 \le i \le D$, when applying the algorithm D-DIMENSIONAL GENERAL POSITION BOX-DRAWING,

$$\alpha_i(v) = \left\lceil \left(\frac{\max\{s(v), p(v)\}}{D}\right)^{1/(D-1)} \right\rceil .$$

Hence v is a cube, and for each dimension i, the side length of the bounding box is

$$\begin{aligned}
\sum_v \alpha_i(v) &= \sum_v \left\lceil \left(\frac{\max\{s(v), p(v)\}}{D} \right)^{1/(D-1)} \right\rceil \\
&< n + \sum_v \left(\frac{\max\{s(v), p(v)\}}{D} \right)^{1/(D-1)} \\
&\leq n + \left(n^{D-2} \sum_v \frac{\max\{s(v), p(v)\}}{D} \right)^{1/(D-1)} && \text{(by Cauchy-Schwarz)} \\
&\leq n + \left(\frac{n^{D-2}}{D} \left(\frac{3m}{2} + \frac{n}{4} \right) \right)^{1/(D-1)} && \text{(by Theorem 4.2)}
\end{aligned}$$

The result for the bounding box volume follows.

For a diagonal layout, it is easily seen that there are no edge crossings (see Section 7.1.3), so there is no need to apply Algorithm Box-Drawing Remove Edge Crossings. Hence the algorithm Diagonal General Position D-Dimensional Cube-Drawing takes $O(D(m+n))$ time. □

We now present an algorithm for producing D-dimensional orthogonal line-drawings using a diagonal layout.

Algorithm 7.8. Diagonal General Position D-Dimensional Line-Drawing

---

Input: graph G.

Output: D-dimensional orthogonal line-drawing of G.

---

1. Determine a diagonal D-dimensional general position vertex layout of G with the corresponding vertex ordering determined by Algorithm 4.1 Median Placement Ordering (with insertion ordering determined by Algorithm 4.2 Insertion Ordering).

2. Determine a $(D-1)$-dimensional arc-routing with Algorithm 7.3 General Position Arc-Routing.

3. Representing each vertex by a D-axis-parallel line, apply Algorithm 7.1 D-Dimensional General Position Box-Drawing.

Theorem 7.7. *The algorithm* DIAGONAL GENERAL POSITION D-DIMENSIONAL LINE-DRAWING *determines a D-dimensional orthogonal line-drawing of G in $O(D(m+n))$ time such that:*

- *Each edge route has $D-1$ bends.*
- *Each vertex has aspect ratio at most $\deg(v)/(D-1)+O(1)$.*
- *Each vertex is a 2-degree-restricted D-axis parallel line.*
- *The bounding box volume is at most*

$$n^{D-1}\left(\frac{(2D-3)n+3m}{2(D-1)}\right)$$

Proof. This proof is similar to that of Theorem 7.6. Algorithm D-DIMENSIONAL GENERAL POSITION ARC-ROUTING determines a $(D-1)$-dimensional arc-routing such that, for each i, $1 \le i \le D-1$,

$$M_i(v) \ \le\ \left\lceil \frac{\max\{s(v),p(v)\}}{D-1} \right\rceil \quad .$$

We represent each vertex v by a line of length

$$\begin{aligned} \alpha_D(v) \ &=\ \left\lceil \frac{\max\{s(v),p(v)\}}{D-1} \right\rceil \\ &\le \frac{\max\{s(v),p(v)\}+D-2}{D-1} \quad . \end{aligned}$$

The aspect ratio bound follows, and

$$\text{surface}\,(v) \ =\ 2\left((D-1)\alpha_D(v)+1\right) \ \le\ 2\left(\max\{s(v),p(v)\}+D-1\right) \quad .$$

Since $\max\{s(v),p(v)\} \le \deg(v)$, the drawing is 2-degree-restricted and has height

$$\begin{aligned} \sum_v \alpha_D(v) \ &\le\ \sum_v \frac{\max\{s(v),p(v)\}+D-2}{D-1} \\ &\le \left(\frac{D-2}{D-1}\right)n + \frac{6m+n}{4(D-1)} \qquad \text{(by Theorem 4.2)} \\ &\le \frac{4(D-2)n+6m+n}{4(D-1)} \\ &\le \frac{(4D-7)n+6m}{4(D-1)} \quad . \end{aligned}$$

The bounding box volume bound follows.

As was the case for cube-drawings with a diagonal layout, there is no need to apply Algorithm 7.2 Box-Drawing Remove Edge Crossings. Hence the algorithm Diagonal General Position D-Dimensional Cube-Drawing takes $O(D(m+n))$ time. □

7.3 3-D Routing-Based Algorithm

In this section we describe a routing-based approach to 3-D orthogonal box-drawing in the general position model. The following algorithm determines a general position vertex layout with respect to a predetermined arc-routing. Recall that for a given arc-routing of a graph G, for each dimension $i \in \{X, Y, Z\}$, the subgraph of $\overleftrightarrow{G}$ induced by the arcs coloured i is denoted $\overleftrightarrow{G}[i]$.

Algorithm 7.9. 3-D General Position Routing-Based Layout

Input: • graph G

• 3-D general position arc-routing of $A(G)$

Output: 3-D general position vertex layout of $V(G)$.

---

for $i \in \{X, Y, Z\}$ **do**

Determine the i-ordering

by applying Algorithm 4.1 Median Placement Ordering to $\overleftrightarrow{G}[i]$.

end-for

If $\overleftrightarrow{G}[i]$ is acyclic for each dimension $i \in \{X, Y, Z\}$, we say the arc-routing is *acyclic*, and by Theorem 4.1, Algorithm 4.1 Median Placement Ordering determines minimum cost orderings. We now describe algorithms for finding 2- and 3-colour acyclic arc-routings.

7.3.1 Acyclic Arc-Routing

To determine a 2-colour acyclic arc-routing of G, start with a vertex ordering $<$ of G, and for each edge $vw \in E(G)$ $(v < w)$, colour the arc $\overrightarrow{vw}$ with colour X and $\overrightarrow{wv}$ with colour Y. Clearly $\overleftrightarrow{G}[X]$ and $\overleftrightarrow{G}[Y]$ are both acyclic. This approach is used by Biedl and Kaufmann [30] for 2-D orthogonal graph drawing. Biedl [27] uses this 2-colour acyclic arc-routing method to determine the X- and Y-orderings of a 3-D general position vertex layout; each vertex is then represented by a line parallel to the Z-axis. The 3-D drawings produced have small volume ($O(n^2m)$) but are inherently two-dimensional. The following algorithm determines a 3-colour acyclic arc-routing and is illustrated in Figure 7.6.

Algorithm 7.10. 3-Colour Acyclic Arc-Routing

Input: A graph G.

Output: A 3-colour acyclic arc-routing of G.

Determine a 1-balanced vertex ordering $<$ of G using
    Algorithm 4.3 1-Balanced Vertex Ordering.
for each vertex $v \in V(G)$ **do**
    for $k = 1, 2, \ldots, \lfloor c(v)/2 \rfloor$ **do**
        assign the arc $\overrightarrow{vv^k}$ the colour Z
    end-for
end-for
for each uncoloured arc $\overrightarrow{vw}$ **do**
    if $v < w$ **then** assign to $\overrightarrow{vw}$ the colour X **else** assign to $\overrightarrow{vw}$ the colour Y
end-for

Lemma 7.10. *Algorithm* 3-Colour Acyclic Arc-Routing *determines a 3-colour routing of G.*

Proof. Obviously if $\overrightarrow{vw}$ is coloured X (respectively, Y) then the reversal arc $\overrightarrow{wv}$ cannot be coloured X (Y). If $\overrightarrow{vw}$ is coloured Z then $\overrightarrow{wv}$ cannot also be coloured Z, as otherwise

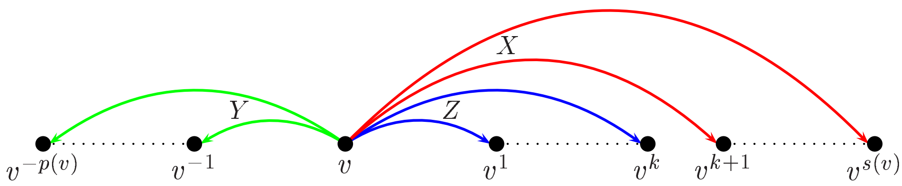


Figure 7.6: Routing arcs at a positive vertex v; $k = \lfloor c(v)/2 \rfloor$.

w would be opposite to v, and v could move past w under rule **M1**. By Lemma 4.5, reversal arcs are coloured differently and the colouring is an arc-routing. Clearly $\overleftrightarrow{G}[X]$ and $\overleftrightarrow{G}[Y]$ are acyclic. A positive vertex v cannot have an incoming arc $\overrightarrow{wv} \in \overleftrightarrow{G}[Z]$ with $v < w$ as otherwise w could move past v under rule **M1** (see Corollary 4.2). Similarly for negative vertices. Hence $\overleftrightarrow{G}[Z]$ is also acyclic. $\square$

Algorithm 7.11. Routing-Based 3-D General Position Box-Drawing

---

Input: graph G.

Output: 3-D orthogonal box-drawing of G.

---

1. Determine a 3-D arc-routing of $A(G)$ with Algorithm 7.10 3-Colour Acyclic Routing.

2. Determine a layout with Algorithm 7.9 3-D General Position Routing-Based Layout.

3. Apply Algorithm 7.1 3-Dimensional General Position Box-Drawing.

Theorem 7.8. *The algorithm* Routing-Based 3-D General Position Box-Drawing *determines a 3-D orthogonal box-drawing in $O(mn\Delta)$ time such that*

- *Each edge route has 2 bends.*
- *Each vertex v is 2-degree-restricted and has aspect ratio at most* $\deg(v)/4$.
- *The bounding box volume is*

$$\frac{\Delta(G)}{4}\left(n\left(\frac{2}{3}m + O(1)\ n\right)\right)^{3/2}$$

Proof. For a positive vertex v,

$$\deg_{\overleftrightarrow{G}[Z]}(v) = \left\lfloor \frac{c(v)}{2} \right\rfloor, \ \deg_{\overleftrightarrow{G}[Y]}(v) = \min\{s(v), p(v)\}, \text{ and } \deg_{\overleftrightarrow{G}[X]}(v) = \left\lceil \frac{\deg(v)}{2} \right\rceil .$$

For each $i \in \{X, Y, Z\}$, since $\overleftrightarrow{G}[X]$ is acyclic, by Theorem 4.1, in each of the orderings of $\overleftrightarrow{G}[X]$, $\overleftrightarrow{G}[Y]$ and $\overleftrightarrow{G}[Z]$ the cost $c_i(v) \leq 1$, for every vertex v.

$$\begin{aligned}
M_X(v) &\leq \left\lceil \frac{1}{2} \left\lceil \frac{\deg(v)}{2} \right\rceil \right\rceil &= \frac{\deg(v)}{4} + O(1) , \\
M_Y(v) &\leq \left\lceil \frac{\min\{s(v), p(v)\}}{2} \right\rceil &= \frac{\min\{s(v), p(v)\}}{2} + O(1) , \\
M_Z(v) &\leq \left\lceil \frac{1}{2} \left\lfloor \frac{c(v)}{2} \right\rfloor \right\rceil &= \frac{c(v)}{4} + O(1) \quad .
\end{aligned}$$

So, for each positive vertex v and similarly for negative vertices,

$$\begin{aligned}
& M_X(v) + M_Y(v) + M_Z(v) \\
\leq & \frac{\deg(v)}{4} + \frac{\min\{s(v), p(v)\}}{2} + \frac{c(v)}{4} + O(1) \\
\leq & \frac{\deg(v) + 2\min\{s(v), p(v)\} + \deg(v) - 2\min\{s(v), p(v)\}}{4} + O(1) \text{ (by (4.1))} \\
= & \frac{\deg(v)}{2} + O(1) \quad .
\end{aligned}$$

By Lemma 7.3,

$$\text{surface}\,(v) \ \leq \ 2\deg(v) + O(1) \quad ,$$

and v is 2-degree-restricted. A vertex v has maximum aspect ratio if, in the locally balanced vertex ordering, $c(v) = 0$, $s(v) = 0$ or $p(v) = 0$, in which case v is a line of length $\deg(v)/4$. Applying Theorem 7.2 we have

$$\text{volume (bounding box)} \ \leq \ \frac{\Delta(G)}{4} \left(n \left(\frac{2}{3}m + \frac{O(1)}{6} n \right) \right)^{3/2}$$

Applying Algorithm D-Dimensional General Position Box-Drawing, which takes $O(mn\Delta)$ time, is the most time-consuming step of the algorithm. So Algorithm Routing-Based 3-D General Position Box-Drawing takes $O(mn\Delta)$ time. □

The drawings produced by the above algorithm have smaller aspect ratio, on average, than those produced by the algorithm based on a 2-colour acyclic routing [27]. Furthermore, edges can be routed on all sides of a vertex. Hence the drawings are orientation-independent.

Part III

Other Orthogonal Graph Drawing Models

Chapter 8

Equitable Edge-Colouring

In this chapter we present and analyse a greedy algorithm for determining a (non-proper) edge-colouring of a multigraph such that for each vertex the colours are evenly distributed about the edges incident to that vertex. Such a colouring is called an equitable edge-colouring. This algorithm is used in subsequent graph drawing algorithms presented in Chapters 9 and 10 to assign ports to edge routes.

8.1 Simple Graphs

We initially recall a result due to Hilton and de Werra [117] concerning equitable edge-colourings of graphs. An edge-colouring of a graph G with k colours is said to be *equitable* if for each vertex $v \in V(G)$ and each pair of colours i and j, the number of edges incident to v coloured i and j differ by at most one.

Theorem 8.1 ([117]). *If $k \geq 2$ and G is a graph such that no vertex degree is a multiple of k, then G has an equitable edge-colouring with k colours.*

We have the following result.

Corollary 8.1. *If $k \geq 2$ and G is a graph, then there is an edge-colouring of G with k colours such that for each vertex $v \in V(G)$ and colour i, the number of edges incident with v coloured i is at most $\lceil (\deg(v) + 1)/k \rceil$.*

Proof. For each vertex $v \in V(G)$ with degree a multiple of k, add a new vertex v' and a new edge vv' to G to create a graph G'. G' has no vertex with degree a multiple of k, so G' has an equitable edge-colouring with k colours. At each vertex $v \in V(G')$ and colour i the number of edges incident to v coloured i is at most $\lceil \deg_{G'}(v)/k \rceil \leq \lceil (\deg_G(v)+1)/k \rceil$. □

8.2 Multigraphs

The result of Hilton and de Werra is dependent on the graph being simple. We now present a greedy heuristic for edge-colouring multigraphs with k colours. Given a partial edge-colouring $\mathrm{col} : E(G) \to \{1, 2, \ldots, k\}$ of a multigraph G we define

$$\begin{aligned} N(v) &= |\{vw \in E(G) : vw \text{ is coloured}\}| \\ M(v) &= \max_i |\{vw \in E(G) : \mathrm{col}(vw) = i\}| \\ C(v) &= \{i \in \{1,2,\ldots,k\} : M(v) = |\{vw \in E(G) : \mathrm{col}(vw) = i\}|\} \ . \end{aligned}$$

$M(v)$ is the maximum number of edges incident with v assigned the same colour, and $C(v)$ is the set of colour(s) most abundant at v.

Algorithm 8.1. Quasi-Equitable Edge-Colour

Input: multigraph G, positive integer k.

Output: edge-colouring of G with at most k colours.

---

for each edge $vw \in E(G)$ **do**

  if $C(v) \cup C(w) \neq \{1, 2, \ldots, k\}$ **then** Choose $i \in \{1, 2, \ldots, k\} \setminus (C(v) \cup C(w))$.

  else if $C(v) = C(w)$ **then** Choose $i \in \{1, 2, \ldots, k\}$.

  else if $|C(v)| \geq |C(w)|$ **then** Choose $i \in C(v) \setminus C(w)$.

  else $(|C(w)| > |C(v)|)$ Choose $i \in C(w) \setminus C(v)$.

  Set the colour of vw to be i.

end-for

Theorem 8.2. *The algorithm* QUASI-EQUITABLE EDGE-COLOUR *will determine, in* $O(m^2)$ *time, a edge k-colouring of a multigraph G, such that for every vertex $v \in V(G)$,*

$$M(v) \leq \frac{2\deg(v)}{k} + 1 \ .$$

Proof. Firstly, observe that

$$N(v) \geq |C(v)| \cdot M(v) \ . \tag{8.1}$$

At each step of the algorithm the only vertex u for which $M(u)$ can possibly increase is v and w. So, for each vertex v we apply induction on $N(v)$ with the following inductive hypothesis.

$$\text{if } N(v) \leq t \text{ then } M(v) \leq \frac{2N(v)}{k} + 1 \ . \tag{8.2}$$

The basis for the induction is trivial. Now, suppose that for $N(v) = t$, $M(v) \leq 2N(v)/k + 1$ and the next edge incident to v to be coloured is vw.

In the first case of the algorithm vw is coloured with a colour not in $C(v)$, so $M(v)$ does not increase. By (8.2) for $N(v) = t$, (8.2) holds for $N(v) = t + 1$.

In the second case, $C(v) = C(w) = \{1, 2, \ldots, k\}$. By (8.1), $N(v) \geq |C(v)| \cdot M(v) = k \cdot M(v)$. So $M(v) \leq N(v)/k \leq 2(N(v) + 1)/k + 1$, and (8.2) holds for $N(v) = t + 1$.

In the third case, $C(v) \cup C(w) = \{1, 2, \ldots, k\}$ and $|C(v)| \geq |C(w)|$. So $|C(v)| \geq k/2$. By (8.1) $N(v) \geq kM(v)/2$, so $M(v) \leq 2N(v)/k$, and (8.2) holds for $N(v) = t + 1$.

In the fourth case, the edge vw is coloured with a colour not in $C(v)$, so $M(v)$ does not increase. By (8.2) for $N(v) = t$, (8.2) holds for $N(v) = t + 1$.

Upon termination of the algorithm $N(v) = \deg(v)$, so for every vertex $v \in V(G)$, $M(v) \leq 2\deg(v)/k + 1$.

We now analyse the time complexity of the algorithm. It is easily seen that the iteration of the algorithm corresponding to the colouring of the edge vw takes $O(\deg(v) + \deg(w) + k)$ time. So the algorithm takes

$$\sum_{vw \in E(G)} O(\deg(v) + \deg(w) + k) \ = \ O\left(mk \ + \sum_{v \in V(G)} \deg(v)^2 \right)$$

time. We now prove that for non-negative numbers $d_1, d_2, \ldots, d_n$,

$$\sum_{i=1}^{n} d_i^2 \ \leq \ \left(\sum_{i=1}^{n} d_i \right)^2 \ . \tag{8.3}$$

The result will follow. We proceed by induction on n. For $n = 1$, equality holds in (8.3). Assume that (8.3) holds for all $n_0 < n$. Then

$$\begin{aligned}
\sum_{i=1}^{n} d_i^2 &= \sum_{i=1}^{n-1} d_i^2 + d_n^2 \\
&\leq \left(\sum_{i=1}^{n-1} d_i\right)^2 + d_n^2 \text{ (by induction)} \\
&\leq \left(\sum_{i=1}^{n-1} d_i\right)^2 + d_n^2 + 2d_n \left(\sum_{i=1}^{n-1} d_i\right) \\
&= \left(\left(\sum_{i=1}^{n-1} d_i\right) + d_n\right)^2 \\
&= \left(\sum_{i=1}^{n} d_i\right)^2
\end{aligned}$$

So the time taken by the algorithm is

$$O\left(mk + \left(\sum_{v} \deg(v)\right)^2\right) = O\big(mk + 4m^2\big) \ .$$

If $k > m$ then trivially there is an edge m-colouring of G with the required properties, so we can assume that $k \leq m$. Hence the algorithm takes $O\big(m^2\big)$ time. □

Finally, we present a well-known algorithm for the case of $k = 2$, which provides an improvement on the previous result. This technique has been employed for graph drawing in [30, 31] for example.

Algorithm 8.2. 2-Edge-Colour

Input: multigraph G.

Output: edge 2-colouring of G.

---

1. Pair the odd degree vertices of G, and add an edge to G between the paired vertices. All vertices now have even degree.

2. Follow an Eulerian tour of G, and colour the edges alternately with different colours.

Theorem 8.3. *The algorithm* 2-EDGE-COLOUR *will, in $O(m)$ time, determine a edge 2-colouring of a multigraph G, such that for every vertex $v \in V(G)$,*

$$M(v) \leq \left\lceil \frac{\deg(v)}{2} \right\rceil + 1 \ .$$

Proof. In any graph there is an even number of vertices with odd degree, so the first step of the algorithm is valid. An undirected graph has an Eulerian tour if and only if every vertex has even degree. See Even [90] for an algorithm for finding an Eulerian tour in $O(m)$ time.

At each vertex v, there is at most one 'extra' edge incident with v added in Step 1. If the Eulerian tour has odd length then the first and last edges in the tour will receive the same colour. Therefore, at every vertex v, there will be at least $\lceil \deg(v)/2 \rceil - 1$ pairs of edges incident with v receiving different colours. The remaining (≤ 2) edges incident to v may receive the same colour, so the maximum number of edges incident with v and receiving the same colour is $\lceil \deg(v)/2 \rceil + 1$. □

Chapter 9

The Coplanar Vertex Layout Model for Three-Dimensional Orthogonal Graph Drawing

In this chapter we present algorithms for producing 3-D orthogonal drawings in the coplanar vertex layout model; i.e., there exists a single grid-plane intersecting every vertex. We present three algorithms, for producing (1) 1-bend line-drawings, (2) drawings with optimal volume, and (3) cube-drawings with optimal volume. A disadvantage of this model is that the drawings produced are inherently orientation-dependent.

In this chapter we present algorithms for determining *coplanar* 3-D orthogonal drawings; i.e., there exists a grid-plane intersecting every vertex. Section 9.1 describes an algorithm which represents the vertices by Z-lines positioned in a 2-D diagonal, and produces 1-bend line-drawings based on a book embedding of the graph.

The algorithms in the remainder of the chapter are a product of joint research with Therese Biedl and Torsten Thiele [34]. In Section 9.2 we present an algorithm which positions the vertices in $O(\sqrt{n}) \times O(\sqrt{n})$ grid, and produces line-drawings with optimal volume for regular graphs, and four bends per edge route. A variation of this algorithm produces 3-bend drawings with an increase in the volume. Our algorithm presented in Section 9.3 positions the vertices in a $O\big(\sqrt{m+n}\big) \times O\big(\sqrt{m+n}\big)$ grid, and determines degree-restricted cube-drawings with $O\big((m+n)^{3/2}\big)$ volume, which is optimal. This algorithm, which can be considered a generalisation of the COMPACT algorithms of Eades *et al.* [86, 87] for 3-D point-drawing, is an improvement on the line-drawing algorithm of Wood [223].

9.1 1-Bend Box-Drawing Algorithm

Biedl *et al.* [32, 33] construct 3-D orthogonal drawings of K_n, and hence for any simple graph, with $O\big(n^3\big)$ volume and one bend per edge route. This construction, called the LIFTING-EDGES algorithm by Biedl [27], represents the vertices as Z-lines of length n positioned in a 2-D diagonal layout. Each edge is routed with one bend in some Z-plane. As mentioned in [32, 33], the assignment of Z-planes to edge routes is closely related to the assignment of pages to edges in book embeddings. The following algorithm, illustrated in Figure 9.1, exploits a book embedding to construct 3-D orthogonal drawings with one bend per edge route.

Algorithm 9.1. COPLANAR 1-BEND DRAWING

Input: n-vertex m-edge multigraph G with genus g.

Output: 3-D orthogonal drawing of G.

---

1. Find a book-embedding of G using the algorithm of Malitz [150] (see Section 1.3). Suppose $(v_1, v_2, \dots, v_n)$ is the spine ordering and page : $E(G) \to \{1, 2, \dots, P\}$ is the page numbering with $P = O\big(\sqrt{g}\big)$.

2. Orient each edge $v_i v_j \in E(G)$ from left to right in the ordering $(v_1, v_2, \dots, v_n)$; i.e., if $i < j$ then the edge $v_i v_j$ is directed from v_i to v_j.

3. Denote by G^R the subgraph of G consisting of the edges in any page $p \in \{1, 2, \dots, \lceil P/2 \rceil\}$, and by G^L the subgraph of G consisting of the edges in the remaining pages. (Edges in G^R will be routed through grid-points (x, y, z) with $x \geq y$, and edges in G^L will be routed through grid-points (x, y, z) with $y \geq x$.)

4. Determine edge-colourings of G^R and of G^L, each with $\lceil 2m/n \rceil$ colours, using Algorithm 8.1 QUASI-EQUITABLE EDGE-COLOUR. Suppose col : $E(G) \to$

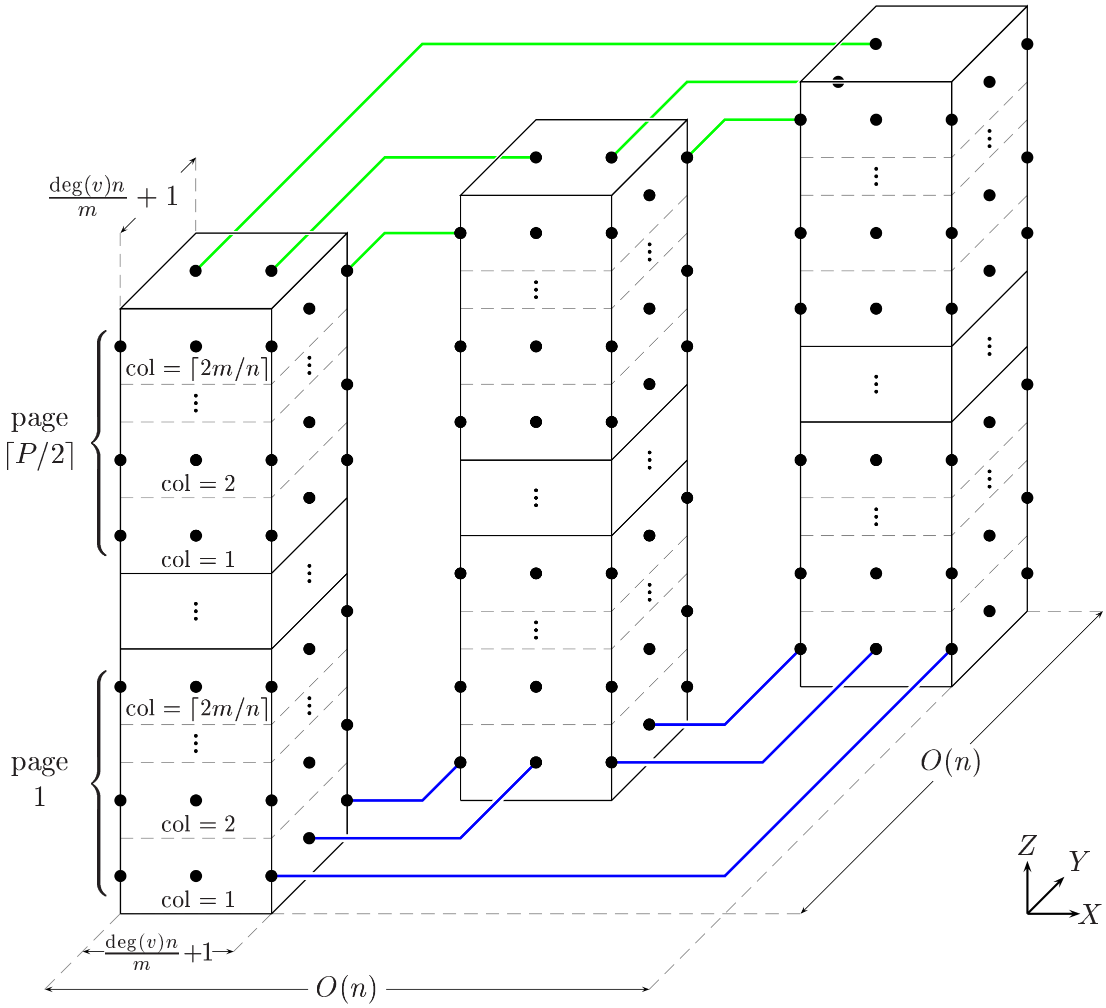


Figure 9.1: Coplanar 1-bend drawing with a diagonal vertex layout.

$\{1, 2, \ldots, \lceil 2m/n \rceil\}$ is the resulting edge-colouring of G.

5. For each vertex $v \in V(G)$, suppose $M^{+R}(v)$ (respectively, $M^{+L}(v)$) is the maximum number of outgoing edges $\overrightarrow{vw} \in E(G^R)$ ($\overrightarrow{vw} \in E(G^L)$) on the same page and receiving the same colour. Similarly, $M^{-R}(v)$ (respectively, $M^{-L}(v)$) is the maximum number of incoming edges $\overrightarrow{wv} \in E(G^R)$ ($\overrightarrow{wv} \in E(G^L)$) on the same page and receiving the same colour.

6. For each vertex $v_i \in V(G)$, set

$$\begin{aligned} M_X(v_i) &= \max\left\{M^{+L}(v_i), M^{-R}(v_i)\right\}, \text{ and} \\ M_Y(v_i) &= \max\left\{M^{+R}(v_i), M^{-L}(v_i)\right\} \quad . \end{aligned}$$

Represent v_i by the

$$M_X(v_i) \times M_Y(v_i) \times \left\lceil \frac{P}{2} \right\rceil \left\lceil \frac{2m}{n} \right\rceil$$

box with maximum corner at

$$\left(\sum_{j \leq i} M_X(v_j), \sum_{j \leq i} M_Y(v_j), \left\lceil \frac{P}{2} \right\rceil \left\lceil \frac{2m}{n} \right\rceil \right) .$$

(Note that for vertices v with degree at most the average degree $\frac{2m}{n}$, $M_X(v)$ and $M_Y(v)$ will probably be 1, and hence v will be represented by a line.)

7. For each vertex $v \in V(G)$, for each page $p \in \{1, 2, \ldots, \lceil P/2 \rceil\}$, and for each colour $c \in \{1, 2, \ldots, \lceil 2m/n \rceil\}$, suppose $\{\overrightarrow{vw}_1, \overrightarrow{vw}_2, \ldots, \overrightarrow{vw}_k\}$ are the outgoing edges at v in G^R which are coloured c and appear in page p, where $w_1 \leq w_2 \leq \cdots \leq w_k$ in the spine ordering. As illustrated in Figure 9.1, assign the X^+-ports at v with Z-coordinates of $(p-1)\lceil 2m/n \rceil + c$ to these edges, such that, if $i < j$ then the Y-coordinate of the port assigned to $\overrightarrow{vw}_i$ is less than the Y-coordinate of the port assigned to $\overrightarrow{vw}_j$. Now suppose $\{\overrightarrow{w_1v}, \overrightarrow{w_2v}, \ldots, \overrightarrow{w_kv}\}$ are the incoming edges at v in G^R which are coloured c and appear in page p, where $w_k \leq w_{k-1} \leq \cdots \leq w_1$ in the spine ordering (taking care to consistently order parallel edges $\{vw\}$ at v and w; see Figure 9.1). Assign the Y^--ports at v with Z-coordinates of $(p-1)\lceil 2m/n \rceil + c$ to these edges, such that, if $i < j$ then the X-coordinate of the port assigned to $\overrightarrow{w_iv}$ is less than the X-coordinate of the port assigned to $\overrightarrow{w_jv}$.

8. For each edge $\overrightarrow{vw} \in E(G^R)$, if $\overrightarrow{vw}$ has been assigned ports at v and w with coordinates of (x_v, y_v, z_0) and (x_w, y_w, z_0) respectively, then route $\overrightarrow{vw}$ with one bend as follows:

$$(x_v, y_v, z_0) \rightarrow (x_w, y_v, z_0) \rightarrow (x_w, y_w, z_0)$$

9. In an analogous manner to the case for edges in G^R, route edges $\overrightarrow{vw} \in E(G^L)$ using Y^+-ports at v and X^--ports at w, as illustrated in Figure 9.1.

Theorem 9.1. *The algorithm* Coplanar 1-Bend Drawing *determines an orthogonal box-drawing of* G *with one bend per edge and* $O(nm\sqrt{g})$ *volume, where* g *is the genus of* G.

Proof. By construction each edge has one bend, and edge routes are assigned unique ports, so two X-segments do not intersect, and two Y-segments do not intersect. An X-segment and a Y-segment can only intersect if they have the same Z-coordinate. Two edges have the same Z-coordinate if and only if they are on the same page of the book embedding and they receive the same colour in Step 4. Hence the X-segment and the Y-segment of edges on different pages of the book embedding or receiving a different colours, will not intersect. By the method used in Step 7 for assigning ports to edges on the same page and receiving the same colour, such edge routes will not intersect. Hence no two edges routes intersect.

In the edge-colouring of G^R, the maximum number of edges incident to a vertex v receiving the same colour, by Theorem 8.2, is at most $2\deg_{G^R}(v)/\lceil 2m/n\rceil + 1 \leq n\deg_G(v)/m + 1$. So each of $M^{+R}(v)$, $M^{-R}(v)$, $M^{+L}(v)$ and $M^{-L}(v)$ is at most $n\deg_G(v)/m+1$, and hence $M_X(v)$ and $M_Y(v)$ are at most $n\deg_G(v)/m+1$. The width and depth of the bounding box is therefore at most $\sum_v (n\deg_G(v)/m + 1) = 3n$. The height of the bounding box is $\lceil P/2\rceil\lceil 2m/n\rceil = O(m\sqrt{g}/n)$. So the bounding box has volume $O(nm\sqrt{g})$. □

Note that smaller drawings can be produced in practice by the following modification to algorithm Coplanar 1-Bend Drawing. For each page p, determine an edge-colouring (still with $\lceil 2m/n\rceil$ colours) of the subgraph of G consisting of the edges in page p such that, for each vertex v, there at at most $\deg(v)n/m$ edges incident to v receiving the same colour. Then we need only allocate as many layers for the routing of edges in page p, as there are used colours.

Since the genus of a multigraph is the same as the genus of the underlying simple graph, and since the genus of a graph is at most m, our volume bound is $O(\min\{n^2m, nm^{3/2}\})$. Note that, for the complete graph K_n, this volume bound is $O(n^4)$, which is more than the volume of the construction of K_n due to Biedl *et al.* [32, 33]. For sparse graphs with $m = O(n^{4/3})$ the above algorithm produces drawings with less volume than the K_n construction. The following open problem is of interest.

Open Problem 9.1. Does every graph have an orthogonal box-drawing with one bend per edge and $O\big(n^2\sqrt{m}\big)$ volume?

9.2 Optimal Volume Line-Drawing Algorithm

The following algorithm for producing coplanar orthogonal line-drawings represents the vertices by Z-lines in a $O(\sqrt{n}) \times O(\sqrt{n})$ grid. Edges are routed with four bends in layers, each consisting of two Z-planes, as illustrated in Figure 9.2.

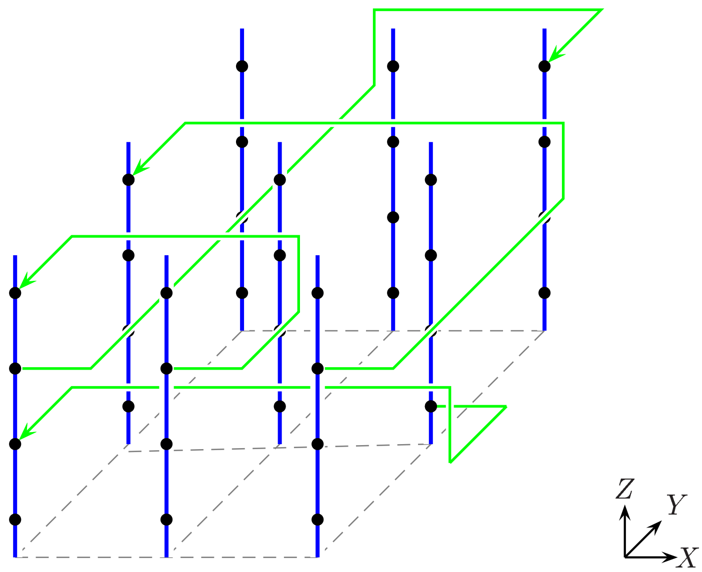


Figure 9.2: 4-bend edge routes.

Algorithm 9.2. OPTIMAL VOLUME LINE-DRAWING

Input: n-vertex m-edge multigraph G with maximum degree Δ.

Output: 3-D orthogonal line-drawing of G.

---

1. Assign to each vertex $v \in V(G)$ a unique pair

$$(x_v, y_v) \in \big\{1, 2, \ldots, \lceil\sqrt{n}\rceil\big\} \times \big\{1, 2, \ldots, \lceil\sqrt{n}\rceil\big\} \quad .$$

2. Pair the odd degree vertices of G and add an edge between the paired vertices. Orient the edges of G by following an Eulerian tour of G. Remove the inserted edges.

3. Construct a graph H with $V(H) = E(G)$, and add an edge to H between the vertices corresponding to oriented edges $\overrightarrow{vw}$ and $\overrightarrow{xy}$ if v is in the same column as x, or w is in the same row as y.

4. Vertex-colour the graph H using the algorithm GREEDY VERTEX-COLOUR with colours $\{0, 1, \ldots, \Delta(H)\}$ (see Section 2.2). For each edge $vw \in E(G)$, if the vertex of H corresponding to vw is coloured $i \in \{0, 1, \ldots, \Delta(H)\}$ then set the *height* $h(vw) \leftarrow 2i$. Suppose $M = \max_{vw \in E(G)} h(vw) + 1$.

5. Represent each vertex v by the line

$$\big(2x_v, 2y_v, 0\big) \longrightarrow \big(2x_v, 2y_v, M\big) \ .$$

6. For each oriented edge $\overrightarrow{vw} \in E(G)$, construct the following edge route for vw, as illustrated in Figure 9.2.

$$(2x_v, 2y_v, h(\overrightarrow{vw})) \to (2x_v + 1, 2y_v, h(\overrightarrow{vw})) \to (2x_v + 1, 2y_v + 1, h(\overrightarrow{vw})) \to$$
$$(2x_v + 1, 2y_v + 1, h(\overrightarrow{vw}) + 1) \to (2x_w, 2y_v + 1, h(\overrightarrow{vw}) + 1) \to (2x_w, 2y_w, h(\overrightarrow{vw}) + 1)$$

Theorem 9.2. *The algorithm* OPTIMAL VOLUME LINE-DRAWING *determines a 3-D orthogonal line-drawing of G in $O(m\Delta\sqrt{n})$ time with $O\big(\Delta n^{3/2}\big)$ volume and four bends per edge route.*

Proof. In each edge route the first, third and fifth segments have unit length. An edge crossing involving a unit-length segment must also involve one of the adjacent segments in the edge route, so to show that the drawing is crossing-free, we need only consider potential intersections between the second and the fourth segments of the edge routes. These segments are parallel to the Y- and X-axes, respectively. Such Y-segments have even Z-coordinate, and such X-segments have odd Z-coordinate, so an X-segment does not intersect a Y-segment. For two X-segments to intersect, they must have the same height and be routed in the same row. Since oriented edges destined for vertices in the same row receive different heights, no two X-segments intersect. Similarly, for two Y-segments to intersect, they must have the same height and be routed in the same column. Since oriented edges starting at vertices in the same column receive different heights, no two Y-segments intersect.

The vertex in H corresponding to an edge $\overrightarrow{vw} \in E(G)$ has degree

$$\sum_{x \text{ in row}(v)} \deg(x) \; + \sum_{y \text{ in row}(w)} \deg(y) \enspace .$$

So the maximum degree of H is at most $2\Delta \lceil\sqrt{n}\rceil$. Hence the maximum height of an edge route is $4\Delta \lceil\sqrt{n}\rceil + 1 = O(\Delta\sqrt{n})$. Since the width and depth of the drawing are both $2\lceil\sqrt{n}\rceil$, the bounding box has $O\big(\Delta n^{3/2}\big)$ volume.

The greedy vertex-colouring of H takes $O(|E(H)|)$ time. Since $|V(H)| = m$ and $\Delta(H) \leq 2\Delta\lceil\sqrt{n}\rceil$, the algorithm takes $O(m\Delta\sqrt{n})$ time. □

For regular graphs, the above algorithm produces drawings with $O(m\sqrt{n})$ volume, which by Theorem 3.2 is optimal for any 3-D orthogonal graph drawing. By drawing vertices of large degree separately, and using a particular layout of the remaining vertices, a modification of the above algorithm achieves this optimal bound for all graphs (see [34]).

If we eliminate the middle segment from each edge route used in Algorithm 9.2 Optimal Volume Line-Drawing, and assign each edge a unique height then we obtain the following result.

Theorem 9.3. *A 3-D orthogonal line-drawing of a multigraph G can be determined in $O(m)$ time with $O(nm)$ volume and three bends per edge route.* □

This algorithm is particularly appropriate for multilayer VLSI as there are no vertical edge segments, which are called *cross-cuts*; see [2].

9.3 Optimal Volume Cube-Drawing Algorithm

In the following algorithm for producing coplanar orthogonal drawings, vertices are initially represented by squares in the $(Z = 0)$-plane, and their positions are determined by an $O\left(\sqrt{m+n}\right) \times O\left(\sqrt{m+n}\right)$ square-packing. Vertices are then extended in the Z dimension to form cubes, and edges are routed either above or below the vertices. By Theorem 3.2, the bounding box volume of $O\left((m+n)^{3/2}\right)$ is optimal for degree-restricted orthogonal box-drawings with bounded aspect ratio (assuming $m = \Omega(n)$).

Algorithm 9.3. Optimal Volume Cube-Drawing

Input: n-vertex m-edge multigraph G.

Output: 3-D orthogonal cube-drawing of G.

---

1. Determine an edge 2-colouring of G using Algorithm 8.2 2-Edge-Colour. Suppose the induced subgraphs are G^+ and G^-, and for each vertex $v \in V(G)$ set

$$M(v) = \max\{\deg_{G^+}(v), \deg_{G^-}(v)\} \quad .$$

Orient the edges of G by following the Eulerian tour used in Algorithm 8.2.

2. For each vertex $v \in V(G)$, initially represent v by a square S_v of size

$$\left(2\left\lceil\sqrt{M(v)}\right\rceil + 2\right) \times \left(2\left\lceil\sqrt{M(v)}\right\rceil + 2\right) \quad .$$

3. Position the squares $\{S_v : v \in V(G)\}$ in the $(Z = 0)$-plane with the square-packing algorithm of Kleitman and Krieger [127].

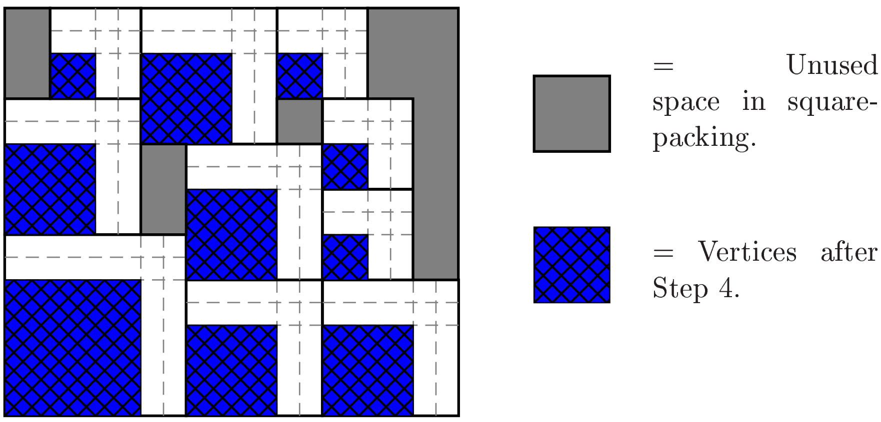


Figure 9.3: Square packing.

4. For each vertex $v \in V(G)$, let $(x_0, y_0, 0)$ be the grid-point in S_v with minimum even X-coordinate and minimum even Y-coordinate. Replace S_v by the

$$\left(2\left\lceil\sqrt{M(v)}\right\rceil - 1\right) \times \left(2\left\lceil\sqrt{M(v)}\right\rceil - 1\right) \times \left(2\left\lceil\sqrt{M(v)}\right\rceil - 1\right)$$

cube with minimum corner at $\left(x_0, y_0, 2 - 2\left\lceil\sqrt{M(v)}\right\rceil\right)$ (see Figure 9.3).

5. Assign each edge $vw \in E(G^+)$ unique Z^+-ports at v and w both with even X-coordinate and even Y-coordinate.

6. Construct a graph H with $V(H) = E(G^+)$, and add the edge $\{vw, xy\}$ to $E(H)$ if the port assigned to vw at v is in the same column as the port assigned to xy at x, or the port assigned to vw at w is in the same row as the port assigned to xy at y.

7. Vertex-colour the graph H using the algorithm Greedy Vertex-Colour with colours $\{1, 2, \ldots, \Delta(H) + 1\}$ (see Section 2.2). For each vertex $v \in V(H)$ coloured i corresponding to an edge vw, set the *height* $h(vw) \leftarrow i$.

8. For each oriented edge $vw \in E(G^+)$, construct an edge route for vw as follows. Suppose the ports on v and w assigned to vw have coordinates $(v_X, v_Y, 0)$ and $(w_X, w_Y, 0)$, respectively. Route the edge vw with one of the following four or six bend routes, as illustrated in Figure 9.4.

   - $v_X = w_X$:

     $(v_X, v_Y, 0) \to (v_X, v_Y, 2h(vw)) \to (v_X + 1, v_Y, 2h(vw)) \to$
     $(v_X + 1, w_Y, 2h(vw)) \to (v_X, w_Y, 2h(vw)) \to (v_X, w_Y, 0)$

   - $v_Y = w_Y$:

     $(v_X, v_Y, 0) \to (v_X, v_Y, 2h(vw) + 1) \to (v_X, v_Y + 1, 2h(vw) + 1) \to$
     $(w_X, v_Y + 1, 2h(vw) + 1) \to (w_X, v_Y, 2h(vw) + 1) \to (w_X, v_Y, 0)$

   - $v_X \neq w_X$ and $v_Y \neq w_Y$:

     $(v_X, v_Y, 0) \to (v_X, v_Y, 2h(vw)) \to (v_X + 1, v_Y, 2h(vw)) \to$
     $(v_X + 1, w_Y + 1, 2h(vw)) \to (v_X + 1, w_Y + 1, 2h(vw) + 1) \to$
     $(w_X, w_Y + 1, 2h(vw) + 1) \to (w_X, w_Y, 2h(vw) + 1) \to (w_X, w_Y, 0)$

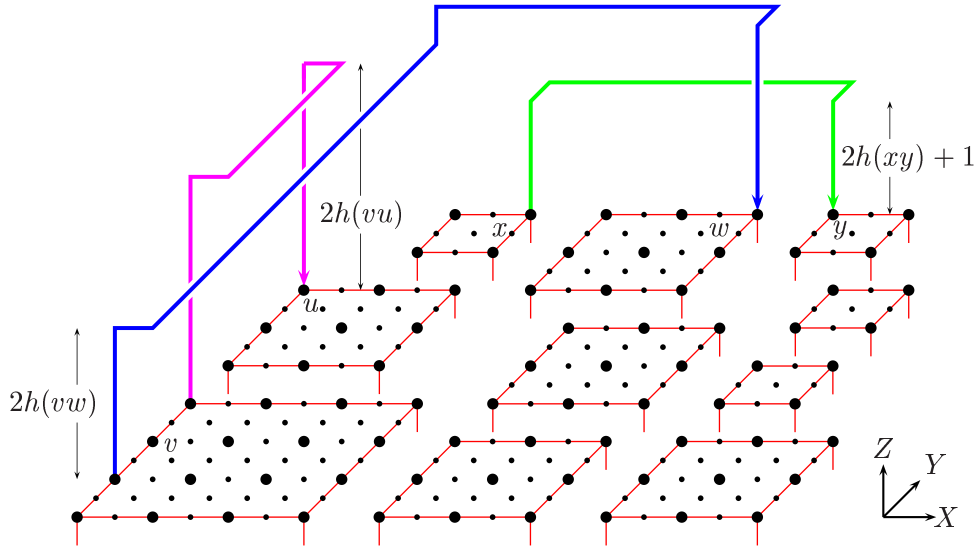


Figure 9.4: Routing edges above the plane $Z = 0$.

9. Repeat Steps 5-8 for the edges in $E(G^-)$, assigning Z^--ports and constructing edge routes below the vertices.

Theorem 9.4. *The algorithm* Optimal Volume Cube-Drawing *determines an orthogonal cube-drawing of G in $O\big(m\sqrt{m+n}\big)$ time, with $O\big((m+n)^{3/2}\big)$ bounding box volume and at most six bends per edge. Each vertex is 12-degree-restricted.*

Proof. After step 3, vertices are disjoint with Z^+-faces in the $(Z = 0)$-plane, and with corners at grid-points with even coordinates. So, for each vertex v, the the number of Z^+-ports on S_v with even X- and even Y-coordinate is $\left\lceil \sqrt{M(v)} \right\rceil^2 \geq M(v)$, so there are enough ports on v for the routing of edges in G^+ on the Z^+-face, and for edges in G^- on the Z^--face.

In each edge route, there are no consecutive unit length segments. Therefore to show that the drawing is crossing-free, we need only show that non-unit length edge segments do not intersect. Vertical segments cannot intersect because unique ports are assigned to the edges. X-parallel segments have odd Z-coordinate and Y-parallel segments have even Z-coordinate, so an X-parallel segment cannot intersect a Y-parallel segment. A vertical segment has even X and Y coordinate, a X-parallel segment has odd Y-

coordinate, and a Y-parallel segment has odd X-coordinate, so a vertical segment cannot intersect a X- or Y-parallel segment. Two Y-parallel segments can only intersect if they overlap. Since edges originating in the same column have different heights, two Y-parallel segments cannot intersect. Similarly, two X-parallel segments can only intersect if originating in the same row and in this case they have different heights, so they cannot intersect. So no two edges can intersect.

For each vertex v, the surface (v) is

$$6\left(2\left\lceil\sqrt{M(v)}\right\rceil-1\right)^2 \leq 6\left(\sqrt{2\deg(v)}+O(1)\right)^2 = 12\deg(v)+O\left(\sqrt{\deg(v)}\right) \quad .$$

Thus v is 12-degree-restricted.

The total area of the squares $\{S_v : v \in V(G)\}$ (before Step 3) is $\sum_v \left(2\left\lceil\sqrt{M(v)}\right\rceil+1\right)^2$. By Theorem 8.3, $M(v) \leq \lceil\deg(v)/2\rceil + 1$, thus the total area is at most

$$\begin{aligned}
&\sum_v \left(2\left\lceil\sqrt{\lceil\deg(v)/2\rceil+1}\right\rceil+1\right)^2 \\
\leq &\sum_v \left(\sqrt{2\deg(v)}+O(1)\right)^2 \\
\leq &\sum_v \left(2\deg(v)+O\left(\sqrt{\deg(v)}\right)+O(1)\right) \\
\leq &\ 4m+O\left(n+\sum_v\sqrt{\deg(v)}\right) \\
\leq &\ 4m+O\left(n+\sqrt{n\sum_v\deg(v)}\right) \qquad \text{(by Cauchy-Schwarz)} \\
\leq &\ 4m+O\left(n+\sqrt{nm}\right) \quad .
\end{aligned}$$

The algorithm of Kleitman and Krieger [127] packs squares with a total area of 1 in a $\frac{2}{\sqrt{3}} \times \sqrt{2}$ rectangle. So the squares $\{S_v : v \in V(G)\}$ can be packed in a rectangle with size

$$\begin{aligned}
&\left(\frac{2}{\sqrt{3}}\sqrt{4m+O\left(n+\sqrt{nm}\right)}\right) \times \left(\sqrt{2}\sqrt{4m+O\left(n+\sqrt{nm}\right)}\right) \\
\leq &\left(4\sqrt{\frac{m}{3}}+O\left(\sqrt{n+\sqrt{nm}}\right)\right) \times \left(2\sqrt{2m}+O\left(\sqrt{n+\sqrt{nm}}\right)\right) \quad .
\end{aligned}$$

The maximum degree of H is thus

$$\Delta(H) \leq \left(\frac{4}{\sqrt{3}}+2\sqrt{2}\right)\sqrt{m}+O\left(\sqrt{n+\sqrt{nm}}\right) \quad .$$

A greedy vertex-colouring of H requires at most $\Delta(H)+1$ colours; hence the height of the drawing above the $(Z=0)$-plane, and the height below the vertices, which is twice the number of colours, is

$$\left(\frac{8}{\sqrt{3}}+4\sqrt{2}\right)\sqrt{m}+O\left(\sqrt{n+\sqrt{nm}}\right) \ .$$

The height of the vertices is $\max_v 2\left\lceil\sqrt{M(v)}\right\rceil - 1 \ \leq \ \max_v \sqrt{2\deg(v)} + O(1) \ = \sqrt{2\Delta(G)}+O(1) \ \leq \ \sqrt{2m}+O(1)$. Thus the total height of the drawing is at most

$$\left(\frac{16}{\sqrt{3}}+9\sqrt{2}\right)\sqrt{m}+O\left(\sqrt{n+\sqrt{nm}}\right) \ .$$

We have shown that each of the height, width and depth of the drawing is

$$O\left(\sqrt{m}+\sqrt{n+\sqrt{nm}}\right) \ . \tag{9.1}$$

If $n=O(m)$ then (9.1) is $O(\sqrt{m})$, and if $m=O(n)$ then (9.1) is $O(\sqrt{n})$. Hence the height, width and depth of the drawing are each $O(\sqrt{m}+\sqrt{n})$, which is $O\big(\sqrt{m+n}\big)$ by the Cauchy-Schwarz inequality. The volume of the bounding box is therefore $O\big((m+n)^{3/2}\big)$. Note that in most applications $n \ll m$, hence the volume is

$$\approx \frac{4}{\sqrt{3}}\cdot 2\sqrt{2}\cdot\left(\frac{16}{\sqrt{3}}+9\sqrt{2}\right)m^{3/2} \ < \ 144m^{3/2} \ .$$

The time-consuming stage of the algorithm is the vertex-colouring of H. This can be computed in

$$O(|E(H)|) \ = \ O(|V(H)|\Delta(H)) \ = \ O\left(m\left(\sqrt{m}+\sqrt{n+\sqrt{nm}}\right)\right) \ ,$$

which is $O\big(m\sqrt{m+n}\big)$ by the same argument used above. By construction there are at most six bends per edge. □

If we remove the middle segment from each edge and assign each edge a unique height then the overall height is $O(m)$ and we obtain the following result.

Theorem 9.5. *Every graph has an orthogonal cube-drawing, which can be computed in $O(m)$ time, with $O(m(m+n))$ bounding box volume and five bends per edge. Each vertex is 12-degree-restricted.* □

Note that if we reduce the length in the Z-direction of the box representing a vertex then the surface of the box can be reduced at the expense of an increase in the aspect ratio. In particular, for aspect ratio r, it is easily seen that a vertex will be $4(1 + 2/r)$-degree-restricted.

Chapter 10

The Non-Collinear Vertex Layout Model for Three-Dimensional Orthogonal Graph Drawing

In this chapter we present an algorithm for producing 3-D orthogonal box-drawings in the non-collinear model. The box-drawings produced have optimal volume for regular graphs. We use this algorithm as the basis for another algorithm to generate 3-D orthogonal point-drawings with optimal volume. The advantage of this model over the coplanar vertex layout model is that the drawings are orientation-independent, which for point-drawings comes at the cost of one more bend per edge route.

10.1 Box-Drawing Algorithm

The algorithm to follow determines a 3-D non-collinear vertex layout by lifting the vertices from a plane grid into 3-D space in an orientation-independent manner. We call the box surrounding the vertices the *inner box*. For each direction $d \in \{X^{\pm}, Y^{\pm}, Z^{\pm}\}$, the box extending out from the d-face of the inner box is called the d-*outer box*, as shown in Figure 5.18 (page 114). Each edge is routed in an outer box determined by an equitable edge-colouring. Within each outer box, the routing of edges resembles the method employed in Algorithm 9.3 OPTIMAL VOLUME CUBE-DRAWING.

Algorithm 10.1. NON-COLLINEAR BOX-DRAWING

Input: multigraph G with maximum degree Δ.

Output: 3-D orthogonal cube-drawing of G.

---

1. Assign each vertex $v \in V(G)$ a unique pair $(x(v), y(v))$ with

$$0 \leq x(v), y(v) \leq \lceil \sqrt{n} \rceil - 1 \ .$$

2. For each vertex $v \in V(G)$, set $z(v) \leftarrow x(v) + y(v) \pmod{\lceil \sqrt{n} \rceil}$ (see Figure 10.1).

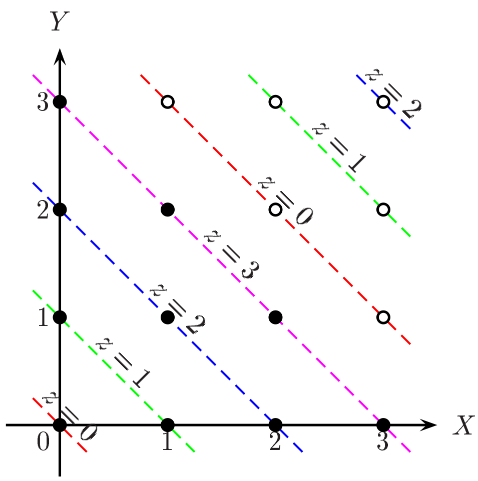


Figure 10.1: Determining $z(v)$.

3. Define the 'vertex spacing' $\Lambda = 2\left(\left\lceil \sqrt{\lceil \Delta(G)/3 \rceil} \right\rceil + 1\right)$.

4. Represent each vertex $v \in V(G)$ by the

$$\left(2\left\lceil \sqrt{\lceil \deg(v)/3 \rceil} \right\rceil + 1\right) \times \left(2\left\lceil \sqrt{\lceil \deg(v)/3 \rceil} \right\rceil + 1\right) \times \left(2\left\lceil \sqrt{\lceil \deg(v)/3 \rceil} \right\rceil + 1\right)$$

cube with minimum corner at $(\Lambda x(v), \Lambda y(v), \Lambda z(v))$, as shown in Figure 10.2.

5. Apply Algorithm 8.1 QUASI-EQUITABLE EDGE-COLOUR to G with $k = 6$. Suppose the edge-colouring determines an assignment of directions $\{X^{\pm}, Y^{\pm}, Z^{\pm}\}$ to $E(G)$.

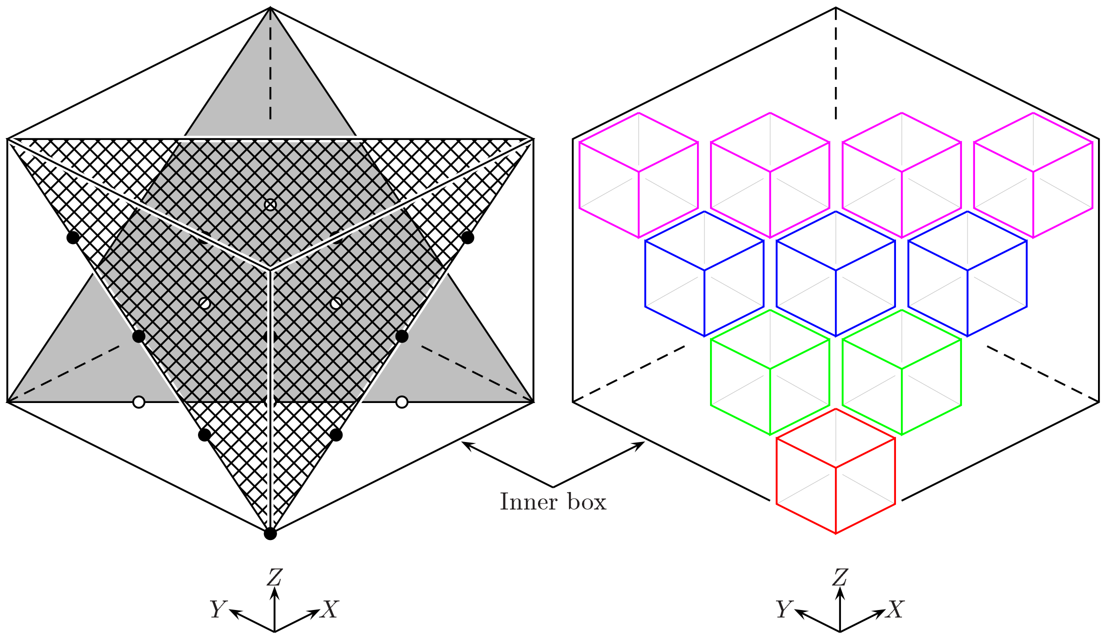


Figure 10.2: Non-Collinear Vertex Layout.

6. For each edge $vw \in E(G)$ in direction $d \in \{X^{\pm}, Y^{\pm}, Z^{\pm}\}$, arbitrarily assign unique ports at v and w in direction d with even j-coordinate and odd k-coordinate, where i, j and k are defined in Table 10.1 as functions of d. Call these the *usable* ports, as shown in Figure 10.3.

Table 10.1: Definition of i, j, k

d	i	j	k
$X^{\pm}$	X	Y	Z
$Y^{\pm}$	Y	Z	X
$Z^{\pm}$	Z	X	Y

7. Arbitrarily orient the edges of G.

8. For each direction $d \in \{X^{+}, Y^{+}, Z^{+}\}$ apply the following steps.

   (a) Construct a graph H with $V(H)$ corresponding to the edges of G in direction d. Add the edge $\{\overrightarrow{vw}, \overrightarrow{xy}\}$ to $E(H)$ if the port assigned to $\overrightarrow{vw}$ at v has the

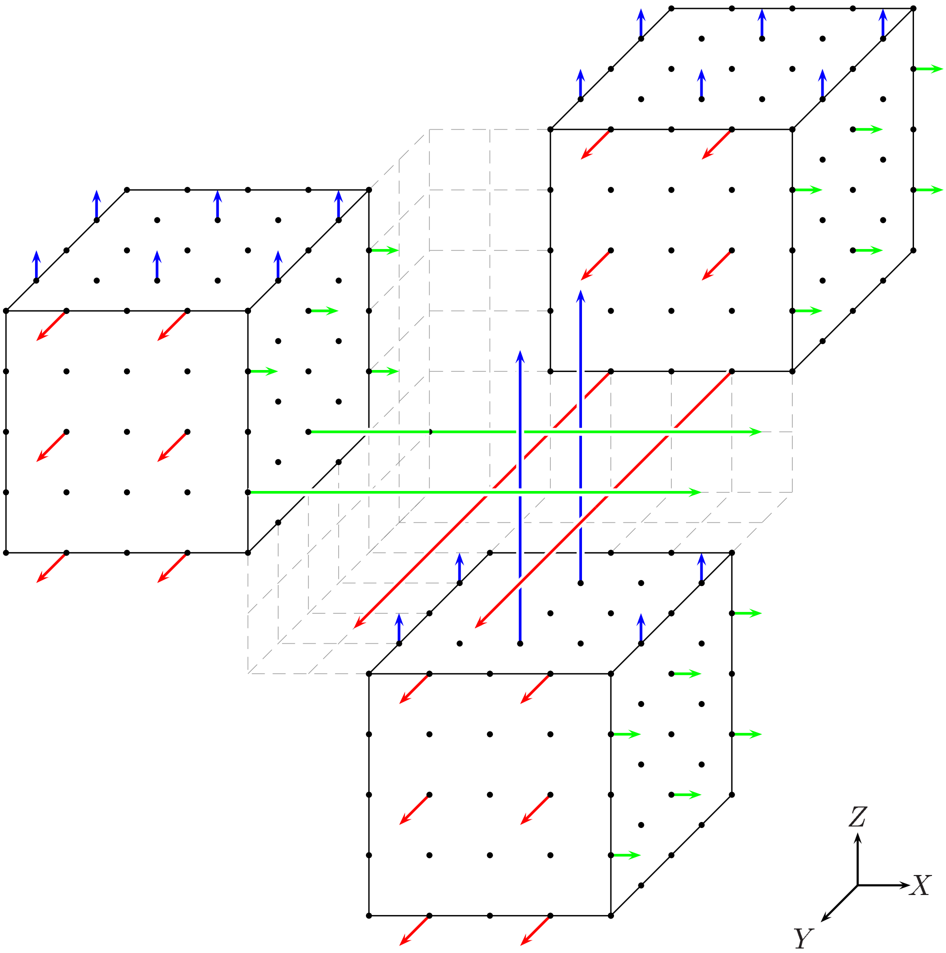


Figure 10.3: Usable ports on near-by vertices.

same j-coordinate as the port assigned to $\overrightarrow{xy}$ at x, or the port assigned to $\overrightarrow{vw}$ at w has the same k-coordinate as the port assigned to $\overrightarrow{xy}$ at y.

(b) Vertex-colour the graph H using the algorithm Greedy Vertex-Colour with colours $\{1, 2, \ldots, \Delta(H) + 1\}$ (see Section 2.2). For each vertex $v \in V(H)$ coloured α corresponding to an edge $\overrightarrow{vw}$, set the *height* $h(\overrightarrow{vw}) \leftarrow \alpha$.

(c) For each oriented edge $\overrightarrow{vw} \in E(G)$ in direction d, construct the edge route with (i, j, k) coordinates as follows. Suppose $\overrightarrow{vw}$ is assigned the port at (v_i, v_j, v_k) on v and the port at (w_i, w_j, w_k) on w. If $v_k = w_k$ then use the following 4-bend edge route, which extends a distance of $2h(\overrightarrow{vw})$ into the d-outer box, as illustrated in Figure 10.4 (and similarly if $v_j = w_j$).

$(v_i, v_j, v_k) \rightarrow (\Lambda(\lceil\sqrt{n}\rceil - 1) + 2h(\overrightarrow{vw}), v_j, v_k) \rightarrow$

$(\Lambda (\lceil\sqrt{n}\rceil - 1) + 2h(\overrightarrow{vw}), v_j, v_k + 1) \rightarrow$
$(\Lambda (\lceil\sqrt{n}\rceil - 1) + 2h(\overrightarrow{vw}), w_j, v_k + 1) \rightarrow$
$(\Lambda (\lceil\sqrt{n}\rceil - 1) + 2h(\overrightarrow{vw}), w_j, v_k) \rightarrow (w_i, w_j, v_k)$

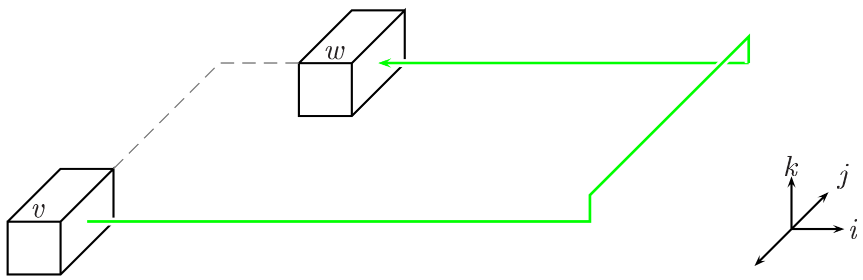


Figure 10.4: Edge route for vw if $v_k = w_k$.

Otherwise use the following 6-bend edge route illustrated in Figure 10.5.

$(v_i, v_j, v_k) \rightarrow (\Lambda (\lceil\sqrt{n}\rceil - 1) + 2h(\overrightarrow{vw}), v_j, v_k) \rightarrow$
$(\Lambda (\lceil\sqrt{n}\rceil - 1) + 2h(\overrightarrow{vw}), v_j, v_k + 1) \rightarrow$
$(\Lambda (\lceil\sqrt{n}\rceil - 1) + 2h(\overrightarrow{vw}), w_j + 1, v_k + 1) \rightarrow$
$(\Lambda (\lceil\sqrt{n}\rceil - 1) + 2h(\overrightarrow{vw}) + 1, w_j + 1, v_k + 1) \rightarrow$
$(\Lambda (\lceil\sqrt{n}\rceil - 1) + 2h(\overrightarrow{vw}) + 1, w_j + 1, w_k) \rightarrow$
$(\Lambda (\lceil\sqrt{n}\rceil - 1) + 2h(\overrightarrow{vw}) + 1, w_j, w_k) \rightarrow (w_i, w_j, w_k)$.

9. Repeat Step 8 for directions X^-, Y^- and Z^-, routing edges in the X^-, Y^- and Z^- outer boxes, respectively.

Theorem 10.1. *For every multigraph G, the algorithm* NON-COLLINEAR BOX-DRAWING *determines a 3-D orthogonal cube-drawing in $O(m^2)$ time, with $O((n\Delta)^{3/2})$ bounding box volume and six bends per edge route. Each vertex is 8-degree-restricted.*

Proof. The number of usable ports on a face of a vertex v is

$$\left\lceil\sqrt{\lceil \deg(v)/3\rceil}\right\rceil \left(\left\lceil\sqrt{\lceil \deg(v)/3\rceil}\right\rceil + 1\right) \geq \lceil \deg(v)/3\rceil + 1 \ .$$

By Theorem 8.2, there are at most $\lceil \deg(v)/3\rceil + 1$ edges incident to v in a given direction so there are enough usable ports at v. It is easily seen that no two vertices are intersected by a single grid-line.

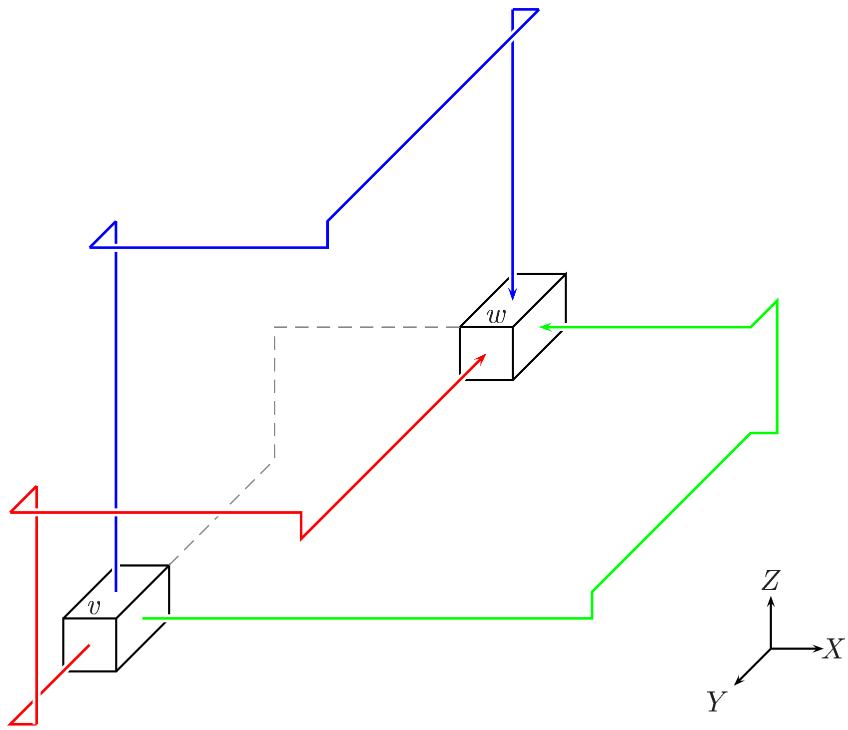


Figure 10.5: Edge routes vw in the non-collinear model.

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, so to show that the drawing is crossing-free, we need only consider intersections between non-unit-length segments. We distinguish between segments contained within the outer boxes, and the segments incident with vertices.

Clearly, segments contained in different outer boxes cannot intersect, and in an i-outer box, the j-parallel segments have even i-coordinate and the k-parallel segments have odd i-coordinate. Hence no two segments contained in an outer box can intersect.

Consider a segment contained in an i-outer box and a segment incident to a vertex. If the segment incident to a vertex is not in direction i then no intersection can occur. If this segment is in direction i then it has even j-coordinate and odd k-coordinate, whereas a j-parallel segment in the i-outer box has even k-coordinate, and a k-parallel segment in the i-outer box has odd j-coordinate. So a segment incident to a vertex and a segment contained in an outer box cannot intersect.

Now consider two segments incident to different vertices. (Segments incident to the same vertex are assigned unique ports so no intersection can occur.) If one such segment is in a positive direction and the other is in a negative direction then no intersection can occur. If the two segments are in the same direction then they are parallel so no intersection can occur. If the two segments are in directions i and j then one will have even k-coordinate and the other will have odd k-coordinate, so they cannot intersect. Therefore no two edge routes intersect.

The inner box has corners at

$$(0,0,0) \text{ and } \left(\Lambda\left(\lceil\sqrt{n}\rceil-1\right),\Lambda\left(\lceil\sqrt{n}\rceil-1\right),\Lambda\left(\lceil\sqrt{n}\rceil-1\right)\right) \ ,$$

so the width, depth and height of the inner box is $\Lambda\left(\sqrt{n}\right)$. The graph H has $\Delta(H) = 2\Lambda\lceil\sqrt{n}\rceil$, so the height of an edge is at most $4\Lambda\lceil\sqrt{n}\rceil$. Hence the bounding box has width, depth and height $8\Lambda\lceil\sqrt{n}\rceil$. Since $\Lambda = O\left(\sqrt{\Delta}\right)$, the bounding box volume is $O\left((n\Delta)^{3/2}\right)$.

For each vertex $v \in V(G)$, the surface (v) is

$$6\left(2\left\lceil\sqrt{\lceil\deg(v)/3\rceil}\right\rceil+1\right)^2 = 8\deg(v)+o\left(\deg(v)\right) \ .$$

So the drawing is 8-degree-restricted.

By Theorem 8.2, Step 5 of the algorithm takes $O\left(m^2\right)$ time. The six vertex-colourings of H each take $O(|E(H)|) = O(|V(H)|\Delta(H)) = O\left(m\sqrt{n\Delta}\right)$ time. Now, $\Delta \le m$, so assuming $m \le n$, we have $\Delta \le m^2/n$. So $\sqrt{n\Delta} \le m$ and $m\sqrt{n\Delta} \le m^2$. Hence Step 5 is most time-consuming step of the algorithm, and the total time taken is $O\left(m^2\right)$. □

For simple graphs we can use an equitable edge-colouring of G (see Corollary 8.1) instead of Algorithm Quasi-Equitable Edge-Colour in Step 5 of the above algorithm. The 'vertex spacing' is defined as $\Lambda = 2\left(\left\lceil\sqrt{\lceil\Delta(G)/6\rceil}\right\rceil+1\right)$ and each vertex is a

$$\left(2\left\lceil\sqrt{\lceil\deg(v)/6\rceil}\right\rceil+1\right)\times\left(2\left\lceil\sqrt{\lceil\deg(v)/6\rceil}\right\rceil+1\right)\times\left(2\left\lceil\sqrt{\lceil\deg(v)/6\rceil}\right\rceil+1\right)$$

cube. We obtain the following result.

Corollary 10.1. *For every graph with maximum degree Δ there exists a 3-D orthogonal cube-drawing with $O\left((n\Delta)^{3/2}\right)$ bounding box volume and at most six bends per edge route. Each vertex is 4-degree-restricted.*

For regular (multi)graphs the bounding box volume bound in Theorem 10.1 is $O(m^{3/2})$, which, by Theorem 3.2 is optimal for degree-restricted orthogonal box-drawings with bounded aspect ratio.

Open Problem 10.1. Can the algorithm Non-Collinear Box-Drawing be modified to produce box-drawings with bounding box volume $O((m+n)^{3/2})$? This amounts to finding a non-collinear vertex layout with an $O(\sqrt{m+n}) \times O(\sqrt{m+n}) \times O(\sqrt{m+n})$ inner box.

10.2 Point-Drawing Algorithm

We now present our algorithm for producing 3-D orthogonal point-drawings in the non-collinear model. This algorithm follows a similar approach as the previous box-drawing algorithm except that only the X^+, Y^+ and Z^+ outer boxes are used, and a cycle cover decomposition determines the port assignment instead of an equitable edge-colouring.

Algorithm 10.2. Non-Collinear Point-Drawing

Input: multigraph G with $\Delta(G) \leq 6$.
Output: 3-D orthogonal point-drawing of G.

---

1. Assign each vertex $v \in V(G)$ a unique pair $(x(v), y(v))$ with
$$0 \leq x(v), y(v) \leq \lceil \sqrt{n} \rceil - 1 \ .$$
2. For each vertex $v \in V(G)$, set $z(v) \leftarrow x(v) + y(v) \pmod{\lceil \sqrt{n} \rceil}$, and place v at $(4x(v), 4y(v), 4z(v))$.
3. Determine a cycle cover decomposition of G (see Theorem 2.1) and assign directions X^+, Y^+ and Z^+ to the edges appearing in the first, second and third cycle covers, respectively.
4. Considering v to be represented by the $3 \times 3 \times 3$ box centred at v, determine edge routes as described in Steps 6-8 of Algorithm 10.1 Non-Collinear Box-Drawing.

5. For each vertex $v \in V(G)$ connect the edges incident with v from the surface of the $3 \times 3 \times 3$ box to the point representing v, as shown in Figure 10.6.

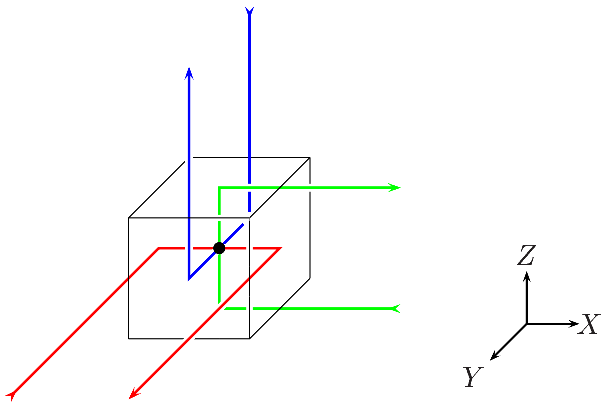


Figure 10.6: A vertex inside a 3×3 box.

Theorem 10.2. *The algorithm* Non-Collinear Point-Drawing *determines in* $O(n^{3/2})$ *time a 3-D orthogonal point-drawing of the given graph* G*, with* $O(n^{3/2})$ *bounding box volume and at most 8 bends per edge route.*

Proof. This result follows immediately from Theorem 10.1 and the observations that edges will be routed by algorithm Non-Collinear Box-Drawing as indicated in Figure 10.6, and one extra bend is added to each end of an edge route. □

Chapter 11

Multi-Dimensional Orthogonal Point-Drawing

In this chapter we study multi-dimensional orthogonal point-drawings of graphs, as suggested by Liu [145, Note 8.5.2]. In particular, we present an algorithm for generating minimum-dimensional orthogonal point-drawings of arbitrary degree graphs in the non-collinear coplanar vertex layout model with at most six bends per edge. We also construct minimum-dimensional orthogonal point-drawings of K_n with at most two bends per edge, a result first presented in [219].

We say a D-dimensional orthogonal point-drawing of a graph G is *minimum-dimensional* if there does not exist a $(D-1)$-dimensional orthogonal point-drawing of G. Consider an orthogonal point-drawing of an arbitrary degree graph G. At a vertex in the D-dimensional orthogonal grid there are $2D$ ports, so an orthogonal point-drawing of G requires at least $\lceil \Delta(G)/2 \rceil$ dimensions. We shall show that only a few graphs G do not have an orthogonal point-drawing in $\lceil \Delta(G)/2 \rceil$ dimensions. We define the *bend number* of G to be the minimum integer b such that there exists a minimum-dimensional point-drawing of G with at most b bends per edge route.

Trivially K_1 and K_2 have minimum-dimensional orthogonal point-drawings without any bends (in the 0- and 1-dimensional grids, respectively). K_3 is our first example of a graph G which does not have an orthogonal point-drawing in $\lceil \Delta(G)/2 \rceil$ $(= 1)$

dimensions. The 1-bend 2-D orthogonal point-drawing of K_3 establishes that the bend number of K_3 is one. In fact, all cycles C_n $(n \geq 3)$ do not have an orthogonal point-drawing in $\lceil \Delta(C_n)/2 \rceil$ $(= 1)$ dimensions. C_n does have a 1-bend 2-D orthogonal point-drawing so the bend number of C_n is one.

If we define 'minimum-dimensional' so that edge-crossings are allowed in 2-D orthogonal point-drawings, by the algorithms of Biedl and Kant [28] and Papakostas and Tollis [165], all maximum degree four graphs have bend number at most two. If 2-D drawings must be crossing-free, then by the algorithm of Biedl and Kant [28], the bend number of a planar graph with maximum degree at most four is at most two (except the octahedron graph which requires a 3-bend edge route [91]).

By Theorem 5.4, graphs with maximum degree at most five have a 2-bend 3-D orthogonal point-drawing, so the bend number of such graphs is at most two. Maximum degree six multigraphs have a 3-bend 3-D orthogonal point-drawing (see Section 5.5), so maximum degree six multigraphs have bend number at most three.

In Section 11.1 we shall show that the bend number of K_n is two. To do so, we initially prove a tight bound for the number of dimensions required for a 1-bend orthogonal point-drawing of K_n. We then construct minimum-dimensional point-drawings of K_n with at most two bends per edge route, a result which establishes the bend number of K_n to be two except for some isolated cases. The algorithm presented in Section 11.2 establishes an upper bound of six for the bend number of an arbitrary multigraph.

11.1 Drawings of K_n

We now prove a lower bound for the number of dimensions required for a 1-bend orthogonal point-drawing of K_n.

Theorem 11.1. *For $n \geq 3$, a 1-bend orthogonal point-drawing of K_n requires at least $n-1$ dimensions.*

Proof. To construct a 1-bend $(n-1)$-dimensional point-drawing of K_n, for each dimension i, $1 \leq i \leq n-1$, place a vertex v_i at 1 on the i-axis, and place the remaining vertex at the origin. Connect each v_i to the origin by a 0-bend edge route, and connect vertices v_i and v_j by a 1-bend edge route through $(0,\ldots,0,1,0,\ldots,0,1,0,\ldots,0)$ where the 1's appear in the i- and j-coordinates.

Suppose there is a $(n-2)$-dimensional orthogonal point-drawing of K_n with at most one bend per edge route. Let v be some vertex of K_n. Define T_0 to be the set of dimensions i, $1 \leq i \leq n-2$, such that no edge route uses port$(v,+i)$ or port$(v,-i)$. Let T_1 be the set of dimensions with exactly one port at v in use, and let T_2 be the set of dimensions with both ports at v in use. Clearly $|T_0|+|T_1|+|T_2| = n-2$ and $0|T_0|+1|T_1|+2|T_2| = n-1$, implying $|T_0| = |T_2|-1$ and $|T_2| \geq 1$.

Let $i \in T_2$ and let va and vb be the edges assigned port$(v,-i)$ and port$(v,+i)$, respectively. Now, va and vb cannot both be 0-bend edge routes, as otherwise ab would have to be a 2-bend edge route. Suppose one of va or vb is a 0-bend edge route and the other is a 1-bend edge route, as shown in Figure 11.1. Let j be the direction of the second segment of the 1-bend edge. Clearly, no edge vx could be routed with port(v,j) as otherwise there would be no possible 0- or 1-bend edge route for xa nor xb. If vx is routed with port$(v,-j)$ then xa or xb would need two bends, so $j \in T_0$.

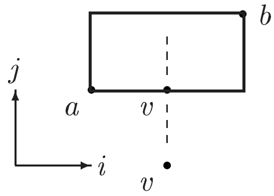


Figure 11.1: A 0-bend and a 1-bend edge

If the edge routes va and vb both have one bend then, as in Figure 11.2, for ab to have a 0- or 1-bend edge route, the second segments of va and vb must point in the same direction j, as in cases (c) and (d). By the same argument as before, this implies that $j \in T_0$.

Suppose $|T_2| > 1$ and dimension $k \in T_2 \setminus \{i\}$. Let vc and vd be the edges routed using port$(v,+k)$ and port$(v,-k)$, respectively. For ac, ad, bc and bd to have 1-bend edge routes, the edges va, vb, vc and vd all must have one bend and their second segments must point in the same direction and have the same length, as in Figure 11.3. Therefore ab and cd must intersect, so $|T_2| = 1$.

$|T_2| = 1$ implies $|T_0| = 0$, but $j \in T_0$, which is a contradiction. Therefore K_n does

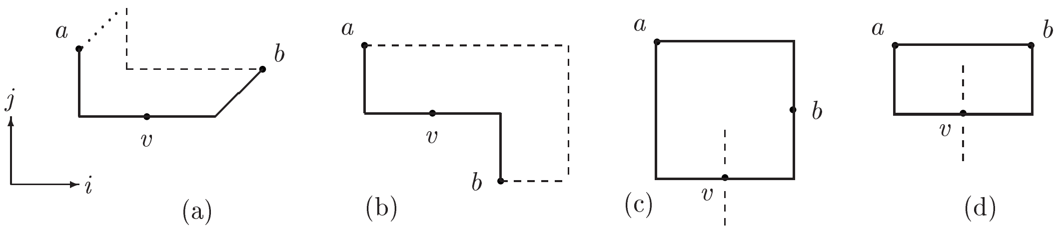


Figure 11.2: Two 1-bend edges

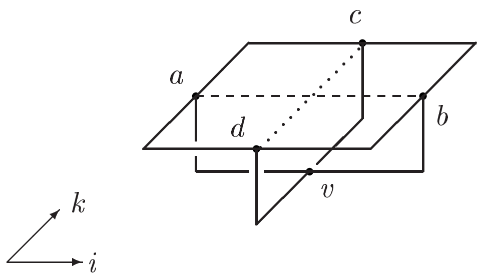


Figure 11.3: $i, k \in T_2$

not have a $(n-2)$-dimensional orthogonal point-drawing with at most one bend per edge route. $\square$

A minimum-dimensional orthogonal point-drawing of K_n has at least $\lceil \Delta(K_n)/2 \rceil = \lfloor n/2 \rfloor$ dimensions. For $n \geq 4$, we have $n - 1 > \lfloor n/2 \rfloor$, so a minimum-dimensional orthogonal point-drawing of K_n ($n \geq 4$) requires at least two bends in some edge route. There is a 2-D 2-bend orthogonal point-drawing of K_4, so the bend number of K_4 is two. K_5 also has a 2-D 2-bend orthogonal point-drawing (of course, with crossings), so it too has bend number two. If we do not allow crossings in 2-D drawings then K_5 requires three dimensions. By Theorem 11.1 a 3-D orthogonal point-drawing of K_5 still requires an edge route with at least two bends. A 2-bend 3-D orthogonal point-drawing of K_5 is provided in Figure 2.3(b) (on page 28). We now construct 2-bend minimum-dimensional orthogonal point-drawings of K_n for $n \geq 6$.

Theorem 11.2. *For every $n \geq 6$, the bend number of K_n is 2.*

Proof. We initially consider the case of odd n. In Figure 3.6 there is a 2-bend 3-D orthogonal point-drawing of K_7, so the result is true for $n = 7$. We now construct a $((n-1)/2)$-dimensional 2-bend point-drawing of K_n for odd $n \geq 9$. Let the vertex set of K_n be

$$V(K_n) = \{v_1, v_2, \ldots, v_7\} \cup \{a_i, b_i : 4 \leq i \leq (n-1)/2\} \quad .$$

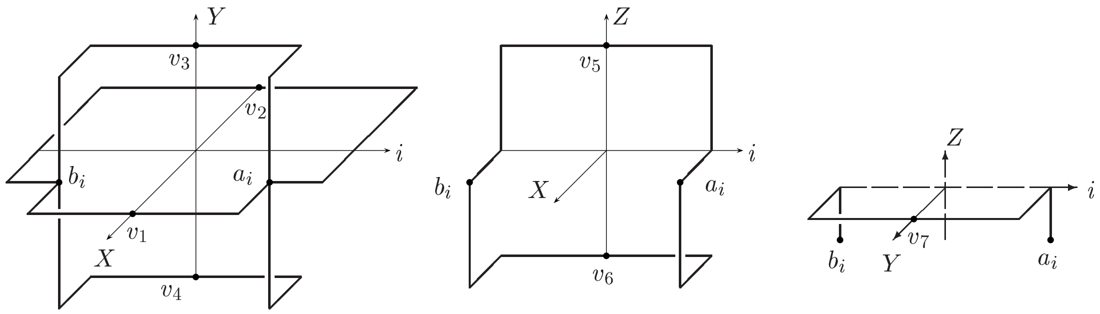
The K_7 subgraph induced by the vertices $\{v_1, v_2, \ldots, v_7\}$ is drawn with two bends per edge route as in Figure 3.6 (on page 53). In particular we place the $\{v_1, v_2, \ldots, v_7\}$ as follows.

$$\begin{array}{llll}
v_1: & (2,0,0,0,\ldots,0) & v_2: & (-2,0,0,0,\ldots,0) \\
v_3: & (0,2,0,0,\ldots,0) & v_4: & (0,-2,0,0,\ldots,0) \\
v_5: & (0,0,2,0,\ldots,0) & v_6: & (0,0,-2,0,\ldots,0) \\
v_7: & (1,1,1,0,\ldots,0) \; . & &
\end{array}$$

For each i, $4 \leq i \leq (n-1)/2$, place a_i and b_i at

$$a_i: (1,0,0,\ldots,2,0,0,\ldots,0) \qquad b_i: (1,0,0,\ldots,-2,0,0,\ldots,0)$$

(with the 2 and -2 at coordinate i). The edge $a_i v_j$ and $b_i v_j$, $4 \leq i \leq (n-1)/2$, $1 \leq j \leq 7$, are routed according to Figure 11.4.



(a) In the $Z = 0$ hyperplane (b) In the $Y = 0$ hyperplane (c) In the $X = 1$ hyperplane

Figure 11.4: Edge routes $a_i v_j$ and $b_i v_j$.

The edges $a_i b_i$, $a_i a_{i+1}$, $b_i a_{i+1}$, $a_i b_{i+1}$ and $b_i b_{i+1}$, $4 \leq i \leq (n-3)/2$, are routed according to Figure 11.5(a). The edges $a_i a_j$, $b_i a_j$, $a_i b_j$, and $b_i b_j$, $4 \leq i \leq (n-3)/2$, $i+2 \leq j \leq (n-1)/2$ are routed according to Figure 11.5(b).

A straight line edge route from $a_{(n-1)/2}$ to $b_{(n-1)/2}$ passing through the vacant grid-point $(1,0,0,\ldots,0)$ completes the drawing.

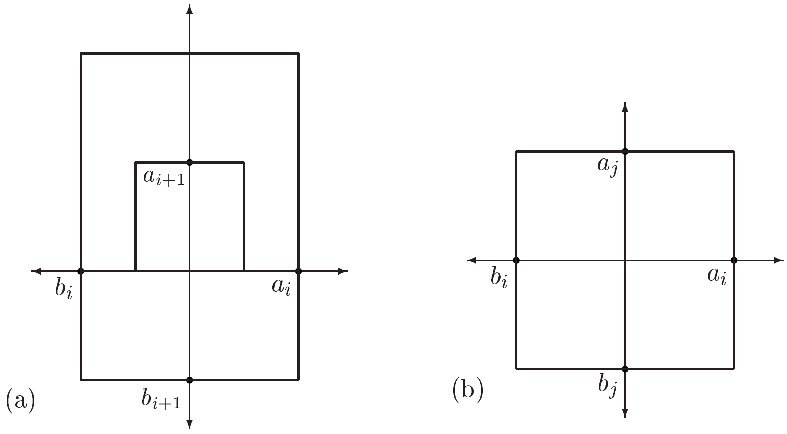


Figure 11.5: Edge routes in the $X = 1$ hyperplane.

It is easily seen that a unique port assignments are determined by this edge routing scheme. The grid-points contained in edge routes described in Figure 11.4 only contain grid-points with a non-zero i coordinate (except for the vertices themselves). So such edge routes cannot cross an edge route in the K_7 subgraph induced by $\{v_1, v_2, \ldots, v_7\}$. Similarly, an edge route $a_i v_k$ or $b_i v_k$ cannot cross an edge route $a_j v_k$ or $b_j v_k$ $(1 \le k \le 7)$.

Except for the grid-points $(1, 0, \ldots, 0, 4, 0, \ldots, 0)$ (in edge $a_i b_i$), $(1, 0, \ldots, 0, 1, 0, \ldots, 0)$ (in edge $a_i a_{i+1}$) and $(1, 0, \ldots, 0, -1, 0, \ldots, 0)$ (in edge $b_i a_{i+1}$), the edge routes described in Figure 11.5 only contain grid-points with non-zero i and j coordinates. They will therefore not cross other edges. By checking grid-points in the $X = 1$ hyperplane it is easily seen that these particular grid-points are not in any other edge routes. So no two edge routes cross.

Hence there is a 2-bend minimum-dimensional orthogonal point-drawing of K_n for odd $n \ge 7$. In fact there are $O(n^2)$ 1-bend edge routes and only $O(n)$ 2-bend edge routes. For even $n \ge 6$, removing a single vertex from the drawing of K_{n+1} provides a minimum-dimensional 2-bend orthogonal point-drawing of K_n. By Theorem 11.1, $n-1$ dimensions are required for a 1-bend point-drawing of K_n, so the bend number of K_n is 2, for $n \ge 6$. $\square$

11.2 Algorithm

As mentioned by Eades *et al.* [87], their 3-BENDS algorithm easily generalises to give an algorithm for producing a minimum-dimensional orthogonal point-drawing of a graph G with at most $\lceil \Delta(G)/2 \rceil$ bends per edge route. This algorithm places the vertices along the main diagonal of a $\lceil \Delta(G)/2 \rceil$-dimensional hypercube. Here we place the vertices along a 2-D diagonal within $\lceil \Delta(G)/2 \rceil$-dimensional space, and use at most six bends per edge route[1].

Algorithm 11.1. MINIMUM-DIMENSIONAL POINT-DRAWING

Input: A multigraph G with maximum degree $\Delta(G) \geq 5$.
Output: A minimum-dimensional orthogonal point drawing of G.

---

1. Determine G' and its cycle covers $C_1, C_2, \ldots, C_d$ where $d = \lceil \Delta(G)/2 \rceil$ (see Theorem 2.1).

2. Arbitrarily assign the numbers $\{1, 2, \ldots, n\}$ to the vertices of G.
   (We shall refer to a vertex by its number.)

3. Position vertex a at $(2a, 3a, 0, \ldots, 0) \in \mathbb{Z}^d$.

4. Construct edge routes for each arc in G', as described below.

5. For each edge of G, draw the edge route of the corresponding arc in G'.

The following method used to classify arcs according to a vertex ordering is due to Eades *et al.* [86, 87]. Consider an arc $ab \in E(G')$ in cycle cover C_1, and suppose bc is the next arc in the cycle containing ab. We route the arc ab depending on the relative values of a, b and c. In the following figures, the arrow head indicates the port at b to be assigned to the arc bc.

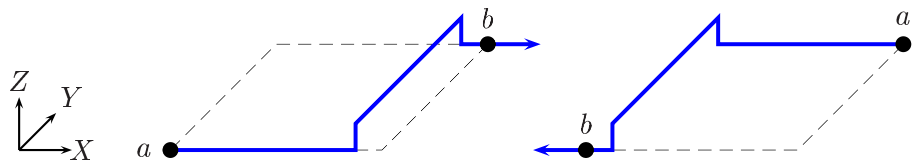
[1] In [219] it was erroneously stated that using a 3-D diagonal vertex layout, five bends per edge route was possible.

Case 1.1: If $a < b < c$ then we say ab is *normal increasing*. As in Figure 11.6(a), route ab with the 4-bend edge:

$$(2a, 3a, 0, 0, \dots, 0) \to (2b-1, 3a, 0, 0, \dots, 0) \to (2b-1, 3a, 1, 0, \dots, 0)$$
$$\to (2b-1, 3b, 1, 0, \dots, 0) \to (2b-1, 3b, 0, 0, \dots, 0) \to (2b, 3b, 0, 0, \dots, 0)$$

Case 1.2: If $a > b > c$ then we say ab is *normal decreasing*. As in Figure 11.6(b), route ab with the 4-bend edge:

$$(2a, 3a, 0, 0, \dots, 0) \to (2b+1, 3a, 0, 0, \dots, 0) \to (2b+1, 3a, 1, 0, \dots, 0)$$
$$\to (2b+1, 3b, 1, 0, \dots, 0) \to (2b+1, 3b, 0, 0, \dots, 0) \to (2b, 3b, 0, 0, \dots, 0)$$



(a) Increasing (b) Decreasing

Figure 11.6: Normal arcs ab in C_1.

Case 1.3: If $a < b > c$ then we say ab is *increasing to a local maximum*. As in Figure 11.7(a), route ab with the 4-bend edge:

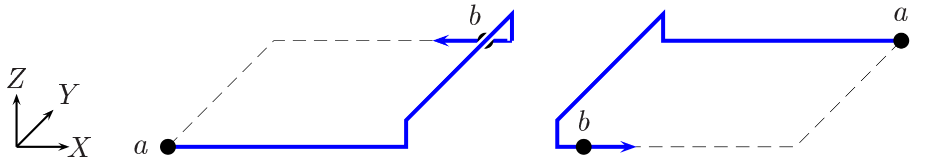
$$(2a, 3a, 0, 0, \dots, 0) \to (2b+1, 3a, 0, 0, \dots, 0) \to (2b+1, 3a, 1, 0, \dots, 0)$$
$$\to (2b+1, 3b, 1, 0, \dots, 0) \to (2b+1, 3b, 0, 0, \dots, 0) \to (2b, 3b, 0, 0, \dots, 0)$$

Case 1.4: If $a > b < c$ then we say ab is *decreasing to a local minimum*. As in Figure 11.7(b), route ab with the 4-bend edge:

$$(2a, 3a, 0, 0, \dots, 0) \to (2b-1, 3a, 0, 0, \dots, 0) \to (2b-1, 3a, 1, 0, \dots, 0)$$
$$\to (2b-1, 3b, 1, 0, \dots, 0) \to (2b-1, 3b, 0, 0, \dots, 0) \to (2b, 3b, 0, 0, \dots, 0)$$

Observe that all arcs ab in C_1 are routed using the X-ports at a and b. Now consider an arc $ab \in E(G')$ in cycle cover C_2 and, as before, suppose bc is the next arc in the cycle containing ab.

Case 2.1: If ab is normal increasing then, as in Figure 11.8(a), route ab with the 5-bend edge:



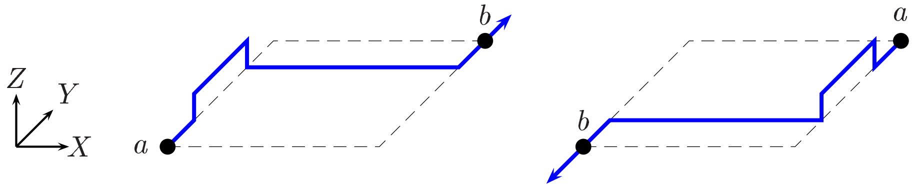
(b) Increasing to a local maximum (b) Decreasing to a local minimum

Figure 11.7: Local min/max arcs ab in C_1.

$(2a, 3a, 0, 0, \dots, 0) \to (2a, 3a + 1, 0, 0, \dots, 0) \to (2a, 3a + 1, 1, 0, \dots, 0) \to$
$(2a, 3b - 1, 1, 0, \dots, 0) \to (2a, 3b - 1, 0, 0, \dots, 0) \to$
$(2b, 3b - 1, 0, 0, \dots, 0) \to (2b, 3b, 0, 0, \dots, 0)$

Case 2.2: If ab is normal decreasing then, as in Figure 11.8(b), route ab with the 5-bend edge:

$(2a, 3a, 0, 0, \dots, 0) \to (2a, 3a - 1, 0, 0, \dots, 0) \to (2a, 3a - 1, 1, 0, \dots, 0) \to$
$(2a, 3b + 1, 1, 0, \dots, 0) \to (2a, 3b + 1, 0, 0, \dots, 0) \to$
$(2b, 3b + 1, 0, 0, \dots, 0) \to (2b, 3b, 0, 0, \dots, 0)$



(a) Increasing (b) Decreasing

Figure 11.8: Normal arcs ab in C_2.

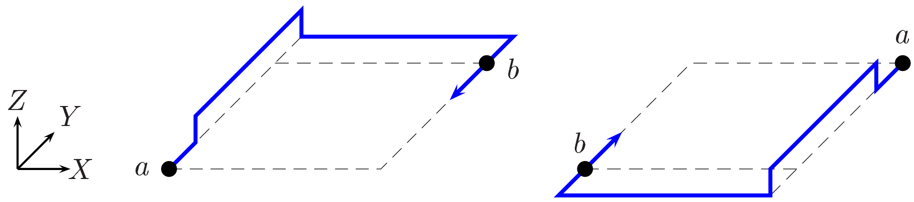
Case 2.3: If ab is increasing to a local maximum then, as in Figure 11.9(a), route ab with the 5-bend edge:

$(2a, 3a, 0, 0, \dots, 0) \to (2a, 3a + 1, 0, 0, \dots, 0) \to (2a, 3a + 1, 1, 0, \dots, 0) \to$
$(2a, 3b + 1, 1, 0, \dots, 0) \to (2a, 3b + 1, 0, 0, \dots, 0) \to$
$(2b, 3b + 1, 0, 0, \dots, 0) \to (2b, 3b, 0, 0, \dots, 0)$

Case 2.4: If ab is decreasing to a local minimum then, as in Figure 11.9(b), route ab with the 5-bend edge:

$(2a, 3a, 0, 0, \dots, 0) \to (2a, 3a - 1, 0, 0, \dots, 0) \to (2a, 3a - 1, 1, 0, \dots, 0) \to$

$(2a, 3b-1, 1, 0, \ldots, 0) \to (2a, 3b-1, 0, 0, \ldots, 0) \to$
$(2b, 3b-1, 0, 0, \ldots, 0) \to (2b, 3b, 0, 0, \ldots, 0)$



(a) Increasing to a local maximum (b) Decreasing to a local minimum

Figure 11.9: Local min/max arcs ab in C_2.

Observe that arcs in C_2 are assigned the Y-ports at both ends. We now describe how to route arcs in cycle cover C_j, $3 \le j \le \lceil \Delta(G)/2 \rceil$. Suppose $(a_1, a_2, \ldots, a_k)$ is a cycle in C_j. As illustrated in Figure 11.10, the incoming arc at a vertex a_i uses the $-j/+j$ port and the outgoing arc uses the $+j/-j$ port, for odd/even i.

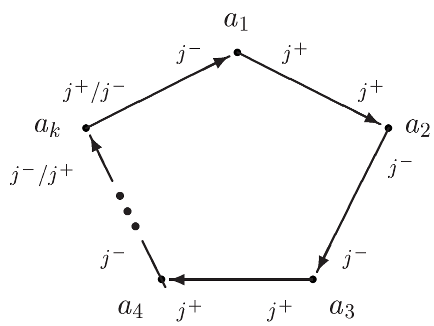


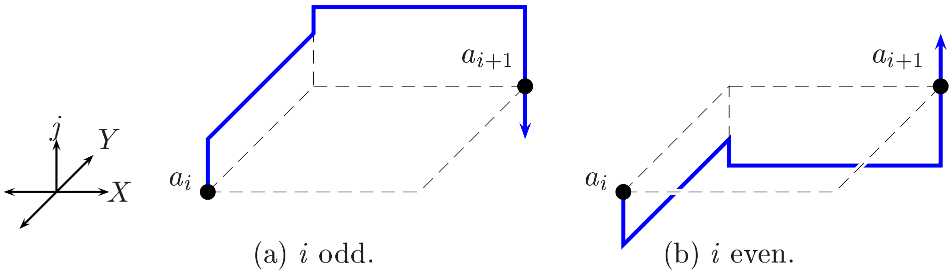
Figure 11.10: Port assignment for a cycle in C_j, $j \ge 3$.

- For each odd i, $1 \le i \le k-1$, as in Figure 11.11(a), route the arc $a_i a_{i+1}$ with the 4-bend edge:

  $(2a_i, 3a_i, 0, \ldots, 0) \to (2a_i, 3a_i, 0, \ldots, 2, 0, \ldots, 0) \to$
  $(2a_i, 3a_{i+1}, 0, \ldots, 0, 2, 0, \ldots, 0) \to (2a_i, 3a_{i+1}, 0, \ldots, 0, 3, 0, \ldots, 0) \to$
  $(2a_{i+1}, 3a_{i+1}, 0, \ldots, 0, 3, 0 \ldots, 0) \to (2a_{i+1}, 3a_{i+1}, 0 \ldots, 0)$

- For each even i, $2 \le i \le k$, as in Figure 11.11(b), route the arc $a_i a_{i+1}$ (or $a_i a_1$ if $i = k$) with the 4-bend edge:

$(2a_i, 3a_i, 0, \ldots, 0) \to (2a_i, 3a_i, 0, \ldots, 0, -2, 0, \ldots, 0) \to$
$(2a_i, 3a_{i+1}, 0, \ldots, 0, -2, 0, \ldots, 0) \to (2a_i, 3a_{i+1}, 0, \ldots, 0, -3, 0, \ldots, 0) \to$
$(2a_{i+1}, 3a_{i+1}, 0, \ldots, 0, -3, 0, \ldots, 0) \to (2a_{i+1}, 3a_{i+1}, 0, \ldots, 0)$



(a) i odd. (b) i even.

Figure 11.11: Arc a_ia_{i+1} in cycle cover C_j, $j \geq 3$.

- If k is odd then, as in Figure 11.12, route the arc a_ka_1 with the following 6-bend edge. If $j = D(= \lceil \Delta(G)/2 \rceil)$ then dimension $j+1$ is 3.

  $(2a_k, 3a_k, 0, \ldots, 0) \to (2a_k, 3a_k, 0, \ldots, 0, 2, 0, 0, \ldots, 0) \to$
  $(2a_k, 3a_k, 0, \ldots, 0, 2, 2, 0, \ldots, 0) \to (2a_k, 3a_1, 0, \ldots, 0, 2, 2, 0, \ldots, 0) \to$
  $(2a_k, 3a_1, 0, \ldots, 0, -3, 2, 0, \ldots, 0) \to (2a_k, 3a_1, 0, \ldots, 0, -3, 0, 0, \ldots, 0) \to$
  $(2a_1, 3a_1, 0, \ldots, 0, -3, 0, 0, \ldots, 0) \to (3a_1, 3a_1, 0, \ldots, 0, \ldots, 0)$

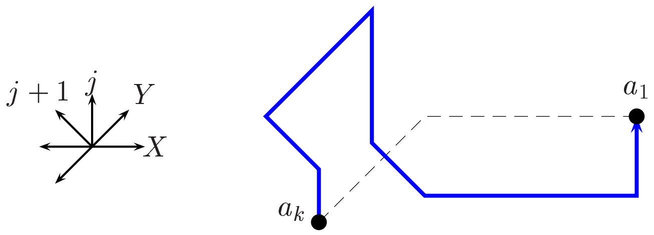


Figure 11.12: Arc a_ka_1 (k odd) in cycle cover C_j, $j \geq 3$.

Theorem 11.3. *The algorithm* MINIMUM-DIMENSIONAL POINT-DRAWING *determines a minimum-dimensional 6-bend orthogonal point-drawing of G, which can be computed in $O(\Delta^2 n)$ time.*

Proof. The cycle cover decomposition gives for each vertex exactly one incoming arc and one outgoing arc in each of the d cycle covers. Observe that arcs in cycle cover C_j use the j-ports at each vertex. Hence a valid port assignment has been determined, and the first segments of edges incident to a particular vertex do not intersect (except at the vertex itself).

Consider edge routes in cycle covers C_1 and C_2. The X-parallel segments lie in the $(Z = 0)$-plane and Y-parallel segments lie in the $Z = 1$ plane, so a X-parallel segment cannot intersect a Y-parallel segment. Note that the X-parallel segments of an arc ab in C_1 lie in the YZ-plane containing a, and the X-parallel segments of an arc ab in C_2 lie in the YZ-plane offset from b by a distance of 1. Similarly for Y-parallel segments, so by the spacing between the vertices, no two edge routes in C_1 or C_2 can intersect.

Now consider edge routes in a cycle cover C_j, $j \geq 3$. Apart from the point $(2a_k, 3a_1, 0, \ldots, 0, 2, 0, \ldots, 0)$ for some arc $a_k a_1$ in C_j (k odd) with the 2 in coordinate $j + 1$, grid-points in edge routes in C_j have non-zero j-coordinate and a zero k-coordinate for each $k \geq 3$ ($k \neq j$). Hence edge routes in C_j and C_k ($j \neq k$, $j, k \geq 3$) do not intersect. X-parallel segments of an edge route in C_j have a j-coordinate of ± 3, and Y-parallel segments of edge routes in C_j have j-coordinate of ± 2, so no two edges in a cycle cover C_j can intersect. The grid-point $(2a_k, 3a_1, 0, \ldots, 2, 0, \ldots, 0)$ with the 2 in coordinate $j + 1$ can only be in the arc $a_k a_1$ (k odd) in cycle cover C_j, so it too does not intersect any other edge routes.

Hence the drawing is crossing-free, and each edge route has at most six bends. By Theorem 2.1, the cycle cover decomposition and hence the whole drawing can computed in $O(\Delta^2 n)$ time. $\square$

Part IV

Conclusion

Chapter 12

Conclusion

In this conclusion we summarise the main achievements of this thesis, the open problems in 3-D orthogonal graph drawing which have been identified, and discuss avenues for future work in 3-D graph drawing.

This thesis has investigated problems related to the automatic generation of 3-D orthogonal graph drawings. Orthogonal graph drawing has applications in VLSI circuit design and software engineering, for example. The methods developed have also been applied to 2-D orthogonal graph drawing and generalised to multi-dimensional space.

12.1 Models and Algorithms

The following models for 3-D orthogonal graph drawing have either been introduced or extended in this thesis. The algorithms in this thesis, which typically have polynomial time complexity, explore tradeoffs between the established aesthetic criteria for measuring the quality of the produced drawings.

General Position Vertex Layout Model:

A 3-D orthogonal graph drawing is in the general position model if no two vertices are intersected by a single grid-plane; e.g., by positioning the vertices along the main diagonal of cube. We have presented algorithms for producing orientation-independent drawings in the general position model with few bends. A disadvantage of this model is that the volume of drawings is necessarily large.

We have described an algorithm which, given a fixed general position vertex layout of an arbitrary degree graph, constructs a general position drawings with bounded degree-restriction and bounded aspect ratio (Algorithm 7.4). This algorithm is also applicable in a 2-D or multi-dimensional setting. Using a balanced vertex layout, our algorithm produces drawings with the smallest known bounds for the degree-restriction of vertices (Algorithm 7.6).

Our algorithm for producing 3-D orthogonal point-drawings of maximum degree six graphs establishes the best known upper bound for the total number of bends in 3-D orthogonal point-drawings (Algorithm 5.8). Another algorithm establishes the best known upper bound for the volume of 3-D orthogonal point-drawings with three bends per edge route (Algorithm 5.11).

Coplanar Vertex Layout Model:

A 3-D orthogonal graph drawing is in the coplanar vertex layout model if there exists a grid-plane which intersects all vertices. We have considered two variations of this model, namely the *non-collinear* coplanar model and the coplanar *grid* model. Our algorithms produce orthogonal drawings in these models with few bends and small volume, respectively. A disadvantage of the coplanar vertex layout model is that the drawings produced are necessarily orientation-dependent.

Our algorithm for orthogonal drawing in the non-collinear coplanar model exploits a book embedding to obtain 1-bend drawings, which for sparse graphs have less volume than existing methods for 1-bend drawing (Algorithm 9.1).

We have presented two algorithms for producing 3-D orthogonal box-drawings in the coplanar grid model. The first algorithm produces drawings with optimal volume for regular graphs (Algorithm 9.2). The second algorithm produces degree-restricted 3-D orthogonal cube-drawings with optimal volume (Algorithm 9.3).

Non-Collinear Vertex Layout Model:

A 3-D orthogonal graph drawing is in the non-collinear vertex layout model if no two vertices are intersected by a single grid-line. In this model, we present an algorithm for producing 3-D orthogonal box-drawings with optimal volume for regular graphs

(Algorithm 10.1). This algorithm is then used as the basis for producing 3-D orthogonal point-drawings with optimal volume (Algorithm 10.2). These are the only known algorithms for producing orientation-independent 3-D orthogonal graph drawings with optimal volume.

12.2 Methods

As part of our investigation into orthogonal graph drawings, we have developed and extended existing methods which may be of independent interest. These include:

- algorithms for the balanced vertex ordering problem, which we use as the basis for determining general position vertex layouts;
- an algorithm for equitable edge-colouring of multigraphs, which we use to determine port assignments;
- an approach to port assignment based on arc-colouring;
- the use of vertex-colouring to determine the heights of edge routes; and
- an exact algorithm for the maximum clique problem, which we use for searching for 2-bend point-drawings.

12.3 Open Problems

In the course of this thesis we have raised many open problems, including the following.

- Does every graph have a degree-restricted 3-D orthogonal box-drawing with at most one bend per edge route? Does every graph have a 3-D orthogonal box-drawing with $O(n^2\sqrt{m})$ volume and at most one bend per edge route? (See Sections 3.5.2 and 9.1.)
- Does every graph have a 3-D orthogonal box-drawing with $O(m\sqrt{n})$ volume and at most three bends per edge route? (See Sections 3.5.2 and 9.2.)

- Does every graph have a degree-restricted 3-D orthogonal cube-drawing with $O\big((m+n)^{3/2}\big)$ volume and at most five bends per edge route? (See Sections 3.5.2 and 9.3.)

- Does every graph with maximum degree six have a 3-D orthogonal point-drawing with at most two bends per edge route? [86, 87] (See Sections 3.5.1 and 5.6.1.)

- Does every graph with maximum degree six have a 3-D orthogonal point-drawing with $O\big(n^{3/2}\big)$ volume and at most six bends per edge route? (See Section 3.5.1.)

- Can the Topology-Shape-Metrics approach be applied to 3-D orthogonal graph drawing? For example, given a (linkless) 3-D embedding of a graph with maximum degree six, can an embedding-preserving 3-D orthogonal point-drawing with the minimum number of bends be determined in polynomial time? (See Section 3.2.2.) Note that a 3-D graph embedding can be represented by a 2-D projection for which 'over/under' crossings are specified.

- Develop bounds for the aesthetic criteria, besides bounding box volume and the number of bends, of 3-D orthogonal graph drawings. For example, the total edge length and the maximum edge length could be studied.

12.4 Future Work

The development of three-dimensional graph drawing is in its infancy. While algorithms for 3-D orthogonal graph drawing have been developed which optimise certain aesthetic criteria, most notably the bounding box volume, it is reasonable to ask whether the drawings produced are feasible for visualisation purposes. We now outline avenues of research aimed at producing more readable 3-D graph drawings.

Firstly, the question of what are the properties of 3-D graph drawings which are most appropriate for visualisation purposes has not been addressed in any scientific manner. It is unrealistic to assume that the aesthetic criteria for 2-D graph drawings automatically apply in a three-dimensional setting. In particular, the experiments of Purchase *et al.* [176] and Purchase [175] confirm that the minimisation of crossings is an important aesthetic criterion for 2-D graph drawings, however in three dimensions all graphs can be drawn without crossings. Also, it would be interesting to determine if 3-D graph drawings are better for visualisation purposes than their two-dimensional counterparts (see Ware and Franck [213] for a preliminary study).

A critical issue in 3-D graph visualisation is the question of how to display a graph drawing on a computer screen. Many issues from computer graphics, such as rendering and shading, immediately arise. A system for displaying, and interacting with, 3-D graph drawings needs to be developed. Such a system could incorporate methods for finding viewpoints of 3-D drawings with few occlusions (see Kamada and Kawai [123], Bose *et al.* [39], Eades *et al.* [81] and Houle and Webber [120]).

As well as solving the open problems discussed in Section 12.3, we now propose a number of research directions to be be pursued with the goal of producing better 3-D orthogonal drawings. Firstly, heuristic improvements can be made to many of the algorithms proposed in the literature and those presented in this thesis. For example, in Section 5.5.2 we discuss the use of a vertex-colouring method to determine the heights of edge routes in Algorithm GENERAL POSITION THREE-BEND POINT-DRAWING, thus reducing the volume of the drawings produced. Secondly, a set of refinements could be developed, which given an arbitrary 3-D orthogonal graph drawing, modify the drawing to improve particular aesthetic qualities. Such refinements could form the basis of a post-processing step in any 3-D orthogonal graph drawing algorithm, as has been done for 2-D orthogonal graph drawing by Fößmeier *et al.* [101] and Six *et al.* [197]. An experimental evaluation of the performance of 3-D orthogonal graph drawing algorithms, measuring the relative improvements gained through heuristics and refinements, could be carried out. A first step in this direction, was the experiment of Di Battista *et al.* [74] measuring the performance of a number of 3-D orthogonal point-drawing algorithms.

To produce 3-D graph drawings which are potentially more readable than 3-D orthogonal drawings a more flexible model could be employed. It is expected that for 3-D polyline graph drawings (see Section 1.4.3), considerably fewer bends will be needed to produce drawings with small volume. The tradeoff between angular resolution and the number of bends in such drawings is an interesting area for research. Of theoretical interest is the development of algorithms for drawing graphs in non-orthogonal three-dimensional grids.

Part V

Appendices

Appendix A

Lower Bounds for Three-Dimensional Orthogonal Point-Drawing

In this Appendix we establish lower bounds for the number of bends in 3-D orthogonal point-drawings of simple graphs and multigraphs. Firstly, we show that a 3-D orthogonal point-drawing of K_5 has at least seven bends. This is the only known non-trivial lower bound for the total number of bends in a 3-D orthogonal point-drawing of a simple graph. Theorem 11.1 shows that a 3-D orthogonal point-drawing of K_5 has an edge route with at least two bends. We then provide a formal proof of the well-known result that the multigraph with two vertices and six edges has an edge route with at least three bends in any 3-D orthogonal point-drawing. Finally, we show this multigraph has at least 12 bends in any 3-D orthogonal point-drawing, and we provide such a drawing. Throughout this appendix we implicitly use obvious symmetries to reduce the number of cases to consider.

A.1 Simple Graphs

Our result for K_5 depends on the following results concerning 3-D orthogonal point-drawings of small cycles. Figure A.1 shows 3-D orthogonal point-drawings of the 4-cycle C_4 and of the 5-cycle C_5, each with no bends.

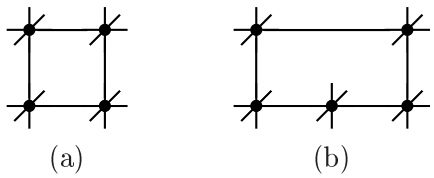


Figure A.1: 0-bend 3-D orthogonal point-drawings of (a) C_4 and (b) C_5.

Lemma A.1. *The only 0-bend 3-D orthogonal point-drawings of C_4 and of C_5 are those shown in Figure A.1.*

Proof. We shall prove this result for C_5. The proof for C_4 is similar. Suppose k is the number of edges in the longest straight-line path in a 0-bend 3-D orthogonal point-drawing of C_5. Obviously $k \leq 4$. If $k = 4$ then, as in Figure A.2(a), there must be a 2-bend edge route. If $k = 3$ then, as in Figure A.2(b), there are two possible place for the final vertex, and in either case there must be a 1-bend edge route.

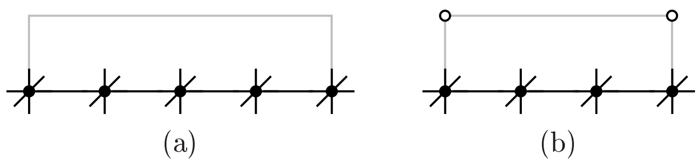


Figure A.2: The cases (a) $k = 4$ and (b) $k = 3$.

If $k = 2$ then, as in Figure A.3, the edges connecting to the ends of the 2-path, may be (a) perpendicular, (b) in opposite directions, or (c) in the same direction. In case (a) there must be a 2-bend edge route. In case (b) there must be a 3-bend edge route, and case (c) produces the 0-bend drawing of C_5 shown in Figure A.1(b).

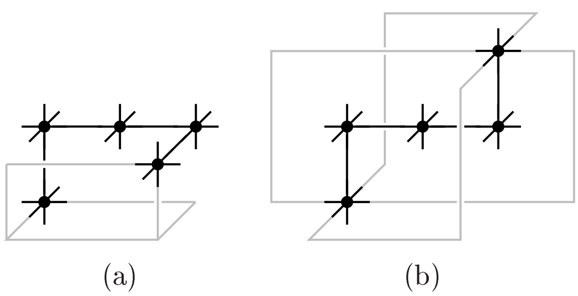
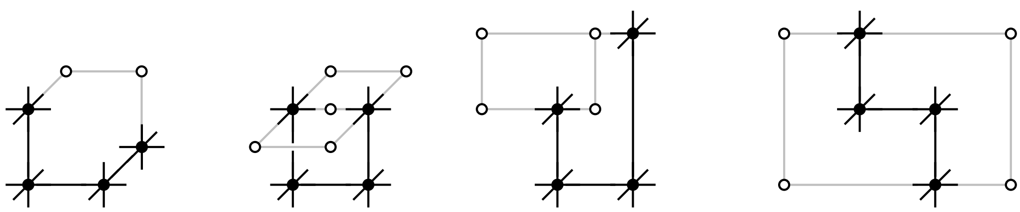


Figure A.3: The case $k = 2$.

If $k = 1$ then, as in Figure A.4, the edges connecting to the ends of the 1-path (which is drawn parallel to the X-axis), may be (a) perpendicular, (b) in the same direction, or (c) in opposite directions. In each case there are no 1-bend edge routes connecting the end-points of the resulting 4-path which do not introduce a straight-line path with two edges. So it is impossible to add the remaining vertex to make a 0-bend 5-cycle with $k = 1$.



(a) (b) (c)

Figure A.4: The case $k = 1$.

Hence the only drawing of the 5-cycle with no bends is that shown in Figure A.1(b) with $k = 2$. □

In Figure 2.3(b) (page 28) there is a 3-D orthogonal point-drawing of K_5 with seven bends. We now show that this is optimal.

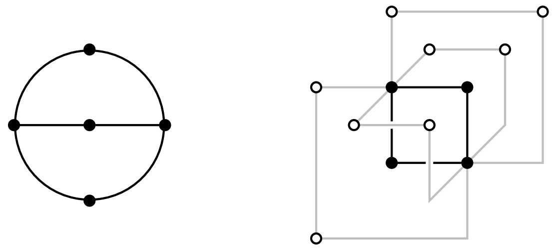
Theorem A.1. *Every 3-D orthogonal point-drawing of K_5 has at least seven bends.*

Proof. Suppose, to the contrary, that there is a 3-D orthogonal point-drawing of K_5 with a total of six bends.

Our proof proceeds by considering the structure of the subgraph of K_5 consisting of the 0-bend edges. It is easily verified that in any subgraph of K_5 with at least seven edges there is a K_3 subgraph. Since K_3 does not have a 0-bend 3-D orthogonal point-drawing, the number of 0-bend edge routes in the drawing of K_5 is at most six.

Clearly, in any K_3-free 6-edge subgraph of K_5 there is a 4-cycle. Given a 4-cycle, the only way to add a fifth vertex and two more edges without creating a triangle is to connect the fifth vertex to the non-adjacent vertices of the 4-cycle. Hence, the only 6-edge K_3-free subgraph of K_5 is that shown in Figure A.5(a), which we call H.

Note that H contains C_4. By Lemma A.1 the only 0-bend 3-D orthogonal point-

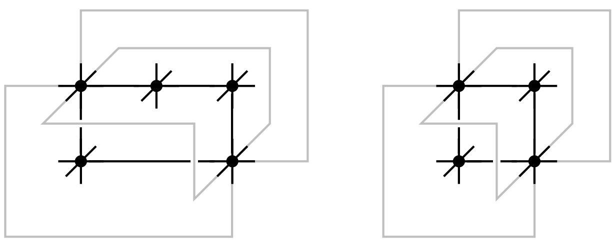


(a) (b)

Figure A.5: (a) K_3-free 6-edge subgraph H of K_5; (b) H does not have a 0-bend 3-D point-drawing

drawing of C_4 is a rectangle. It is not possible to connect the non-adjacent vertices of a rectangle by two 0-bend edges (see Figure A.5(b)). Hence H does not have a 0-bend 3-D orthogonal point-drawing. So the number of 0-bend edge routes in the drawing of K_5 is at most five. By Theorem 11.1, any 3-D point-drawing of K_5 has an edge route with at least two bends. It follows that in a point-drawing of K_5 with six bends there is precisely one 2-bend edge, four 1-bend edges and five 0-bend edges.

A K_3-free subgraph of K_5 with five edges is C_5 or contains C_4. By Lemma A.1, the only 0-bend drawings of C_5 and C_4 are the rectangles shown in Figure A.1. As illustrated in Figure A.6, the diagonally opposite vertices of the rectangles must be connected by a 3-bend edge route, which is a contradiction. The result follows. $\square$



(a) (b)

Figure A.6: 3-bend edge 'across' the 4- and 5-cycle.

A.2 Multigraphs

In Figure A.7 we show 3-D orthogonal point-drawings of the multigraph with two vertices and six edges.

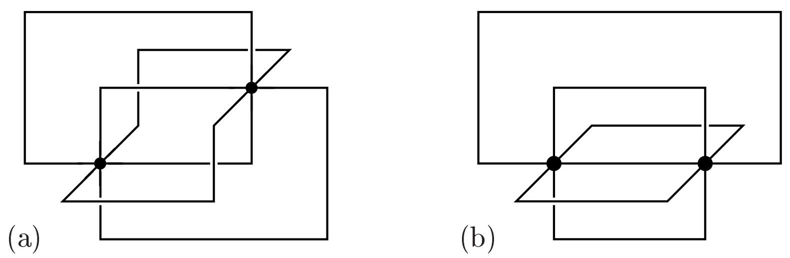


Figure A.7: Drawings of the 2-vertex 6-edge multigraph with (a) a maximum of three bends per edge route, and (b) a total of twelve bends.

We now prove that the maximum number of bends per edge route in the drawing in Figure A.7(a) is optimal.

Lemma A.2. *The multigraph with two vertices and six edges has a 3-bend edge route in every 3-D orthogonal point-drawing.*

Proof. Since the graph is 6-regular every port at the vertices v and w must be used. The two vertices can be (a) collinear, (b) coplanar but not collinear, or (c) not coplanar, as illustrated in Figure A.8.

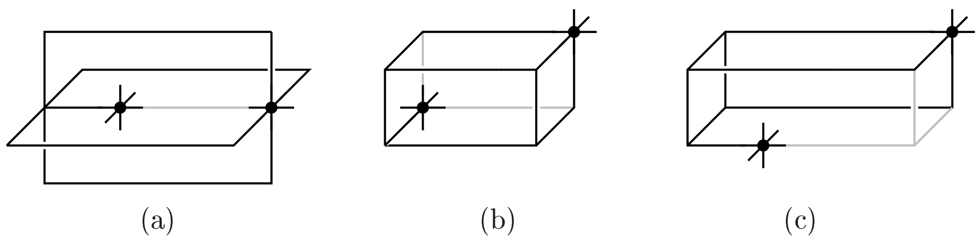


Figure A.8: The 2-vertex 6-edge multigraph needs a 3-bend edge route.

In each case a port at vertex v pointing away from w requires at least three bends to reach w. □

We now prove that the total number of bends in the drawing in Figure A.7(b) is optimal.

Lemma A.3. *The multigraph with two vertices and six edges has at least 12 bends in any 3-D orthogonal point-drawing.*

Proof. If the vertices are not coplanar then at one of the vertices, three of the ports need at least two bends to reach the other vertex, and the other three ports need at least three bends to reach the other vertex. So a non-coplanar drawing has at least 15 bends.

If the vertices are coplanar but not collinear then at one of the vertices, two of the ports need at least one bend to reach the other vertex, two of the ports need at least two bends to reach the other vertex, and the remaining two ports need at least three bends to reach the other vertex. So a non-collinear coplanar drawing has at least 12 bends.

If the vertices are collinear then at one of the vertices, four of the ports need at least two bends to reach the other vertex, and one of the ports needs at least three bends to reach the other vertex. So a non-collinear coplanar drawing has at least 11 bends. Suppose, without loss of generality, that the vertices are in an X-line, and there is such a drawing with 11 bends. Then there must be four 2-bend edge routes, and one 3-bend edge route. These four 2-bend edge routes must use the $Y^{\pm}$ and $Z^{\pm}$ ports at each vertex. Therefore, the edge routed using the X^{-} and X^{+} port must have four bends, which is a contradiction. The result follows. □

Appendix B

3-D Orthogonal 'Cage' Drawings

As discussed in Chapter 3, the 2-bends problem (Problem 3.3) is one of the most interesting open problems in the field of 3-D orthogonal graph drawing. This problem asks whether every maximum degree six graph has a 3-D orthogonal point-drawing with at most two bends per edge route. We now present 3-D orthogonal point-drawings of the 6-regular multi-partite graphs K_7, $K_{2,2,2,2}$, $K_{3,3,3}$ and $K_{6,6}$ with two bends per edge route, thus providing evidence for the conjecture that every maximum degree six graph has a 2-bend 3-D orthogonal point-drawing.

Wood [219] presented the first 2-bend 3-D orthogonal point-drawing of K_7. This drawing is less symmetric than the drawing presented here. In a 2-bend 3-D orthogonal point-drawing the edge routes assigned an 'extreme' port must be planar. The 2-bend point-drawings which follow consist of two parts. The outer 'cage' includes planar and non-planar 2-bend edge routes (see Figure 5.25). The 'interior' consists solely of non-planar 2-bend edge routes.

2-Bend Drawing of K_7:

Figures B.1 and B.2 respectively show a K_6 cage drawing and a $K_{1,6}$ interior drawing which combine to give the $8 \times 8 \times 8$ 2-bend point-drawing of K_7 from Figure 3.6. The vertices are positioned at $(2, 0, 0)$, $(-2, 0, 0)$, $(0, 2, 0)$, $(0, -2, 0)$, $(0, 0, 2)$, $(0, 0, -2)$ and $(1, 1, 1)$.

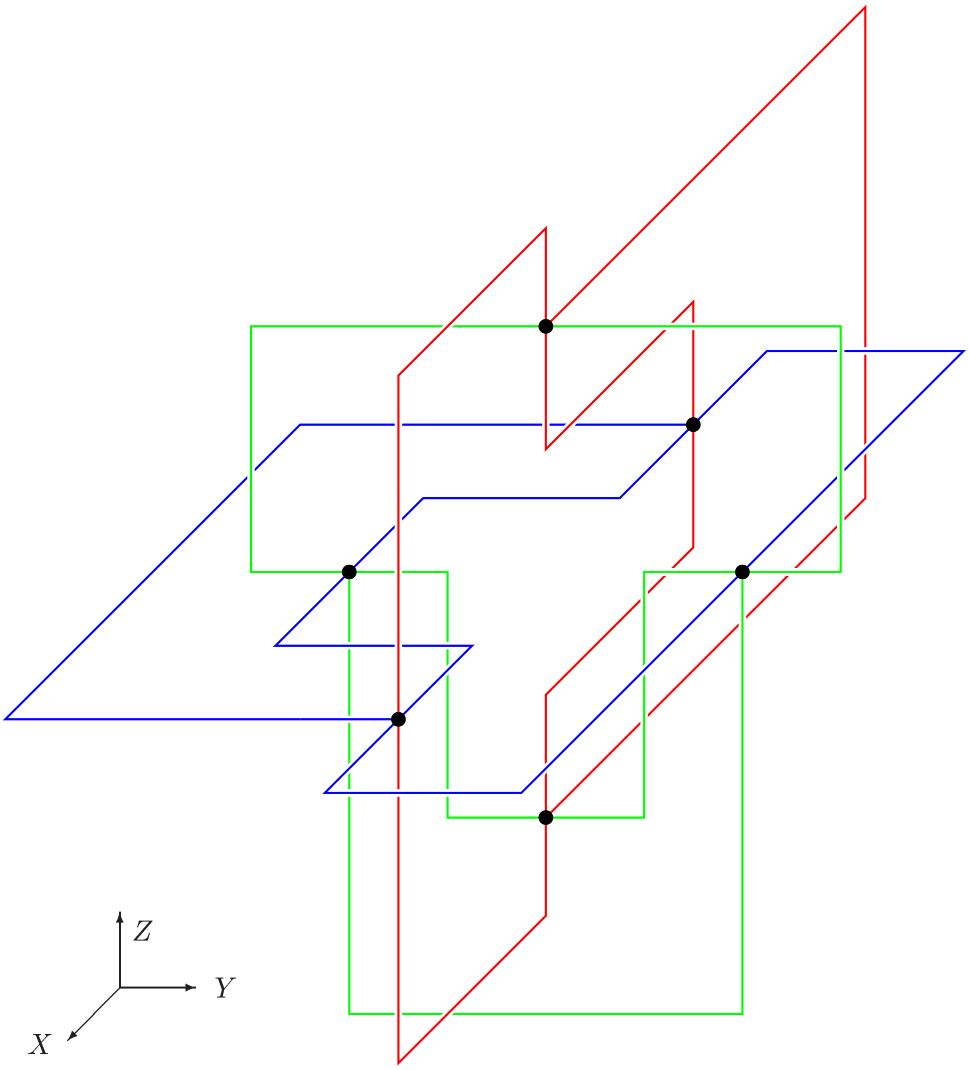


Figure B.1: K_6 cage.

2-Bend Drawings of $K_{2,2,2,2}$ and $K_{3,3,3}$:

Our 2-bend 3-D point-drawings of $K_{2,2,2,2}$ and $K_{3,3,3}$ both use the octahedron graph cage shown in Figure B.3.

Combining the octahedron cage with the $K_{2,6}$ interior drawing shown in Figure B.4 gives a $9 \times 9 \times 9$ 2-bend 3-D orthogonal point-drawing of $K_{2,2,2,2}$.

Combining the octahedron cage with the interior drawing shown in Figure B.5 gives a $10 \times 10 \times 10$ 2-bend 3-D orthogonal point-drawing of $K_{3,3,3}$.

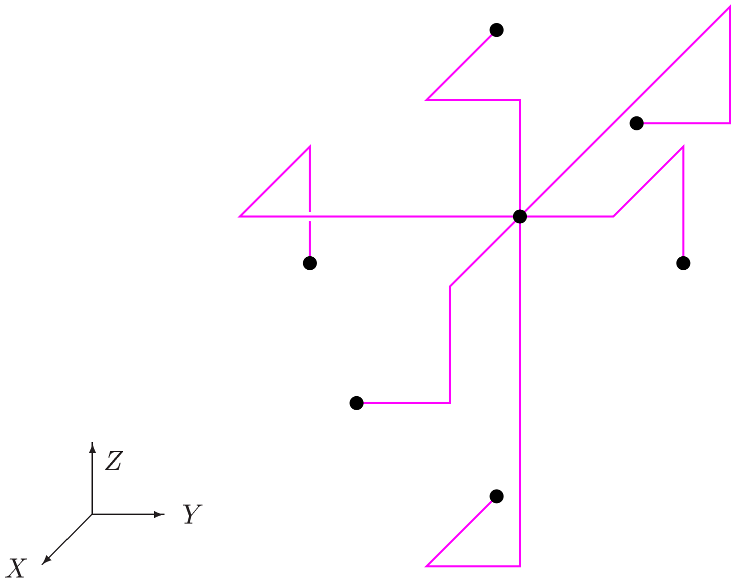


Figure B.2: $K_{1,6}$ drawing forming the interior of K_7.

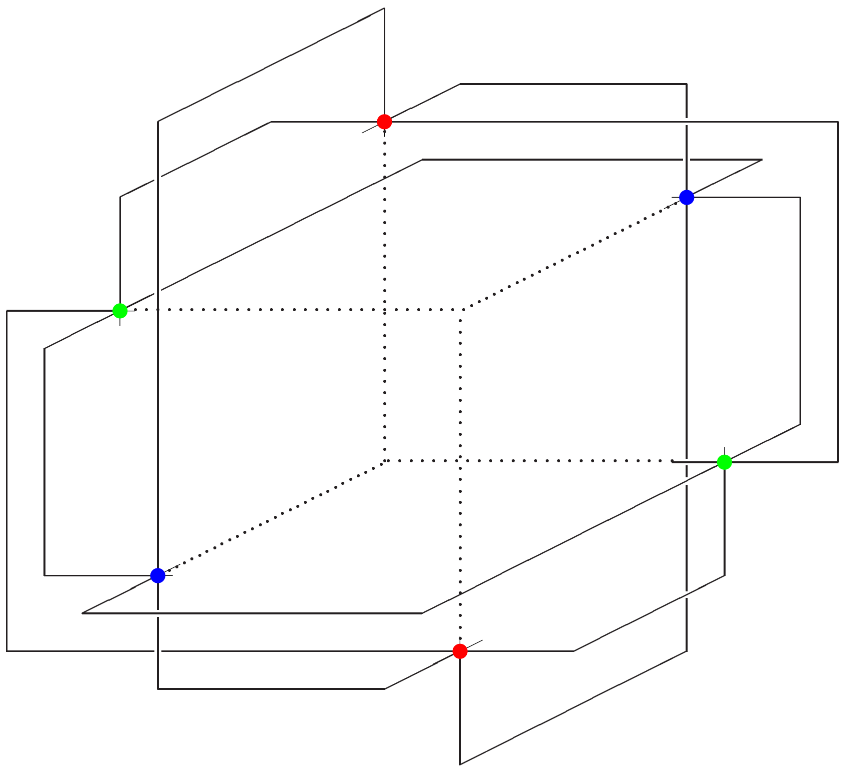


Figure B.3: Octahedron cage

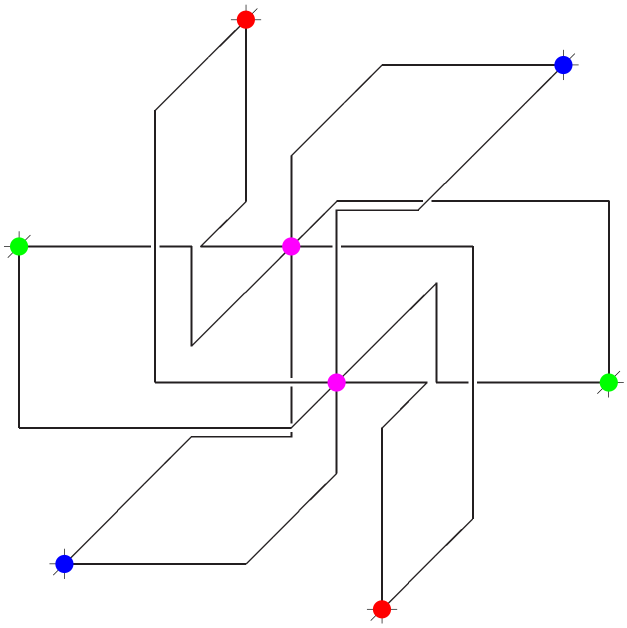


Figure B.4: $K_{2,6}$ drawing forming the interior of $K_{2,2,2,2}$.

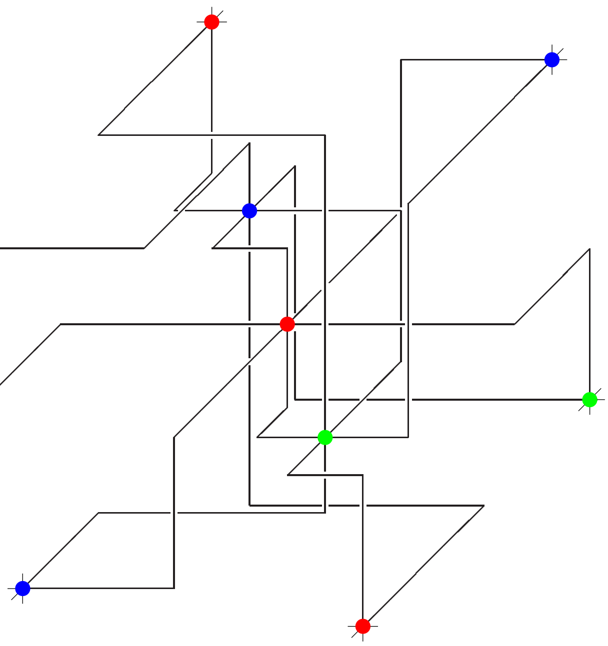


Figure B.5: Interior of $K_{3,3,3}$.

2-Bend Drawing of $K_{6,6}$:

Our 2-bend 3-D orthogonal point-drawing of $K_{6,6}$ consists of the 'bipartite cage' shown in Figure B.6, and the interior drawing of Figure B.7 drawn three times with:
(1) $I = X$, $J = Y$, $K = Z$, (2) $I = Y$, $J = Z$, $K = X$, (3) $I = Z$, $J = X$, $K = Y$.

We position the vertices of $K_{6,6}$ as indicated in Table B.1, obtaining a $12 \times 12 \times 12$ 2-bend 3-D orthogonal point-drawing of $K_{6,6}$. This drawing was found using the search technique presented in Section 5.2.2, which is based on the algorithm in Appendix C for the maximum clique problem.

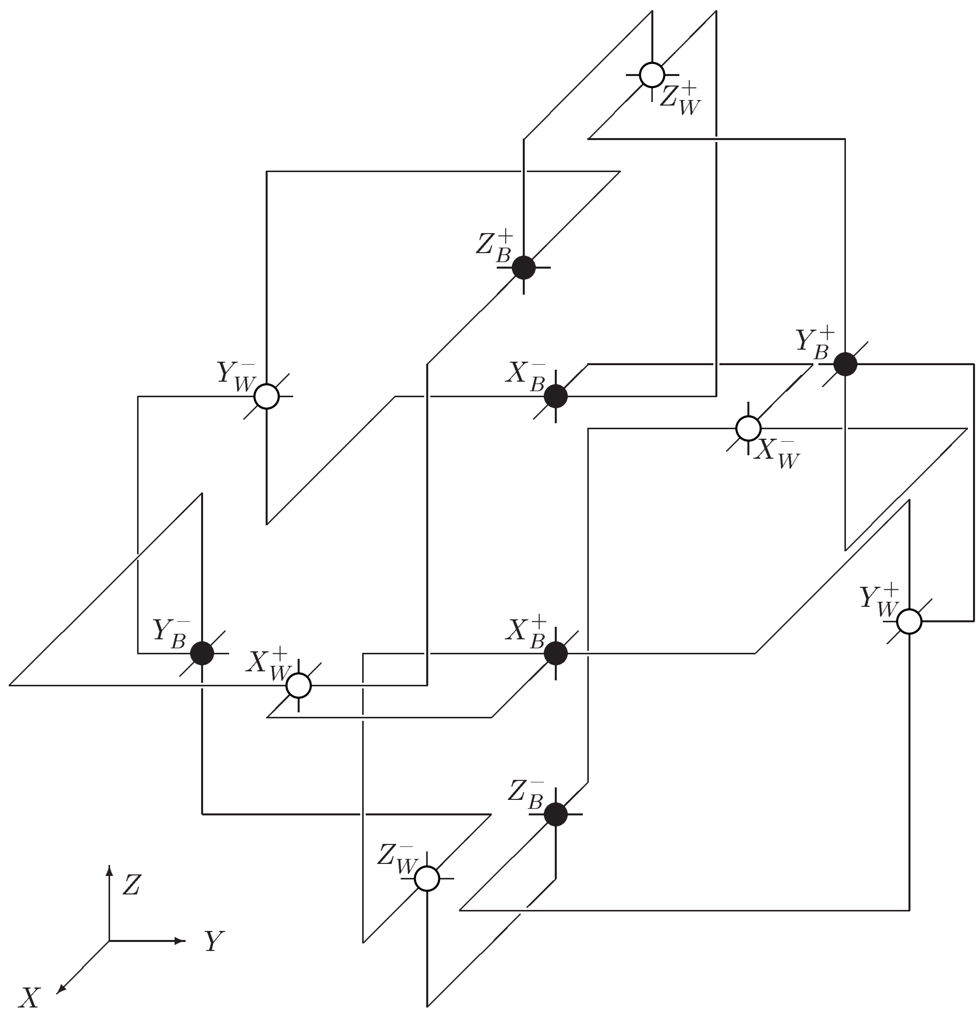


Figure B.6: Bipartite cage.

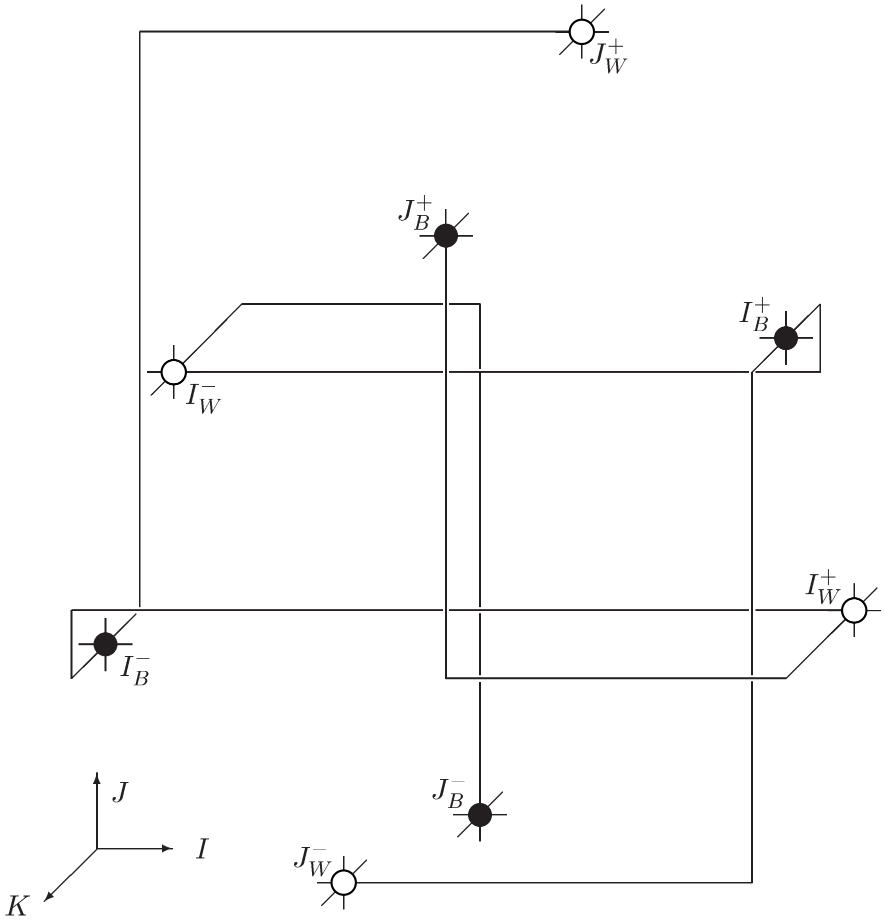


Figure B.7: Interior of $K_{6,6}$.

Table B.1: Coordinates of $V(K_{6,6})$.

X_W^+	(4,-2,0)	Y_W^+	(0,4,-2)	Z_W^+	(-2,0,4)
X_B^+	(3,2,0)	Y_B^+	(0,3,2)	Z_B^+	(2,0,3)
X_W^-	(-4,1,-1)	Y_W^-	(-1,-4,1)	Z_W^-	(1,-1,-4)
X_B^-	(-5,-3,-1)	Y_B^-	(-1,-5,-3)	Z_B^-	(-3,-1,-5)

Appendix C

Maximum Clique Algorithm

In this appendix we describe an algorithm for finding a maximum clique in a graph and compare its performance with leading algorithms for this problem in an experimental study. In Section 5.2.2 we describe how this algorithm can be used for searching for 2-bend 3-D orthogonal point-drawings. For example, it was used to find the 2-bend drawing of $K_{6,6}$ presented in Appendix B. This algorithm and the experimental results were published in [218].

C.1 Introduction

As defined in Section 2.2, a *clique* of an undirected graph G is a set of pairwise adjacent vertices. A set of pairwise non-adjacent vertices is called an *independent set*. In this appendix we address the *Maximum Clique Problem*; i.e., for a given undirected graph G find a maximum cardinality clique of G (whose cardinality we denote by $\omega(G)$).

Clearly the maximum clique problem is equivalent to that of finding a maximum independent set in the complementary graph. Applications for this problem exist in signal processing, computer vision and experimental design for example (see Balas and Yu [13]). Unfortunately, not only is the exact problem NP-hard (see Garey and Johnson [105]), but Arora *et al.* [7] show that approximating the maximum clique problem within a factor of $|V|^{\epsilon}$ for some $\epsilon > 0$ is NP-hard .

Early algorithms included the branch and bound algorithm of Bron and Kerbosch

[46] to generate all the cliques of a graph and the recursive algorithm of Tarjan and Trojanowski [206] to determine a maximum independent set of an n-vertex graph in $O(2^{n/3})$ time. Recent approaches to the maximum clique problem have included the branch and bound algorithms of Carraghan and Pardalos [52], Pardalos and Rodgers [170], Balas and Yu [13], Balas and Xue [11, 12], Babel and Tinhofer [9], and Babel [8]. In their survey paper, Pardalos and Xue [171] identify the following key issues in a branch and bound algorithm for the maximum clique problem.

1. How to find a good lower bound, i.e., a clique of large size?
2. How to find a good upper bound on the size of a maximum clique?
3. How to branch, i.e., break a problem into smaller subproblems?

In Section C.2 we address the first two of these questions. In Section C.3 we present our branch and bound algorithm, and in Section C.4 we discuss computational results of our algorithm in comparison with leading algorithms for the maximum clique problem.

C.2 Heuristics

The algorithm of Balas and Yu [13] concentrates on the determination of lower bounds using an algorithm to find a maximum clique of a maximal triangulated induced subgraph at selected search tree nodes. This method is extended to the maximum weight clique problem by Balas and Xue [11]. The algorithm to follow and the algorithm of Balas and Xue [12] determine a lower bound at the root node of the search tree, using the algorithm of Balas [10] to find a maximum clique of an edge-maximal triangulated subgraph. To provide lower bounds at non-root search tree nodes we use the following well-known heuristic which we call Greedy Clique. Given a graph G, maintain a set S (initially $S \leftarrow V(G)$) of candidate vertices to be added to the current clique. Add a vertex $v \in S$ to the current clique, set $S \leftarrow (S \setminus \{v\}) \cap V_G(v)$, and continue until $S = \emptyset$.

We now turn our attention to the determination of upper bounds. The algorithms of Carraghan and Pardalos [52] and Pardalos and Rodgers [170] use the size of a given subgraph as an upper bound for the size of a clique in that subgraph. Vertex-colourings provide much tighter upper bounds. A vertex *k-colouring* of a graph G partitions $V(G)$

into k independent sets $(C_1, C_2, \ldots, C_k)$ called *colour classes*. Each vertex of a clique must be coloured differently, so k is an upper bound for $\omega(G)$. As discussed in Section 2.2, the algorithm Greedy Vertex-Colour is a simple heuristic for determining a vertex-colouring of a graph.

In [8, 9, 12] upper bounds for the maximum clique problem are determined using the Dsatur vertex-colouring heuristic of Brelaz [43]. Brelaz defines the *saturation degree* of an uncoloured vertex v to be the number of colours assigned to the vertices adjacent to v. While uncoloured vertices remain, the Dsatur heuristic chooses an uncoloured vertex v with maximum saturation degree (breaking ties by higher degree), and colouring v with the minimum colour not already assigned to an adjacent vertex.

This method colours the connected components of G in turn, and within each connected component the initial vertices chosen form a clique. So Dsatur provides both a lower and upper bound for $\omega(G)$. Comparisons of Greedy Vertex Colour and Dsatur in [12, 217] show that for all but a few of the tested graphs Dsatur requires (up to 27.5%) fewer colours than Greedy Vertex Colour, although Dsatur is considerably slower. For very sparse and very dense graphs, Dsatur is an order of magnitude more expensive than colour [12].

A *fractional colouring* of a graph G is a set C of (possibly intersecting) weighted colour classes (i.e., independent sets), such that for each vertex $v \in V(G)$ the sum of the weights of the colour classes containing v is at least one. Since a colour class can contain at most one vertex of a clique, in a fractional colouring the sum of the weights of those colour classes intersecting a clique Q is at least $|Q|$. Therefore the total weight of a fractional colouring of a graph G is an upper bound for $\omega(G)$. The upper bound from a minimum weight fractional colouring is in general tighter than that provided by a minimum vertex-colouring [12]; unfortunately determining such a fractional colouring is NP-hard [112] .

Balas and Xue [12] use the following heuristic FCP for the fractional colouring problem to provide upper bounds for the maximum clique problem. After i iterations of FCP, each vertex is coloured exactly i times, and each colour class is assigned weight $1/i$, so $t_i = |C|/i$ is an upper bound for $\omega(G)$. Initially $C \leftarrow \emptyset$, $i \leftarrow 1$ and $t_0 \leftarrow \infty$. Iteration i of FCP executes the following algorithm.

For each vertex v, include v in the first colour class $C_j \in C$, if one exists, such that $C_j \cup \{v\}$ remains an independent set. Suppose U is the set of vertices not included in a colour class. Find a vertex-colouring $(C_1, C_2, \ldots, C_k)$ of $G[U]$ (using GREEDY VERTEX COLOUR or DSATUR), and set $C \leftarrow C \cup \{C_1, C_2, \ldots, C_k\}$ and $t_i \leftarrow |C|/i$. If $t_i < t_{i-1}$ then set $i \leftarrow i + 1$ and repeat, otherwise return the upper bound $\lfloor t_{i-1} \rfloor$.

To prove a time complexity result for FCP, the authors amend the stopping rule so that the number of colour classes $|C|$ does not exceed the number of vertices $|V|$. Our implementation also includes this feature. Note that for many graphs a tighter upper bound can be calculated by reiterating the algorithm after either stopping condition is satisfied.

By FCP_G and FCP_D we refer to algorithm FCP with GREEDY VERTEX COLOUR and DSATUR determining vertex-colourings, respectively. The comparison of these heuristics in [12, 217] show that the improvements in upper bound by FCP_G over GREEDY VERTEX COLOUR range from 0–21 colours, and for FCP_D over DSATUR the improvements range from 0–7 colours.

C.3 Maximum Clique Algorithm

We now present our branch and bound algorithm MC for the maximum clique problem, which uses the FCP heuristic to determine upper bounds, and, like the algorithms in [52, 170], activates exactly one new search tree node at each branching stage. Other branch and bound algorithms for the maximum clique problem activate many search tree nodes at each branching step. This is inefficient as new bounds need to be determined for each subgraph considered. A lower bound (i.e., a large maximum clique) is only determined at the root node of the search tree. To do so we use the linear-time algorithm of Balas and Yu [13] (also see Xue [225]) for finding a maximum clique in an edge-maximal triangulated subgraph of the input graph.

Given a graph G, algorithm MC maintains the following conditions:

- If h is the current depth of the search tree then the set of vertices $\{v_1, v_2, \ldots, v_{h-1}\} \subseteq V(G)$ is a clique of G.

- M is the current largest clique found by the algorithm; $h - 1 \leq |M| \leq \omega(G)$.

- For $1 \le i \le h$, the vertex set $S_i \subseteq \bigcap_{j=1}^{i-1} V_G(v_j)$ consists of candidates for enlarging $\{v_1, v_2, \ldots, v_{i-1}\}$.
- For each i, $1 \le i \le h$, $(C_1^i, C_2^i, \ldots, C_{k_i}^i)$ is a vertex-colouring of $G[S_i]$. Both k_i and k_i' (determined by FCP) are upper bounds for $\omega(G[S_i])$, with $k_i' \le k_i$.
- An active node of the search tree corresponds to the subproblem of finding a maximum clique larger than M of the subgraph:

  $$G_i = G[\{v_1, v_2, \ldots, v_{i-1}\} \cup S_i], \text{ for } 1 \le i \le h.$$

  Clearly $\omega(G_i) \le i - 1 + k_i' \le i - 1 + k_i$.

Algorithm C.1. MaxClique

Input: graph G

Output: maximum clique of G

Step 0: *Initialisation*

Find a maximum clique M of an edge-maximal triangulated subgraph of G [13, 225].

Set $h \leftarrow 1$, $S_h \leftarrow V(G)$ and go to Step 2.

Step 1: *Calculate Lower Bound*

$Q \leftarrow$ Greedy Clique($G[S_h]$).

if $h - 1 + |Q| > |M|$ **then** set $M \leftarrow \{v_1, v_2, \ldots, v_{h-1}\} \cup Q$.

Go to Step 2.

Step 2: *Calculate Upper Bound*

Find a vertex-colouring $(C_1^h, C_2^h, \ldots, C_{k_h}^h)$ of $G[S_h]$.

if $h - 1 + k_h \le |M|$ **then** go to Step 4.

Apply FCP to $G[S_h]$ to obtain a further upper bound $k_h' \ge \omega(G[S_h])$.

if $h - 1 + k_h' \le |M|$ **then** go to Step 4.

Go to Step 3.

Step 3: *Branching*

Choose a vertex $v_h \in C_{k_h}^h$ with maximum $\deg_G(v_h)$.

Set $S_{h+1} \leftarrow S_h \cap V_G(v_h)$, $S_h \leftarrow S_h \setminus \{v_h\}$, $C^h_{k_h} \leftarrow C^h_{k_h} \setminus \{v_h\}$.

if $C^h_{k_h} = \emptyset$ **then** set $k_h \leftarrow k_h - 1$ and **if** $k_h < k'_h$ **then** set $k'_h \leftarrow k_h$.

Set $h \leftarrow h + 1$.

Go to Step 1.

Step 4: *Backtracking*

if $h = 1$ **then stop**: M is a maximum clique of G.

Set $h \leftarrow h - 1$.

if $h - 1 + k'_h \leq |M|$ **then** go to Step 4.

Go to Step 3.

In the second line of Step 3, the problem of finding a maximum clique of G_h is divided into two sub-problems. If v_h is a vertex of $G[S_h]$ then a clique Q of G_h will be contained in either:

$$\begin{aligned} & G_{h+1} = G[\{v_1, v_2, \ldots, v_h\} \cup (S_h \cap V_G(v_h))] \qquad (\text{if } v_h \in Q) \\ \text{or } & G_h = G[\{v_1, v_2, \ldots, v_{h-1}\} \cup (S_h \setminus \{v_h\})] \qquad (\text{if } v_h \notin Q). \end{aligned}$$

We choose v_h from the final colour class $C^h_{k_h}$, as the latter colour classes generated by Greedy Vertex Colour and by Dsatur tend to be smaller than the initial ones. Therefore the upper bound k_h is reduced more quickly than if an arbitrary vertex in S_h was chosen. Note that, since $|M| \geq h-1$ and $h-1+k_h > |M|$ whenever the algorithm goes to Step 3, we have $k_h \geq 1$ at this stage, and hence the colour class $C^h_{k_h}$ must exist.

Theorem C.1. *Given an undirected graph G, algorithm MC finds a maximum clique M of G.*

Proof. This result follows immediately from the observation that algorithm MC maintains the abovementioned conditions throughout the algorithm. □

C.4 Experimental Results

See [217] for a complete description of the implementation of our algorithms in GAP [193] on a Sun Sparcstation 10.

To evaluate the effectiveness of the FCP heuristic as an upper bounding device for the maximum clique problem, we have also developed an algorithm MC′ which skips the third and fourth lines of Step 2, thus not using FCP to calculate a further upper bound. $\mathrm{MC_G}$ (respectively, $\mathrm{MC'_G}$) uses GREEDY CLIQUE to determine a clique in Step 1, and $\mathrm{FCP_G}$ (GREEDY VERTEX COLOUR) to determine upper bounds in Step 2. $\mathrm{MC_D}$ (respectively, $\mathrm{MC'_D}$) uses $\mathrm{FCP_D}$ (DSATUR) for these purposes.

We now compare the performance of algorithms $\mathrm{MC_G}$, $\mathrm{MC_D}$, $\mathrm{MC'_G}$ and $\mathrm{MC'_D}$ with existing algorithms for the maximum clique problem. By BXB we refer to a combination of the algorithms of Babel [8] and Balas and Xue [12], the most efficient known algorithms for the maximum clique problem. BXB uses $\mathrm{FCP_D}$ to calculate lower and upper bounds at each search tree node, and uses branching rule II in [8], their best performing branching rule. The branching rules in [8] and [12] (which is stated for weighted graphs) both generally activate more than one new search tree node.

Table C.1 shows the average size of the lower bound determined at the root node (LB), the average size of a maximum clique ($|M|$), the average CPU time taken by each of the algorithms, and the average number of search tree nodes generated by each algorithm, for 10 uniform random graphs with $n = |V(G)|$ vertices and % edge density $d = 200|E|/n(n-1)$.

In Table C.2 we compare the algorithms for a selection of the DIMACS benchmark graphs which were developed as part of the 1993 DIMACS Challenge (see Johnson and Trick [122]). They include non-uniform random graphs with relatively large clique sizes, and graphs which have arisen in coding theory, the Steiner Triple Problem, tiling of hypercubes, vertex cover problems and fault diagnosis. Table B.2 shows the size n and % density d of the graph, the CPU time taken by each algorithm, and the number of search tree nodes generated by each algorithm. Column BX refers to the number of search tree nodes for the algorithm of Balas and Xue [12], as stated in their paper. To accurately compare algorithms we use the values presented in [12] for the lower bound at the root node for each of the tested algorithms.

In most cases the algorithms $\mathrm{MC_D}$, BXB and BX, which use the upper bound heuristic $\mathrm{FCP_D}$, generate the least number of search tree nodes. $\mathrm{MC_D}$ on average generates less search tree nodes than BXB for 12 of the 16 sets of random graphs. For

Table C.1: Performance of Maximum Clique Finding Algorithms on Uniform Random Graphs

n	d	LB	$\lvert M\rvert$	CPU Time (seconds)					Search Tree Nodes				
				MC_G	MC_D	MC'_G	MC'_D	BXB	MC_G	MC_D	MC'_G	MC'_D	BXB
100	10	3.7	3.9	0.222	0.428	0.160	0.280	0.432	24.3	18.7	24.8	18.7	23.1
100	20	4.7	5.1	0.363	0.755	0.277	0.523	0.752	38.1	33.9	40.6	34.7	39.2
100	30	5.6	6.3	0.959	1.590	0.422	0.883	1.390	53.4	45.6	79.2	52.9	50.3
100	40	6.9	7.6	1.325	3.148	0.613	1.508	3.020	109.8	82.4	165.6	102.8	89.1
100	50	8.1	9.1	2.515	6.894	1.478	3.780	6.458	254.1	198.7	344.5	234.9	201.9
100	60	10.4	11.6	5.497	14.18	1.932	6.860	14.87	468.4	328.7	707.5	405.8	365.4
100	70	12.8	14.8	14.31	36.85	3.445	18.08	38.38	1,048	672.7	1,705	893.4	698.1
100	80	18.0	20.0	35.43	92.84	6.525	46.46	88.62	1,786	1,253	2,961	1,696	1,160
100	90	28.0	30.7	73.84	150.1	12.12	71.30	134.1	2,126	1,109	4,043	1,523	974.3
200	10	4.0	4.3	1.013	2.498	0.962	1.715	2.705	92.3	83.5	98.2	83.7	91.2
200	20	5.1	5.9	2.708	5.810	1.548	4.217	5.965	140.3	120.7	202.2	137.6	126.9
200	30	6.1	7.3	7.030	17.71	3.187	9.095	18.56	519.9	396.2	699.5	476.8	386.0
200	40	7.6	9.0	16.04	47.64	5.510	26.04	49.85	1,539	1,162	2,011	1,279	1,317
200	50	10.0	11.1	57.49	161.5	12.68	81.31	168.1	4,295	2,810	6,846	3,622	2,889
200	60	12.1	14.0	249.9	755.6	45.66	380.4	820.4	17,461	11,704	26,857	14,712	13,109
200	70	15.3	18.1	1,993	5,830	341.9	2,945	5,829	102,122	64,430	173,810	88,354	63,972

12 of the DIMACS benchmark graphs, the lower bound and upper bound calculated at the root node by these algorithms are equal, and therefore only one search tree node is generated. Of the other 26 DIMACS benchmark graphs, MC_D uses the least search tree nodes of these algorithms 15 times, BXB 10 times, and BX 8 times.

Table C.2: Performance of Maximum Clique Finding Algorithms on the DIMACS Benchmark Graphs

DIMACS Graph	n	d	$\|M\|$	CPU Time (seconds)					Search Tree Nodes					
				MC_G	MC_D	MC'_G	MC'_D	BXB	MC_G	MC_D	MC'_G	MC'_D	BXB	BX
brock200_1	200	75	21	4,911	15,186	805.2	7,951	16,320	218,853	149,153	379,810	211,013	163,348	113,244
brock200_2	200	50	12	26.72	149.7	3.833	74.22	158.4	1,790	3,018	2,594	3,593	3,018	2,965
brock200_3	200	61	15	230.1	573.6	38.50	281.0	815.9	15,354	7,818	24,113	10,113	12,717	8,155
brock200_4	200	66	17	568.2	1,926	92.95	931.5	1,530	31,751	25,105	52,332	33,693	19,316	25,705
c-fat200-1	200	8	12	0.283	2.200	0.017	0.150	2.133	8	1	8	4	1	1
c-fat200-2	200	16	24	0.317	0.183	0.017	0.183	0.167	7	1	7	1	1	1
c-fat200-5	200	43	58	0.683	3.467	0.133	2.217	3.284	27	27	27	27	27	29
c-fat500-1	500	4	14	0.534	0.616	0.017	0.617	2.217	13	1	13	1	1	1
c-fat500-2	500	7	26	1.417	0.700	0.083	0.700	0.750	23	1	23	1	1	1
c-fat500-5	500	19	64	1.450	0.984	0.166	0.950	0.983	23	1	23	1	1	1
c-fat500-10	500	37	126	0.017	1.400	0.033	1.400	1.450	1	1	1	1	1	1
hamming6-2	64	90	32	0.017	0.050	0.001	0.067	0.066	1	1	1	1	1	1
hamming6-4	64	35	4	0.133	0.850	0.067	0.300	0.800	81	29	81	58	29	48
hamming8-2	256	97	128	0.017	0.733	0.001	0.750	0.717	1	1	1	1	1	1

continued on next page

Table C.2: *continued*

DIMACS Graph	n	d	$\lvert M\rvert$	CPU Time (seconds)					Search Tree Nodes					
				MC_G	MC_D	MC'_G	MC'_D	BXB	MC_G	MC_D	MC'_G	MC'_D	BXB	BX
hamming8-4	256	64	16	344.2	155.7	79.15	137.6	156.5	28,593	357	36,441	2,045	357	373
hamming10-2	1,024	99	512	0.050	10.57	0.066	10.47	12.28	1	1	1	1	1	1
johnson8-2-4	28	56	4	0.050	0.050	0.017	0.083	0.033	20	1	23	26	1	1
johnson8-4-4	70	77	14	0.533	0.300	0.183	0.534	0.300	115	1	115	19	1	1
johnson16-2-4	120	76	8	770.8	0.417	195.8	2,046	0.384	190,084	1	256,099	355,522	1	1
keller4	171	65	11	113.1	256.5	18.45	137.5	256.7	6,543	3,700	12,829	5,195	3,700	4,164
MANN_a9	45	93	16	0.617	1.033	0.100	0.384	1.017	46	19	60	20	19	23
MANN_a27	378	99	126	23,286	26,524	704.3	9,753	25,549	39,087	8,704	47,264	9,874	8,714	14,145
p_hat300-1	300	24	8	8.800	38.93	1.467	20.12	37.53	1,032	819	1,310	928	819	832
p_hat300-2	300	49	25	75.05	225.6	10.05	129.2	225.5	1,888	1,304	2,801	1,579	1,304	1,613
p_hat500-1	500	25	9	76.48	384.8	13.72	231.4	389.5	7,454	6,179	9,772	6,724	6,179	6,105
p_hat500-2	500	50	36	2,695	9,790	267.1	5,796	6,320	35,657	27,182	59,393	34,787	17,019	31,746
p_hat700-1	700	25	11	272.8	1,915	40.32	1,060	1,408	17,629	19,337	25,805	23,150	15,310	14,040
p_hat1000-1	1,000	24	10	1,883	13,060	283.2	6,974	13,150	122,182	90,607	179,082	111,897	91,159	93,004
san200_0.7_1	200	70	30	6.617	36.37	0.917	18.85	95.73	53	231	206	348	645	635

continued on next page

Table C.2: *continued*

DIMACS Graph	n	d	$\|M\|$	CPU Time (seconds)					Search Tree Nodes					
				MC_G	MC_D	MC'_G	MC'_D	BXB	MC_G	MC_D	MC'_G	MC'_D	BXB	BX
san200_0.7_2	200	70	18	3.700	20.80	0.466	10.65	36.53	110	154	195	182	363	852
san200_0.9_1	200	90	70	73.75	45.72	11.48	24.92	255.4	715	121	2,069	201	631	10
san200_0.9_2	200	90	60	5,988	612.6	1,052	348.0	2,036	71,114	1,553	211,889	2,365	5,655	1,825
san400_0.5_1	400	50	13	51.03	81.73	11.22	64.83	247.7	1,223	378	3,465	523	1,689	1,194
san400_0.7_1	400	70	40	1,681	2,455	198.7	1,430	10,263	15,903	5,604	38,989	8,507	30,707	20,913
san400_0.7_2	400	70	30	36,486	39,100	6,228	24,285	66,579	690,806	139,092	1,591,030	231,593	295,314	75,773
san1000	1,000	50	15	2,281	32,630	653.9	40,814	9,277	43,623	44,408	106,823	78,698	12,996	21,897
sanr200_0.7	200	70	18	1,711	4,608	338.2	2,372	4,076	87,012	51,610	150,861	71,799	44,278	40,496
sanr400_0.5	400	50	13	2,352	9,094	350.9	4,955	8,617	155,285	115,210	233,381	136,636	114,208	112,932

Those algorithms which use the vertex-colouring heuristic Greedy Vertex Colour, while generating the most search tree nodes, are generally the fastest. In particular, for the random graphs, $\mathrm{MC}'_{\mathrm{G}}$ is the fastest of the tested algorithms, using on average only 18.41% of the CPU time used by BXB. $\mathrm{MC}'_{\mathrm{G}}$ is again the fastest for all but four of the DIMACS benchmark graphs (and for two of these the difference is only a few microseconds). We have also implemented a variant $\mathrm{MC2}_{\mathrm{G}}$ of $\mathrm{MC}'_{\mathrm{G}}$ which only finds a lower bound at the root node of the search tree. For the random graphs (DIMACS benchmark graphs) this algorithm uses 0.65% (0.20%) more search tree nodes than $\mathrm{MC}'_{\mathrm{G}}$, yet is on average 4.34% (12.04%) faster than $\mathrm{MC}'_{\mathrm{G}}$. This indicates that the determination of lower bounds at non-root nodes is not time-efficient.

We have observed that for graphs with fixed size and density the difficulty of the maximum clique problem is generally inversely correlated to the size of a maximum clique in the graph. This is apparent for the *san* graphs with equal n and d. Similar results occur with the random graphs. For example, the 10 uniform random graphs (used in Table C.1) with $n = 100$ and $d = 90\%$ have a maximum clique of size 29(2), 30(3), 31(2), 32(2) or 33(1). For each maximum clique size, Table C.3 shows the average CPU time taken, and the average number of search tree nodes generated by each algorithm.

Table C.3: Performance of Maximum Clique Finding Algorithms on Uniform Random Graphs with $n = 100$ and $d = 90\%$

$\|M\|$	CPU Time (seconds)					Search Tree Nodes				
	MC_{G}	MC_{D}	$\mathrm{MC}'_{\mathrm{G}}$	$\mathrm{MC}'_{\mathrm{D}}$	BXB	MC_{G}	MC_{D}	$\mathrm{MC}'_{\mathrm{G}}$	$\mathrm{MC}'_{\mathrm{D}}$	BXB
29	160.0	263.1	26.34	122.5	285.8	4,957	2,014	9,854	2,721	2,216
30	66.09	158.2	10.38	74.80	134.4	1,885	1,183	3,259	1,620	966
31	53.27	94.34	9.740	45.92	79.52	1,392	643.5	2,815	938.5	522
32	50.80	138.0	7.809	66.97	92.19	1,323	1,005	2,465	1,372	623
33	12.03	36.32	2.300	17.90	22.80	256	217	391	307	123

Bibliography

[1] M. A. Aboelaze and B. W. Wah, Complexities of layouts in three-dimensional VLSI circuits. *Inform. Sci.*, **55(1-3)**:167–188, 1991.

[2] A. Aggarwal, M. Klawe, and P. Shor, Multilayer grid embeddings for VLSI. *Algorithmica*, **6(1)**:129–151, 1991.

[3] N. Alon and M. Tarsi, Colourings and orientations of graphs. *Combinatorica*, **12**:125–134, 1992.

[4] K. Appel and W. Haken, Every planar map is four colorable. I. Discharging. *Illinois J. Math.*, **21(3)**:429–490, 1977.

[5] K. Appel, W. Haken, and J. Koch, Every planar map is four colorable. II. Reducibility. *Illinois J. Math.*, **21(3)**:491–567, 1977.

[6] D. Archdeacon, A Kuratowski theorem for the projective plane. *J. Graph Theory*, **5**:243–246, 1981.

[7] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy, Proof verification and intractability of approximation problems. In *Proc. 33rd IEEE Symp. on Foundations of Computer Science (FOCS'92)*, pp. 13–22, 1992.

[8] L. Babel, Finding maximum cliques in arbitrary and in special graphs. *Computing*, **46**:321–341, 1991.

[9] L. Babel and G. Tinhofer, A branch and bound algorithm for the maximum clique problem. *Methods and Models of Operations Research*, **34**:207–217, 1990.

[10] E. Balas, A fast algorithm for finding an edge-maximal subgraph with a TR-formative coloring. *Discrete Applied Mathematics*, **15**:123–134, 1986.

[11] E. Balas and J. Xue, Minimum weighted coloring of triangulated graphs, with application to maximum weight vertex packing and clique finding in arbitrary graphs. *SIAM J. Comput.*, **20(2)**:209–221, 1991.

[12] E. Balas and J. Xue, Weighted and unweighted maximum clique algorithms with upper bounds from fractional coloring. *Algorithmica*, **15**:397–412, 1996.

[13] E. Balas and C. S. Yu, Finding a maximum clique in an arbitrary graph. *SIAM J. Comput.*, **15(4)**:1054–1068, 1986.

[14] C. Batini, E. Nardelli, and R. Tamassia, A layout algorithm for data-flow diagrams. *IEEE Trans. on Software Engineering*, **SE-12(4)**:538–546, 1986.

[15] C. Batini, M. Talamo, and R. Tamassia, Computer aided layout of entity-relationship diagrams. *J. Systems and Software*, **4**:163–173, 1984.

[16] L. W. Beineke, The decomposition of complete graphs into planar subgraphs. In F. Harary, ed., *Graph Theory and Theoretical Physics*, pp. 139–154, Academic Press, London, 1967.

[17] L. W. Beineke, F. Harary, and J. W. Moon, On the thickness of the complete bipartite graph. *Proc. Cambridge Philos. Soc.*, **60(1)**:1–5, 1964.

[18] L. W. Beineke and R. J. Wilson, eds., *Graph Connections.* Oxford University Press, 1997.

[19] B. Berger and P. W. Shor, Tight bounds for the maximum acyclic subgraph problem. *J. Algorithms*, **25**:1–18, 1997.

[20] P. Bertolazzi, G. Di Battista, and W. Didimo, Computing orthogonal drawings with the minimum number of bends. In [68], pp. 331–344.

[21] S. N. Bhatt and S. S. Cosmadakis, The complexity of minimizing wire lengths in VLSI layouts. *Inform. Proc. Lett.*, **25**:263–267, 1987.

[22] S. N. Bhatt and F. T. Leighton, A framework for solving VLSI graph layout problems. *J. Comput. System Sci.*, **28(2)**:300–343, 1984.

[23] T. C. Biedl, Improved orthogonal drawings of 3-graphs. In *Canadian Conference on Computational Geometry (CCCG'96)*, vol. 5 of *International Informatics Series*, pp. 295–299, Carleton University Press, 1996.

[24] T. C. Biedl, Optimal orthogonal drawings of triconnected plane graphs. In R. Karlsson and A. Lingas, eds., *Proc. 5th Scandinavian Worksop on Algorithm Theory (SWAT'96)*, vol. 1097 of *Lecture Notes in Comput. Sci.*, pp. 333–344, Springer, Berlin, 1996.

[25] T. C. Biedl, New lower bounds for orthogonal drawings. *J. Graph Algorithms Appl.*, **2(7)**:1–31, 1998.

[26] T. C. Biedl, Relating bends and size in orthogonal graph drawings. *Inform. Process. Lett.*, **65**:111–115, 1998.

[27] T. C. Biedl, Three approaches to 3D-orthogonal box-drawings. In [215], pp. 30–43.

[28] T. C. Biedl and G. Kant, A better heuristic for orthogonal graph drawings. In J. van Leeuwen, ed., *Proc. Algorithms: 2nd Annual European Symp. (ESA'94)*, vol. 855 of *Lecture Notes in Comput. Sci.*, pp. 124–135, Springer, Berlin, 1994.

[29] T. C. Biedl and G. Kant, A better heuristic for orthogonal graph drawings. *Comput. Geom.*, **9(3)**:159–180, 1998.

[30] T. C. Biedl and M. Kaufmann, Area-efficient static and incremental graph drawings. In R. Burkhard and G. Woeginger, eds., *Proc. Algorithms: 5th Annual European Symp. (ESA'97)*, vol. 1284 of *Lecture Notes in Comput. Sci.*, pp. 37–52, Springer, Berlin, 1997.

[31] T. C. Biedl, B. Madden, and I. G. Tollis, The three-phase method: A unified approach to orthogonal graph drawing. In [69], pp. 391–402.

[32] T. C. Biedl, T. Shermer, S. Whitesides, and S. Wismath, Orthogonal 3-D graph drawing. In [69], pp. 76–86.

[33] T. C. Biedl, T. Shermer, S. Whitesides, and S. Wismath, Bounds for orthogonal 3-D graph drawing. *J. Graph Algorithms Appl.*, **3(4)**:63–79, 1999.

[34] T. C. Biedl, T. Thiele, and D. R. Wood, Three-dimensional orthogonal graph drawing with optimal volume. In [152].

[35] N. Biggs, Some heuristics for graph colouring. In R. Nelson and R. Wilson, eds., *Graph colourings*, pp. 87–96, Longman, New York, 1990.

[36] M. Blum, R. W. Floyd, V. Pratt, R. L. Rivest, and R. E. Tarjan, Time bounds for selection. *J. Comput. System Sci.*, **7**:448–461, 1973.

[37] P. Bose, A. Dean, J. Hutchinson, and T. Shermer, On rectangle visibility graphs. In [159], pp. 25–44.

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

[39] P. Bose, F. Gómez, P. A. Ramos, and G. T. Toussaintt, Drawing nice projections of objects in space. In [41], pp. 52–63.

[40] P. Bose, A. Josefczyk, J. Miller, and J. O'Rourke, K_{42} is a box visibility graph. Tech. Rep. 034, Smith College, 1994.

[41] F. J. Brandenburg, ed., *Proc. Graph Drawing: Symp. on Graph Drawing (GD'95)*, vol. 1027 of *Lecture Notes in Comput. Sci.*, Springer, Berlin, 1996.

[42] U. Brandes and D. Wagner, Dynamic grid embedding with few bends and changes. In [57], pp. 89–98.

[43] D. Brelaz, New methods to color the vertices of a graph. *Comm. of the ACM*, **22**:251–256, 1979.

[44] S. S. Bridgeman, J. Fanto, A. Garg, R. Tamassia, and L. Vismara, InteractiveGiotto: An algorithm for interactive orthogonal graph drawing. In [69], pp. 303–308.

[45] G. R. Brightwell and E. R. Scheinerman, Representations of planar graphs. *SIAM J. Disc. Math.*, **6(2)**:214–229, 1993.

[46] C. Bron and J. Kerbosch, Finding all cliques of an undirected graph. *Comm. ACM*, **16(9)**:575–577, 1973.

[47] R. L. Brooks, On colouring the nodes of a network. *Proc. Cambridge Philos. Soc.*, **37**:194–197, 1941.

[48] I. Bruss and A. Frick, Fast interactive 3-D graph visualization. In [41], pp. 99–110.

[49] V. Bryant, A characterisation of some 2-connected graphs and a comment on an algorithmic proof of Brooks' theorem. *Discrete Math.*, **158**:279–281, 1996.

[50] T. Calamoneri and R. Petreschi, An efficient orthogonal grid drawing algorithm for cubic graphs. In D. Ding-Zhu and L. Ming, eds., *Proc. Computing and Combinatorics (COCOON'95)*, vol. 959 of *Lecture Notes in Comput. Sci.*, pp. 31–40, Springer, Berlin, 1995.

[51] T. Calamoneri and R. Petreschi, A parallel algorithm for orthogonal grid drawings of cubic graphs. In A. D. Santis, ed., *Proc. 5th Italian Conf. on Theoretical Computer Science*, pp. 118–133, World Sci. Publishing, New Jersey, 1996.

[52] R. Carraghan and P. M. Pardalos, An exact algorithm for the maximum clique problem. *Oper. Res. Lett.*, **9**:375–382, 1990.

[53] G. Chartrand and L. Lesniak, *Graphs and Digraphs.* Chapman and Hall, London, 1996.

[54] J. Cheriyan and J. H. Reif, Directed s-t numberings, rubber bands, and testing k-vertex connectivity. *Combinatorica*, **14(4)**:435–451, 1994.

[55] K. Chilakamarri, N. Dean, and M. Littman, Three-dimensional Tutte embedding. *Cong. Numer.*, **107**:129–140, 1995.

[56] M. Chrobak, M. Goodrich, and R. Tamassia, Convex drawings of graphs in two and three dimensions. In *Proc. 12th Annual ACM Symp. on Comput. Geom.*, pp. 319–328, 1996.

[57] K. Y. Chwa and O. H. Ibarra, eds., *Proc. Algorithms and Computation (ISAAC'98)*, vol. 1533 of *Lecture Notes in Comput. Sci.*, Springer, Berlin, 1998.

[58] M. Closson, S. Gartshore, J. Johansen, and S. K. Wismath, Fully dynamic 3-dimensional orthogonal graph drawing. In [136], pp. 49–58.

[59] F. Cobos, J. Dana, F. Hurtado, A. Márquez, and F. Mateos, On a visibility representation of graphs. In [41], pp. 152–161.

[60] R. F. Cohen, P. Eades, T. Lin, and F. Ruskey, Three-dimensional graph drawing. *Algorithmica*, **17(2)**:199–208, 1996.

[61] Y. Colin de Verdière, Sur un nouvel invariant des graphes et un critère de planarité. *J. Combin. Theory Ser. B*, **50(1)**:11–21, 1990.

[62] Y. Colin de Verdière, On a new graph invariant and a criterion for planarity. In N. Robertson and P. D. Seymour, eds., *Graph structure theory. Proc. of AMS-IMS-SIAM Joint Summer Research Conf. on Graph Minors*, vol. 147 of *Contempory Mathematics*, pp. 137–147, American Mathematical Society, Rhode Island, 1993.

[63] J. H. Conway and C. M. Gordon, Knots and links in spatial graphs. *J. Graph Theory*, **7**:445–453, 1983.

[64] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms.* McGraw-Hill, New York, 1990.

[65] I. F. Cruz and J. P. Twarog, 3D graph drawing with simulated annealing. In [41], pp. 162–165.

[66] H. de Fraysseix, J. Pach, and R. Pollack, How to draw a planar graph on a grid. *Combinatorica*, **10**:41–51, 1990.

[67] A. Dean and J. Hutchinson, Rectangle-visibility representations of bipartite graphs. *Discrete Appl. Math.*, **75(1)**:9–25, 1997.

[68] F. Dehne, A. Rau-Chaplin, J. R. Sack, and R. Tamassia, eds., *Proc. Algorithms and Data Structures: 5th International Workshop (WADS'97)*, vol. 1272 of *Lecture Notes in Comput. Sci.*, Springer, Berlin, 1997.

[69] G. Di Battista, ed., *Proc. Graph Drawing: 5th International Symp. (GD'97)*, vol. 1353 of *Lecture Notes in Comput. Sci.*, Springer, Berlin, 1998.

[70] G. Di Battista, W. Didimo, M. Patrignani, and M. Pizzonia, Orthogonal and quasi-upward drawings with vertices of arbitrary size. In [136], pp. 297–310.

[71] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs.* Prentice-Hall, New Jersey, 1999.

[72] G. Di Battista, A. Garg, G. Liotta, R. Tamassia, E. Tassinari, and F. Vargiu, An experimental comparison of four graph drawing algorithms. *Comput. Geom.*, **7(5-6)**:303–325, 1997.

[73] G. Di Battista, G. Liotta, and F. Vargiu, Spirality and optimal orthogonal drawings. *SIAM J. Comput.*, **27(6)**:1764–1811, 1998.

[74] G. Di Battista, M. Patrignani, and F. Vargiu, A split&push approach to 3D orthogonal drawing. In [215], pp. 87–101.

[75] W. Didimo and G. Liotta, Computing orthogonal drawings in a variable embedding setting. In [57], pp. 79–88.

[76] R. Diestel, *Graph Theory.* Springer, New York, 1997.

[77] P. F. Dietz and D. D. Sleator, Two algorithms for maintaining order in a list. In *Proc. 19th Annual ACM Symp. Theory Comput. (STOC'87)*, pp. 365–372, 1987.

[78] D. Dolev, F. Leighton, and H. Trickey, Planar embeddings of planar graphs. In [174], pp. 147–161.

[79] P. Eades and Q. Feng, Drawing clustered graphs on an orthogonal grid. In [69], pp. 146–157.

[80] P. Eades and P. Garvan, Drawing stressed planar graphs in three dimensions. In [41], pp. 212–223.

[81] P. Eades, M. E. Houle, and R. Webber, Finding the best viewpoints for three-dimensional graph drawings. In [69], pp. 87–98.

[82] P. Eades and X. Lin, A heuristic for the feedback arc set problem. *Australas. J. Combin.*, **12**:15–25, 1995.

[83] P. Eades and X. Lin, Spring algorithms and symmetry. In T. Jiang and D. T. Lee, eds., *Proc. Computing and Combinatorics: Third Annual International Conf. (COCOON'97)*, vol. 1276 of *Lecture Notes in Comput. Sci.*, pp. 202–211, Springer, Berlin, 1997.

[84] P. Eades, X. Lin, and W. F. Smyth, A fast and effective heuristic for the feedback arc set problem. *Inform. Process. Lett.*, **47(6)**:319–323, 1993.

[85] P. Eades, C. Stirk, and S. Whitesides, The techniques of Komolgorov and Bardzin for three dimensional orthogonal graph drawings. *Inform. Proc. Lett.*, **60(2)**:97–103, 1996.

[86] P. Eades, A. Symvonis, and S. Whitesides, Two algorithms for three dimensional orthogonal graph drawing. In [159], pp. 139–154.

[87] P. Eades, A. Symvonis, and S. Whitesides, Three dimensional orthogonal graph drawing algorithms. *Discrete Applied Math.*, **103**:55–87, 2000.

[88] T. Endo, The pagenumber of toroidal graphs is at most seven. *Discrete Math.*, **175(1-3)**:87–96, 1997.

[89] P. Erdős, F. Harary, and W. T. Tutte, On the dimension of a graph. *Mathematika*, **12**:118–122, 1965.

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

[91] S. Even and G. Granot, Rectilinear planar drawings with few bends in each edge. Tech. Rep. 797, Computer Science Department, Technion, Israel Inst. of Tech., 1994.

[92] S. Even and G. Granot, Grid layouts of block diagrams — bounding the number of bends in each connection (extended abstract). In [204], pp. 64–75.

[93] S. Even and R. E. Tarjan, Computing an st-numbering. *Theoret. Comput. Sci.*, **2**:339–344, 1976.

[94] I. Fáry, On straight line representation of planar graphs. *Acta Sci. Math. Szeged*, **11**:229–233, 1948.

[95] S. Fekete, M. Houle, and S. Whitesides, New results on a visibility representation of graphs in 3D. In [41], pp. 234–241.

[96] S. Fekete and H. Meijer, Rectangle and box visibility graphs in 3D. *Internat. J. Comput. Geom. Appl.*, **9(1)**:1–27, 1999.

[97] Q. Feng, *Algorithms for Drawing Clustered Graphs.* Ph.D. thesis, The University of Newcastle, Australia, 1997.

[98] H. Fleischner, *Eulerian graphs and related topics*, vol. 45 of *Annals of Discrete Mathematics.* North-Holland, New York, 1990.

[99] H. Fleischner and M. Stiebitz, A solution to a colouring problem of P. Erdős. *Discrete Math.*, **101**:39–48, 1992.

[100] U. Fössmeier, Interactive orthogonal graph drawing: Algorithms and bounds. In [69], pp. 111–123.

[101] U. Fössmeier, C. Hess, and M. Kaufmann, On improving orthogonal drawings: the 4M-algorithm. In [215], pp. 125–137.

[102] U. Fössmeier, G. Kant, and M. Kaufmann, 2-visibility drawings of planar graphs. In [159], pp. 155–158.

[103] U. Fössmeier and M. Kaufmann, Drawing high degree graphs with low bend numbers. In [41], pp. 254–266.

[104] U. Fössmeier and M. Kaufmann, Algorithms and area bounds for nonplanar orthogonal drawings. In [69], pp. 134–145.

[105] M. R. Garey and D. S. Johnson, *Computers And Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.

[106] A. Garg and R. Tamassia, On the computational complexity of upward and rectilinear planarity testing (extended abstract). In [204], pp. 286–297.

[107] A. Garg and R. Tamassia, A new minimum cost flow algorithm with applications to graph drawing. In [159], pp. 201–216.

[108] A. Garg, R. Tamassia, and P. Vocca, Drawing with colors. In J. Diaz and M. Serna, eds., *Proc. Algorithms: 4th Annual European Symp. (ESA'96)*, vol. 1136 of *Lecture Notes in Comput. Sci.*, pp. 12–26, Springer, Berlin, 1996.

[109] M. A. Garrido and A. Márquez, Embedding a graph in the grid of a surface with the minimum number of bends is NP-hard. In [69], pp. 124–133.

[110] P. L. Garvan, Drawing and labelling graphs in three-dimensions. In M. Patel, ed., *Proc. 20th Australasian Comput. Sci. Conf. (ACSC'97)*, vol. 19 (1) of *Australian Comput. Sci. Comms.*, pp. 83–91, Macquarie University, 1997.

[111] A. Gregori, Unit-length embedding of binary trees on a square grid. *Inform. Process. Lett.*, **31(4)**:167–173, 1989.

[112] M. Grötschel, L. Lovász, and A. Schrijver, Polynomial algorithms for perfect graphs. *Ann. Discrete Math.*, **21**:325–356, 1984.

[113] K. Hagihara, N. Tokura, and N. Suzuki, Graph embedding on a three-dimensional model. *Systems-Comput.-Controls*, **14(6)**:58–66, 1983.

[114] P. Hall, On representation of subsets. *J. London Math. Soc.*, **10**:26–30, 1935.

[115] L. S. Heath and S. Istrail, The pagenumber of genus g graphs is $O(g)$. *J. Assoc. Comput. Mach.*, **39(3)**:479–501, 1992.

[116] P. J. Heawood, Map colour theorem. *Quart. J. Pure Appl. Math.*, **24**:332–338, 1890.

[117] A. J. W. Hilton and D. de Werra, A sufficient condition for equitable edge-colourings of simple graphs. *Discrete Math.*, **128(1-3)**:179–201, 1994.

[118] C. Holroyd, *Michael Angelo Buonarrotti; with translations of the Life of the master by his scholar, Ascanio Condivi; and Three dialogues from the Portuguese by Francisco D'Ollanda.* Duckworth, London, 2nd edn., 1911.

[119] J. Hopcroft and R. Tarjan, Efficient planarity testing. *J. Assoc. Comput. Mach.*, **21**:549–568, 1974.

[120] M. E. Houle and R. Webber, Approximation algorithms for finding best viewpoints. In [215], pp. 210–223.

[121] F. Jaeger, A survey of the cycle double cover conjecture. *Annals of Discrete Mathematics*, **27**:1–12, 1985.

[122] D. S. Johnson and M. A. Trick, eds., *Cliques, Coloring, and Satisfiability: Second DIMACS Challenge.* DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, Rhode Island, 1996.

[123] T. Kamada and S. Kawai, A simple method for computing general position in displaying three-dimensional objects. *Computer Vision, Graphics, and Image Processing*, **41(1)**:43–56, 1988.

[124] G. Kant, Drawing planar graphs using the canonical ordering. *Algorithmica*, **16(1)**:4–32, 1996.

[125] R. M. Karp, Reducibility among combinatorial problems. In *Complexity of Computer Communications*, pp. 85–103, Plenum Press, New York, 1972.

[126] N. D. Kazarinoff, *Analytic Inequalities.* Holt, Rinehart and Winston, New York, 1961.

[127] D. Kleitman and M. Krieger, An optimal bound for two dimensional bin packing. In *16th Annual Symp. on Foundations of Computer Science (FOCS'75)*, pp. 163–168, IEEE, 1975.

[128] D. E. Knuth, Computer-drawn flowcharts. *Commun. ACM*, **6**:555–563, 1963.

[129] P. Koebe, Kontaktprobleme der konformen Abbildung. *Berichte über die Verhandlungen der Sächsischen Akademie der Wissenschaften zu Leipzig. Math.-Phys. Klasse*, **88**:141–164, 1936.

[130] T. Kohara and S. Suzuki, Some remarks on knots and links in spatial graphs. In *Knots 90 (Osaka, 1990)*, pp. 435–445, de Gruyter, Berlin, 1992.

[131] H. Koike, The role of another spatial dimension in software visualization. *ACM Trans. Inf. Syst.*, **11(3)**:266–286, 1993.

[132] A. N. Kolmogorov and Y. M. Barzdin, On the realization of nets in 3-dimensional space. *Problems in Cybernetics*, **8**:261–268, 1967.

[133] D. König, Über Graphen und ihre Anwendungen auf Determinantentheorie und Mengenlehre. *Mathematische Annalen*, **77**, 1916.

[134] A. Kotlov, L. Lovász, and S. Vempala, The Colin de Verdière number and sphere representations of a graph. *Combinatorica*, **17(4)**:483–521, 1997.

[135] M. R. Kramer and J. van Leeuwen, The complexity of wire-routing and finding minimum area layouts for arbitrary VLSI circuits. In [174], pp. 129–146.

[136] J. Kratochvil, ed., *Proc. Graph Drawing: 7th International Symp. (GD'99)*, vol. 1731 of *Lecture Notes in Comput. Sci.*, Springer, Berlin, 1999.

[137] K. Kuratowski, Sur le probléme des courbes gauches en topologie. *Fund. Math.*, **16**:271–283, 1930.

[138] M. Leeser, W. M. Meleis, M. M. Vai, S. Chiricescu, W. Xu, and P. Zavracky, Rothko: A three dimensional FPGA. *IEEE Design & Test of Computers*, **15(1)**:16–23, 1998.

[139] M. Leeser, W. M. Meleis, M. M. Vai, and P. Zavracky, Rothko: A three dimensional FPGA architecture, its fabrication, and design tools. In W. Luk, P. Y. K. Cheung, and M. Glesner, eds., *Proc. of 7th International Workshop on Field Programmable Logic and Applications (FPL'97)*, vol. 1304 of *Lecture Notes in Computer Science*, pp. 21–30, Springer, Berlin, 1997.

[140] F. T. Leighton and A. L. Rosenberg, Three-dimensional circuit layouts. *SIAM J. Comput.*, **15(3)**:793–813, 1986.

[141] C. E. Leiserson, Area-efficient graph layouts (for VLSI). In *Proc. 21st Annual Symp. on Foundations of Computer Science (FOCS'80)*, pp. 270–281, IEEE, 1980.

[142] A. Lempel, S. Even, and I. Cederbaum, An algorithm for planarity testing of graphs. In *Theory of Graphs, International Symp. (Rome, 1966)*, pp. 215–232, Gordon and Breach, New York, 1967.

[143] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout.* John Wiley, New York, 1990.

[144] N. Linial, L. Lovász, and A. Wigderson, Rubber bands, convex embeddings and graph connectivity. *Combinatorica*, **8(1)**:91–102, 1988.

[145] Y. Liu, *Embeddability in Graphs.* Mathematics and Its Applications, Kluwer, Dordrecht, 1995.

[146] Y. Liu, A. Morgana, and B. Simeone, General theoretical results on rectilinear embeddability of graphs. *Acta Math. Appl. Sinica*, **7**:187–191, 1991.

[147] L. Lovász, Three short proofs in graph theory. *J. Combin. Theory Series B*, **19B**:269–271, 1975.

[148] L. Lovász and M. D. Plummer, *Matching Theory*, vol. 29 of *Annals of Discrete Mathematics.* North-Holland, Amsterdam, 1986.

[149] L. Lovász and A. Schrijver, A Borsuk theorem for antipodal links and a spectral characterization of linklessly embeddable graphs. *Proc. Amer. Math. Soc.*, **126(5)**:1275–1285, 1998.

[150] S. M. MALITZ, Genus g graphs have pagenumber $O(\sqrt{g})$. *J. Algorithms*, **17(1)**:85–109, 1994.

[151] S. M. MALITZ, Graphs with E edges have pagenumber $O(\sqrt{E})$. *J. Algorithms*, **17(1)**:71–84, 1994.

[152] J. MARKS, ed., *Proc. Graph Drawing: 8th International Symp. (GD'00)*, Lecture Notes in Comput. Sci., Springer, Berlin, to appear.

[153] W. M. MELEIS, M. LEESER, P. ZAVRACKY, AND M. M. VAI, Architectural design of a three dimensional FPGA. In *Proc. of 17th Conf. on Advanced Research in VLSI (ARVLSI'97)*, pp. 256–268, IEEE Computer Society Press, 1997.

[154] K. MISUE, P. EADES, W. LAI, AND K. SUGIYAMA, Layout adjustment and the mental map. *J. Visual Languages and Computing*, **6**:183–210, 1995.

[155] B. MOHAR, A linear time algorithm for embedding graphs in an arbitrary surface. *SIAM J. Discrete Math.*, **12(1)**:6–26, 1999.

[156] B. MONIEN, F. RAMME, AND H. SALMEN, A parallel simulated annealing algorithm for generating 3D layouts of undirected graphs. In [41], pp. 396–408.

[157] R. MOTWANI, A. RAGHUNATHAN, AND H. SARAN, Constructive results from graph minors: Linkless embeddings. In *Proc. 29th IEEE Symp. on Foundations of Computer Science (FOCS'88)*, pp. 398–409, IEEE, 1988.

[158] T. MUNZNER AND P. BURCHARD, Visualizing the structure of the World Wide Web in 3d hyperbolic space. In *Proc. VRML*, pp. 33–38, ACM Press, 1995.

[159] S. NORTH, ed., *Proc. Graph Drawing: Symp. on Graph Drawing (GD'96)*, vol. 1190 of *Lecture Notes in Comput. Sci.*, Springer, Berlin, 1997.

[160] D. I. OSTRY, *Some Three-Dimensional Graph Drawing Algorithms*. Master's thesis, Department of Computer Science and Software Engineering, The University of Newcastle, Australia, 1996.

[161] J. PACH, T. THIELE, AND G. TOTH, Three-dimensional grid drawings of graphs. In [69], pp. 47–51.

[162] A. Papakostas, J. M. Six, and I. G. Tollis, Experimental and theoretical results in interactive orthogonal graph drawing. In [159], pp. 371–386.

[163] A. Papakostas and I. G. Tollis, Improved algorithms and bounds for orthogonal drawings. In [204], pp. 40–51.

[164] A. Papakostas and I. G. Tollis, Orthogonal drawing of high degree graphs with small area and few bends. In [68], pp. 354–367.

[165] A. Papakostas and I. G. Tollis, Algorithms for area-efficient orthogonal drawings. *Comput. Geom.*, **9**:83–110, 1998.

[166] A. Papakostas and I. G. Tollis, Incremental orthogonal graph drawing in three dimensions. In [69], pp. 52–63.

[167] A. Papakostas and I. G. Tollis, Interactive orthogonal graph drawing. *IEEE Trans. Comput.*, **47(11)**:1297–1309, 1998.

[168] A. Papakostas and I. G. Tollis, Algorithms for incremental orthogonal graph drawing in three dimensions. *J. Graph Algorithms Appl.*, **3(4)**:81–115, 1999.

[169] A. Papakostas and I. G. Tollis, Efficient orthogonal drawings of high degree graphs. *Algorithmica*, **26**:100–125, 2000.

[170] P. M. Pardalos and G. P. Rodgers, A branch and bound algorithm for the maximum clique problem. *Computers Ops. Res.*, **19(5)**:363–375, 1992.

[171] P. M. Pardalos and J. Xue, The maximum clique problem. *J. Global Optim.*, **4**:301–328, 1994.

[172] J. Peterson, Die Theorie der regulären Graphen. *Acta. Math.*, **15**:193–220, 1891.

[173] F. P. Preparata, Optimal three-dimensional VLSI layouts. *Math. Systems Theory*, **16**:1–8, 1983.

[174] F. P. Preparata, ed., *Advances in Computing Research — Volume 2: VLSI Theory*, JAI Press, Connecticut, 1985.

[175] H. Purchase, Which aesthetic has the greatest effect on human understanding? In [69], pp. 248–261.

[176] H. C. Purchase, R. F. Cohen, and M. James, An experimental study of the basis for graph drawing algorithms. *ACM Journal of Experimental Algorithmics*, **2(4)**, 1997.

[177] N. R. Quinn Jr. and M. A. Breuer, A force directed component placement procedure for printed circuit boards. *IEEE Transactions on Circuits and Systems*, **CAS-26(6)**:377–388, 1979.

[178] S. Rahman, S. Nakano, and T. Nishizeki, A linear algorithm for optimal orthogonal drawings of triconnected cubic plane graphs. In [69], pp. 99–110.

[179] S. Reiss, 3-D visualization of program information. In [204], pp. 12–24.

[180] G. G. Robertson, J. D. Mackinlay, and S. K. Card, Cone trees: Animated 3D visualizations of hierarchical information. In S. Robertson, G. Olsen, and J. Olsen, eds., *Proc. Human Factors in Computing Systems (CHI'91)*, pp. 189–194, ACM, New York, 1991.

[181] N. Robertson, D. Sanders, P. Seymour, and R. Thomas, The four-colour theorem. *J. Combin. Theory Ser. B*, **70(1)**:2–44, 1997.

[182] N. Robertson and P. Seymour, Graph minors XX: Wagner's conjecture, manuscript.

[183] N. Robertson and P. Seymour, Graph minors VIII. A Kuratowski theorem for general surfaces. *J. Combin. Theory Ser. B*, **48(2)**:255–288, 1990.

[184] N. Robertson, P. Seymour, and R. Thomas, Sachs' linkless embedding conjecture. *J. Combin. Theory Series B*, **64**:185–227, 1995.

[185] A. L. Rosenberg, Three-dimensional integrated circuitry. In H. T. Kung, B. Sproull, and G. Steele, eds., *Proc of Conf. on VLSI Systems and Computations*, pp. 69–80, Computer Science Press, Maryland, 1981.

[186] A. L. Rosenberg, Three-dimensional VLSI: A case study. *J. ACM*, **30(2)**:397–416, 1983.

[187] P. Rosenstiehl and R. E. Tarjan, Rectilinear planar layouts and bipolar orientations of planar graphs. *Discrete Comput. Geom.*, **1(4)**:343–353, 1986.

[188] H. Sachs, On a spatial analogue of Kuratowski's theorem on planar graphs — an open problem. In M. Borowiecki, J. Kennedy, and M. Syslo, eds., *Proc. of Conf. on Graph Theory (Lagow, 1981)*, vol. 1018 of *Lecture Notes in Mathematics*, pp. 230–241, Springer, 1983.

[189] H. Sachs, Elementary proof of the cycle-plus-triangles theorem. In *Combinatorics, Paul Erdős is Eighty*, vol. 1, pp. 347–359, Bolyai Society of Mathematical Studies, 1993.

[190] M. Schäffter, Drawing graphs on rectangular grids. *Discrete Applied Math.*, **63**:75–89, 1995.

[191] M. Schlag, F. Luccio, P. Maestrini, D. Lee, and C. Wong, A visibility problem in VLSI layout compaction. In [174], pp. 259–282.

[192] W. Schnyder, Embedding planar graphs on the grid. In *Proc. 1st ACM-SIAM Symp. Discrete Algorithms*, pp. 138–148, 1990.

[193] M. Schönert *et al.*, *GAP — Groups, Algorithms and Programming*. Rheinisch Westfälische Technische Hochschule, Aachen, Germany, 1995.

[194] A. Schrijver, Bipartite edge coloring in $O(\Delta m)$ time. *SIAM J. Comput.*, **28(3)**:841–846, 1998.

[195] T. Shermer, Block visibility representations III: External visibility and complexity. In Kranakis, Fiala, and Sack, eds., *Proc of 8th Canadian Conference on Computational Geometry*, vol. 5 of *International Informatics Series*, pp. 234–239, Carleton University Press, 1996.

[196] M. Shimabara, Knots in certain spatial graphs. *Tokyo J. Math.*, **11(2)**:405–413, 1988.

[197] J. M. Six, K. G. Kakoulis, and I. G. Tollis, Refinement of orthogonal graph drawings. In [215], pp. 302–315.

[198] S. K. Stein, Convex maps. *Proc. Amer. Math. Soc.*, **2**:464–466, 1951.

[199] J. A. Storer, On minimal-node-cost planar embeddings. *Networks*, **14**:181–212, 1984.

[200] R. Tamassia, On embedding a graph in the grid with the minimum number of bends. *SIAM J. Comput.*, **16(3)**:421–443, 1987.

[201] R. Tamassia, G. Di Battista, and C. Batini, Automatic graph drawing and readability of diagrams. *IEEE Trans. on Systems, Man and Cybernetics*, **18(1)**:61–79, 1988.

[202] R. Tamassia and I. G. Tollis, A unified approach to visibility representations of planar graphs. *Discrete Comput. Geom.*, **1**:321–341, 1986.

[203] R. Tamassia and I. G. Tollis, Planar grid embedding in linear time. *IEEE Trans. Circuits Systems*, **36(9)**:1230–1234, 1989.

[204] R. Tamassia and I. G. Tollis, eds., *Proc. Graph Drawing: DIMACS International Workshop (GD'94)*, vol. 894 of *Lecture Notes in Comput. Sci.*, Springer, Berlin, 1995.

[205] R. Tamassia, I. G. Tollis, and J. S. Vitter, Lower bounds for planar orthogonal drawings of graphs. *Inform. Process. Lett.*, **39(1)**:35–40, 1991.

[206] R. E. Tarjan and A. E. Trojanowski, Finding a maximum independent set. *SIAM J. Comput.*, **6(3)**:537–546, 1977.

[207] C. Thomassen, The Jordon-Schönflies Curve Theorem and the classification of surfaces. *Amer. Math. Monthly*, **99**:116–130, 1992.

[208] W. T. Tutte, Convex representations of graphs. *Proc. London Math. Soc.*, **10(3)**:304–320, 1960.

[209] W. T. Tutte, How to draw a graph. *Proc. London Math. Soc.*, **13(3)**:743–768, 1963.

[210] I. Veretennicoff, H. Thienpont, B. Dhoedt, R. Baets, J. Van Campenhout, H. Van Marck, H. Neefs, and J. Depreitere, An optoelectronic 3-D field programmable gate array. In R. W. Hartenstein and M. Z. Servit, eds., *Field-programmable logic. Architectures, synthesis and applications*, vol. 849 of *Lecture Notes in Comput. Sci.*, pp. 352–360, Springer, Berlin, 1994.

[211] K. Wagner, Bemerkung zum Vierfarbenproblem. *Jber. Deutsc. Math. Verein.*, **46**:26–32, 1936.

[212] K. Wagner, Über eine Eigenschaft der ebene Komplexe. *Math. Ann.*, **114**:570–590, 1937.

[213] C. Ware and G. Franck, Viewing a graph in a virtual reality display is three times as good as a 2D diagram. In A. L. Ambler and T. D. Kimura, eds., *Proc. IEEE Symp. Visual Languages (VL'94)*, pp. 182–183, IEEE, 1994.

[214] C. Ware, D. Hui, and G. Franck, Visualizing object oriented software in three dimensions. In *Proc. IBM Centre for Advanced Studies Conf. (CASCON'93)*, pp. 1–11, 1993.

[215] S. Whitesides, ed., *Proc. Graph Drawing: 6th International Symp. (GD'98)*, vol. 1547 of *Lecture Notes in Comput. Sci.*, Springer, Berlin, 1998.

[216] S. Wismath, Characterizing bar line-of-sight graphs. In *Proc. 1st ACM Symp. Comput. Geom.*, pp. 147–152, ACM Press, New York, 1985.

[217] D. R. Wood, An algorithm for finding a maximum clique in a graph. Tech. Rep. 96/260, Department of Computer Science, Monash University, Australia, 1996, (`http://www.csse.monash.edu.au/publications`).

[218] D. R. Wood, An algorithm for finding a maximum clique in a graph. *Oper. Res. Lett.*, **19(5)**:211–217, 1997.

[219] D. R. Wood, On higher-dimensional orthogonal graph drawing. In J. Harland, ed., *Proc. Computing: the Australasian Theory Symp. (CATS'97)*, vol. 19(2) of *Austral. Comput. Sci. Comm.*, pp. 3–8, 1997.

[220] D. R. Wood, Towards a 2-bends algorithm for three-dimensional orthogonal graph drawing. In V. Estivill-Castro, ed., *Proc. Australasian Workshop on Combinatorial Algorithms (AWOCA'97)*, pp. 102–107, Queensland University of Technology, Australia, 1997.

[221] D. R. Wood, An algorithm for three-dimensional orthogonal graph drawing. In [215], pp. 332–346.

[222] D. R. Wood, Multi-dimensional orthogonal graph drawing with small boxes (extended abstract). In [136], pp. 311–222.

[223] D. R. Wood, A new algorithm and open problems in three-dimensional orthogonal graph drawing. In R. Raman and J. Simpson, eds., *Proc. Australasian Workshop on Combinatorial Algorithms (AWOCA'99)*, pp. 157–167, Curtin University of Technology, Perth, 1999.

[224] D. R. Wood, Lower bounds for the number of bends in three-dimensional orthogonal graph drawings. In [152].

[225] J. Xue, Edge-maximal triangulated subgraphs and heuristics for the maximum clique problem. *Networks*, **24(2)**:109–120, 1994.

[226] M. Yannakakis, Embedding planar graphs in four pages. *J. Comput. System Sci.*, **38**:36–67, 1986.

Index