Two-and-a-Half-Dimensional Visualisation of Relational Networks

by

Tim Dwyer

Bachelor Computer Science (Honours), 2000

A thesis submitted to
The School of Information Technologies
The University of Sydney
for the degree of
DOCTOR OF PHILOSOPHY

September, 2004

I hereby certify that the work embodied in this thesis is the result of original research and has not been submitted for a higher degree to any other University or Institution.

_____

Tim Dwyer

Sydney
9th June 2005

For Amelia, Mum and Dad.

Acknowledgements

I would like to thank Professor Peter Eades for being a *super* supervisor. Collaboration of the following people on the various case studies is also gratefully acknowledged: Dr. Falk Schreiber, Professor Ulrik Brandes, Dr. Hardy Rolletschek and Dr. David Gallagher. It was a privilege to work with all of them.

The following clever people provided sage PhD advice at various times: Doctors Masahiro Takatsuka, Carsten Friedrich, Aaron Quigley, Keith Nesbitt, Seokhee Hong, Joe Thurbon, Hugo do Nascimento, Nikola Nikolov and Richard Webber.

My thanks to Ying Xin Wu and Michael Till for their help with the PORTFOLIOSPACE EXPLORER implementation. Thanks, also, to all those who participated in experiments and system evaluations.

Thankyou to Remo Ziegler and Adel Ahmed for being genii with 3DStudioMax; Kevin Pulo for providing a print server and linux support; Kathryn Kasmarik for her help with experimental design and Damian Merrick for a steady supply of lollies. Thanks to Xiaobin Shen for a similar supply of weird Chinese candy. Thanks to all other members of the NICTA HUM and the University of Sydney Infovis research groups for their support and good humour.

Thanks to Margaret Roche and Dr. Kendall St. Clair for their help with final thesis preparation. Barbara Munday, Sharon Chambers, Josephine Spongberg, Michael McCabe and other administrative staff at NICTA and the University of Sydney School of Information Techology all greased the sticky wheels of university procedure for me on many occasions. Remo Di Giovanni helped in obtaining materials for the models used in experiments and also offered helpful advice in their construction.

Thanks to the Capital Markets CRC and Australian Federal Government for keeping me liquid throughout my studies. Thanks to the National ICT Australia for giving me a desk and computer resources. Thanks also to CMCRC, Sirca and Computershare Analytics Ltd. for providing data for my studies. Thanks to all others who gave permission to use their images or data in this thesis.

They are acknowledged individually where appropriate.

Finally, huge thanks go to Amelia de Bie and my parents for their love and support throughout.

Contents

List of Figures

List of Tables

Abstract

This thesis explores methods for creating three-dimensional information visualisations that are compatible with the limitations of human spatial perception. The concept of *two-and-a-half-dimensional visualisation* is loosely defined as visualisation which treats the third spatial dimensional in a different way to the other two. A more precise definition for $2\frac{1}{2}$D visualisation is introduced; followed by some guidelines for designing $2\frac{1}{2}$D visualisations. These design guidelines are explored in a quantitative experiment.

The definition for $2\frac{1}{2}$D visualisation is then extended to $2\frac{1}{2}$*D graph visualisation*: that is, relational-network visualisation which follows the $2\frac{1}{2}$D design guidelines. A $2\frac{1}{2}$D graph visualisation paradigm called *stratified graph visualisation* is introduced for the visualisation of evolving graphs or for the comparison of related sets of graphs. The issue of automatically finding a good layout for stratified graph visualisations is examined. A number of extensions to general graph layout algorithms are proposed with the intention of providing improved results for stratified graph visualisations.

These concepts are explored with case studies in three different application domains: *fund manager movement*, *metabolic pathways* and *phylogenetic trees*.

CHAPTER 1

Introduction

> "Program a map to display frequency of data exchange, every thousand megabytes a single pixel on a very large screen. Manhattan and Atlanta burn solid white. Then they start to pulse, the rate of traffic threatening to overload your simulation. Your map is about to go nova. Cool it down. Up your scale. Each pixel a million megabytes. At a hundred million megabytes per second, you begin to make out certain blocks in midtown Manhattan, outlines of hundred-year-old industrial parks ringing the old core of Atlanta. . ." — William Gibson, *Neuromancer* [79]

1.1 Visualisation

At its most basic, data visualisation involves making pictures to represent data. Two goals for this process are commonly cited. These are briefly explained here with references to two famous nineteenth century examples that are almost mandatory in any introduction to visualisation[1]:

Communication — using visualisation to convey knowledge about data to an audience. This practice is also described by Tufte as *visual explanation* [187]. The classic example is the visual mapping by Minard [160] of the progress of Napoleon's army in its failed 1812–1813 campaign through the Russian winter. Relationships between six variables: time; temperature; the size of the shrinking army; two-dimensional geographic location; and direction of movement — are elegantly demonstrated in a single picture.

Investigation — visually exploring an unknown data-set by applying various mappings of the data to pictures. This process is also called *visual data mining* [176]. The hope is that interesting and possibly unexpected properties of

[1] Tufte [188], Spence [177] and Wainer [191] each discuss the Minard and Snow examples at length.

the data will become obvious with an appropriate visual mapping. For example, a popular story in the visualisation literature describes Dr. John Snow's use of visualisation to discover a link between cholera and the London water supply. By plotting the position of homes of victims he discovered that they were clustered around a water pump [81].

As these examples show, people have been experimenting with both types of visualisation for a long time. However, the intrusion of computers into every aspect of our lives has made the problems involved in turning data into knowledge and then communicating that knowledge more critical, and more difficult, than ever before. Although computer technology has caused these problems it also potentially offers the solutions. Computers are constantly logging vast quantities of data about everything we do. The internet and the world-wide-web make all kinds of raw data available for analysis almost instantly. No human being can hope to sit down and analyse these hoards of information in the manner that Minard and Snow did — armed only with drawing implements and creativity. Thus, the modern idea of visualisation is inextricably linked to computers and computer graphics.

Although long used as a tool for analysis in the sciences, computer visualisation only formally became a field of research in its own right in the late 1980s [133]. Since then, according to a taxonomy given by Card et al. [28, p. 7], visualisation has split into two main subfields:

Scientific visualisation generally includes any sort of graphical representation of scientific data. Examples in the literature, however, are usually characterised by a fairly direct mapping from a physical geometry to a computer enhanced model and imagery. Scientific visualisation includes, for example: medical applications — such as reconstruction of MRI scans into artificial reality models; engineering applications — such as the study of air and fluid flows over surfaces; and in physics, the study of atmospheric models or astronomical data.

Information visualisation is generally concerned with more abstract data: that is, data not directly related to an underlying physical space or geometry. Examples include business data such as price/volume data, or age/population data. Although such data may include geographical attributes, relationships between other variables are generally the focus.

Thus, the designer of an information visualisation must find a mapping from an abstract information space to a geometry that can be viewed in one, two or three spatial dimensions. Data dimensions can be mapped to visual attributes such as colour, texture, transparency, animation, or even non-visual attributes such as sounds or force feedback (*haptics*). The primary concern of this thesis, however, is a method for most effectively utilising the third spatial dimension in information visualisation — particularly in the visualisation of *relational networks*; that is, networks of related concepts or data entities.

The remainder of this introductory chapter is structured as follows. In Section 1.2 a background into two- and three-dimensional computer graphics technology is given. A discussion of the use, and abuse, of three-dimensional (*3D*) computer graphics technology in the field of information visualisation is presented in Section 1.3; leading to the motivation for this thesis. A number of case studies are presented in the later chapters of this thesis. In Section 1.4 an introduction to the application domains considered in the case studies is given, again focusing on the use of 3D visualisation methods in these domains. In Section 1.5 the research methodology used in this work is discussed. Section 1.6 discusses the research contributions of this thesis and, finally, Section 1.7 outlines the organisation of the remaining chapters.

1.2 Three-dimensional Computer Graphics

In essence scientists have been visualising data since the first output devices were connected to valve-based logic engines in the 1940s and 1950s. The Hollywood stereotype of computers as imposing walls of blinking lights descends from these early visualisation experiments where lights were direct representations of the *on/off* states of bits. As computer graphics became more sophisticated, more abstract visualisation became possible.

Computer generated representations of a three-dimensional model are generally called *3D graphics*. Computers have been generating 3D graphics almost as long as two-dimensional (*2D*) graphics. In fact, the term *computer graphics* was first used in 1960 by Fetter [64], to describe his computer generated drawings of the 3D human form which were intended for studying human factors in aeroplane cockpit design for Boeing. The images were produced on a plotter and then photographed to produce a film animation.

This first production of an animated movie from a 3D computer model demonstrates an early understanding of a fundamental problem in 3D graphics: namely, that rendering a static image of

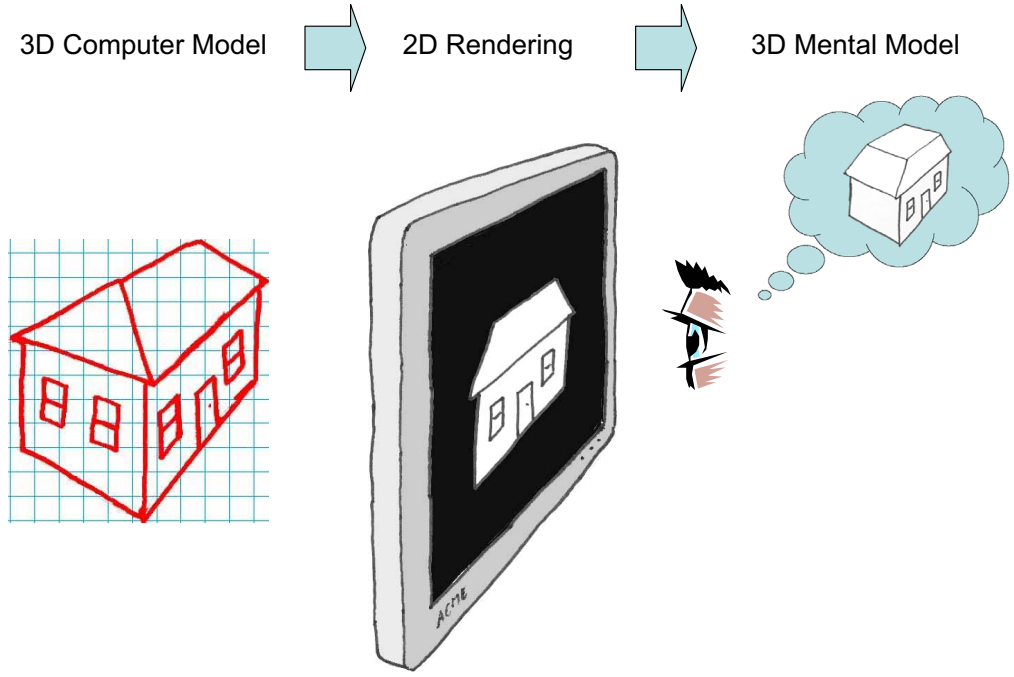


Figure 1.1: Rendering a static image of a 3D model on a 2D surface is a poor way to communicate the geometry of the model. Note that the mental model differs from the original computer model.

the 3D model on a 2D surface is a poor way to communicate the geometry of the 3D model to a viewer. Figure 1.1 shows the situation. Basically, all depth information and information about occluded parts of the model are lost in a 2D rendering. Rotating or flying around the model in an animation, such as the film produced by Fetter, gives the viewer a much better understanding of the underlying 3D geometry.

The ultimate way to study a 3D model is to interact with it as though it were an actual physical object. This need was recognised very early in the development of computer graphics. In the late 1960s Sutherland [181] created the first *virtual reality* system for interactively navigating around a 3D model. The system incorporated a head-mounted display coupled to a pole attached to a heavy computer system in the ceiling. Called the "Sword of Damocles" this pole was used to track the position and orientation of the user. The model was then rendered relative to the user's viewpoint providing a somewhat immersive experience.

Providing a completely immersive experience has proved to be a difficult task. It has become apparent that people are remarkably sensitive to artifacts such as slow refresh rates, poor resolution and imperfect stereo optics. Virtual-reality technology, however, has gradually improved with the availability of better head-mounted displays and better position tracking systems as well as im-

provements in the underlying computer graphics systems. Three-dimensional "printing" systems are available, providing a very good, albeit static, solution to this problem. The task of bringing similar realism to interactive models that can be manipulated and changed in real time is ongoing. Advances include *haptics* or *force-feedback* systems that allow viewers to actually touch and feel the 3D models; and *augmented reality* systems which overlay the user's view of the real world with images of virtual 3D models.

In short, 3D graphics technology has a decades-long history and has seen many improvements. In recent years particularly, industry commentators have noted that improvement in graphics processor power has exceeded Moore's law and that on-line photo-realistic rendering of movies will soon be possible [116]. While realistic virtual- and augmented-reality remains elusive in 2004, the technology moves forward at a pace that makes it seem inevitable that such environments will, one day, be available for applications such as information visualisation.

1.3 Motivation: two- and three-dimensional visualisation

The purpose of this thesis is to find methods for effectively utilising the third dimension that is available in modern computer graphics: that is, to find 3D visualisation paradigms that have concrete advantages over their 2D counterparts.

Since the early days of computer graphics research, advances in visualising 3D models have gone hand-in-hand with general advances in 2D graphics. In many fields 3D graphics has become a standard tool. In 2004 most action movies feature computer-generated imagery based on complex 3D models. Most engineers and architects would not now consider creating a physical prototype of a design without first creating a virtual model and analysing it on a graphics workstation. The motivation for using 3D graphics in these areas is clear: special effects in movies are intended to simulate physical action in the real, three-dimensional world; and Engineers and architects design three-dimensional objects. In both of these situations it makes sense to model the physical reality as closely as possible — this means modelling 3D objects in a 3D virtual space and visualising the results with computer graphics.

Similarly, in scientific visualisation there is often a clear-cut imperative for using 3D computer graphics. In visualising scientific data one is often overlaying a real-world 3D environment with extra information. Volumetric renderings, for example, in medical or geological applications are reconstructions of actual spatial geometry. Another situation where the move to 3D brings clear

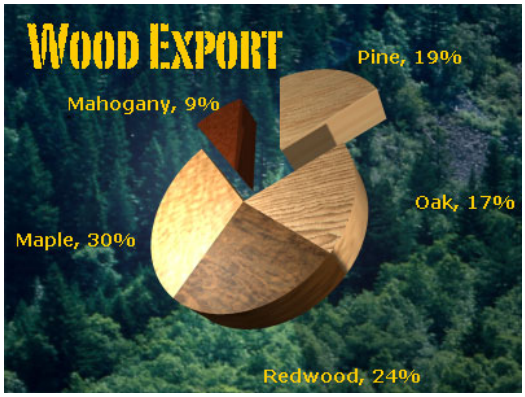
benefits is in mathematics visualisation: where visualising a function of two variables — for example, defining a surface — is simply less intuitive without 3D.

The question of whether to use 3D graphics in abstract information visualisation, however, is less clear cut. When information or data is called *abstract*, the description refers to a lack of a simple, obvious spatial mapping for these entities. In a visualisation of stock-price-by-industry data, for example, such a mapping can be more subjective. When visualising a network of abstract concepts, such as the network of synonyms in a thesaurus, the mapping for each of the concepts to a position in 2D or 3D space may be chosen completely arbitrarily. A wise designer attempts to find a spatial arrangement for the network that best shows its structure.

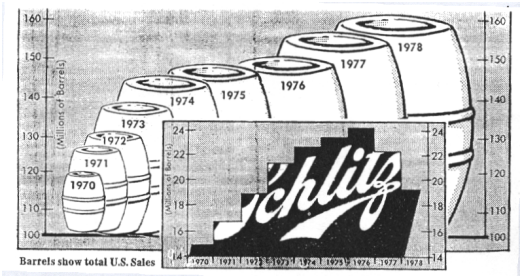
Advances in 3D graphics technology have also made the choice of mapping to a 2D or 3D visualisable space fairly arbitrary from a programmer's point of view. Debate about whether to use 3D graphics in information visualisation, however, tends to be simplistic and uncompromising on both sides. For example, many point out that a computer screen is fundamentally two dimensional, as is the printed page. Therefore why "mess around" with perspective and other depth cues when we can simply map directly to the dimensionality of the target media? However, new technologies such as augmented reality and 3D printing make this argument seem short sighted.

Enthusiasts for 3D information visualisation have claimed that human spatial memory performs better when spatial mappings of information are presented with perspective rendering. The hypothesis is that renderings that simulate depth of field are closer to the way we perceive the real world and are therefore more intuitive than flat renderings [5]. More-recent studies have supported this hypothesis [158, 183], but not without challenge [34, 33]. The latter pair of papers point out that the former studies introduce variables in their conversion of interfaces to 3D that were not controlled experimentally. Indeed, it seems a difficult theory to prove or disprove conclusively.

Another argument in support of 3D graphics for abstract information visualisation, has been its aesthetic appeal. A study by Levy et al. [124] found that undergraduate psychology students preferred to use 3D charts, such as the example shown in Figure 1.2, to present information to others. However, the report did not consider whether such preferences were subject to a fickle fashion sense. The study was completed in 1996 when such 3D charting support was relatively new to spreadsheet tools like EXCEL and may have still held some appeal due to novelty. Although they may be aesthetically pleasing, such 3D chart effects do not utilise the 3rd dimension in a way that increases the amount of information shown over flat versions and probably only makes them more confusing. Tufte [188, Page 118] labels these 3D effects, along with other useless decorations, as



(a) 2D chart extruded into 3D, rotated to an odd angle and projected in perspective.



(b) Use of the 3D barrels implies a cubing of the represented values. From the Washington Post as reproduced by Wainer [191].

Figure 1.2: 3D "Chart junk" might capture attention but it distorts the perceived values.

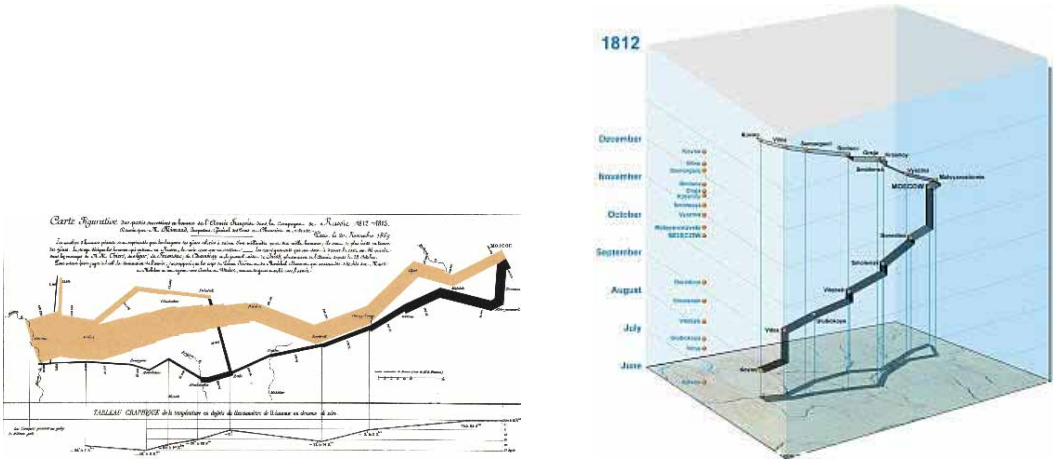
"chart junk".

The motivation for this thesis is that arguments such as those above are based on aesthetics and poorly understood cognitive models, and miss a fundamentally more important point. It is argued that the availability of an extra spatial dimension in modern computer graphics can provide concrete benefits to certain information visualisation tasks (for example, see Figure 1.3) but enthusiasm should be tempered by a knowledge of the limitations of human spatial perception. As discussed in Chapter 2, humans have better visual acuity in the view-plane than in depth of field. Problems with perspective distortion and occlusion are also fairly easy to demonstrate, although the interactive systems described in Section 1.2, above, may alleviate these problems to some degree.

In 1997 Gershon and Eick stated the challenge:

> "With the widespread deployment of 3D graphics chip sets, desktop PCs will soon handle much more sophisticated 3D graphics and animations. The challenge is how best to exploit this forthcoming capability. Currently, we do not always understand when 3D is more effective than 2D and vice versa. As better software makes it easier to produce visualizations, it will be important not to use these new capabilities indiscriminately — only when they are appropriate and convey information effectively." — Gershon and Eick [77, Page 30].

This thesis explores situations where a compromise between 2D and 3D visualisations is reached: that is, visualisation paradigms which exploit 3D graphics whilst, at the same time, retaining the



(a) The original figure produced by Minard in 1861 [160].

(b) A modern reworking of the chart uses a spatial mapping for time. Courtesy Menno-Jan Kraak [120]

Figure 1.3: Two versions of Minard's famous visualisation of the 1812–13 Napoleonic campaign through Russia.

useful properties of visualisations in 2D. Although the term is defined more precisely in later chapters, such a compromise can be thought of loosely as a "two-and-a-half dimensional" approach.

1.4 Application domains

This section briefly introduces the three application domains explored at length in the case studies of later chapters. Particular focus has been given to previous efforts to visualise these domains in 3D.

1.4.1 Fund Manager Portfolios

The data-set instigating the first case study (Chapter 5) is holding-data from the United Kingdom (UK) Stock Exchange Registry[2]. *Issuers* are the companies that float their stocks on the stock market allowing investors, or *holders*, to buy a share of the company. Under UK law issuers must regularly declare to the Stock Exchange Registry a list of their largest holders. Industry analysts associated with this case study[3] were interested in using this data to try to reconstruct the portfolios of large fund managers and, then, to visualise these portfolios in such a way that the investment

[2]Provided by Computershare Analytics (UK) Ltd.

[3]The case study was funded by the Cooperative Research Centre for Technology Enabled Capital Markets (CMCRC Australia Ltd.)

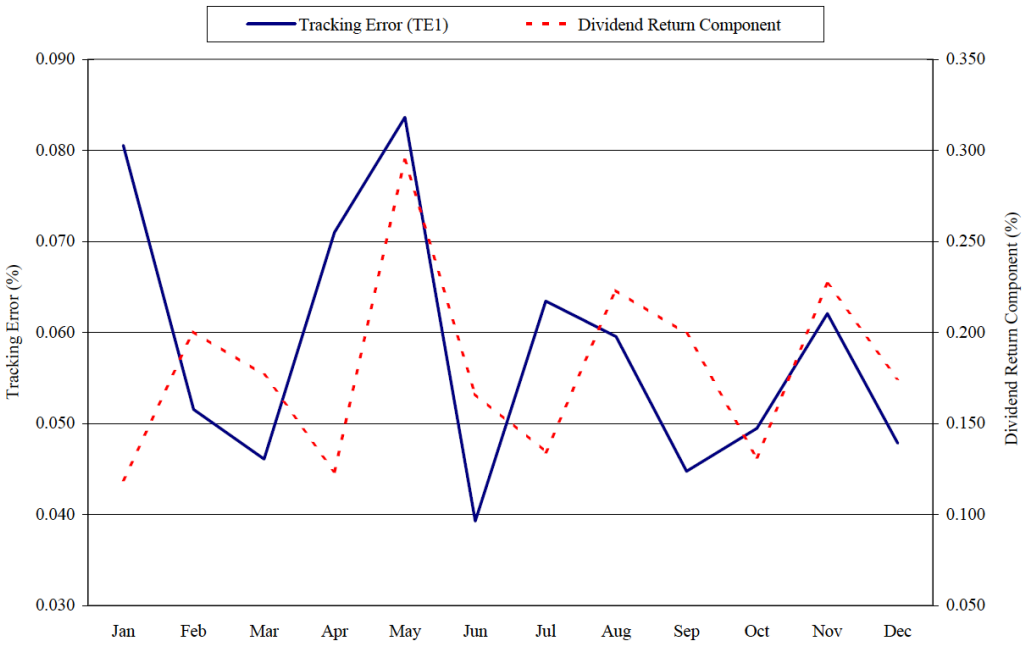


Figure 1.4: A time-series chart used in analysing a mutual fund's performance. Courtesy David Gallagher.

strategies of the fund managers could be studied.

A fund manager maintains a portfolio of stocks spread across a number of industries or market sectors with the aim of maintaining or growing the portfolio's value. A typical example of the type of visualisation familiar to fund-manager performance analysts is shown in Figure 1.4. As described by Gallagher [74], the chart offers an explanation for seasonal variations in tracking error: that is, the difference in performance between a fund and the market index based on the effects of dividend returns. The figure is very effective at communicating a specific point that is already well known to the author: that is, it is a visual explanation. It does, however, show only a simple relationship involving only a very small number of variables.

From the point of view of visualisation research, the charting techniques used by fund-manager-performance analysts are not at all ambitious. In fact, these visual techniques were demonstrated by Playfair in the 18th century [191]. More modern techniques can be used to open up new possibilities in visual data mining and exploration for these analysts. In Chapter 5 novel visualisation methods are used in an application for exploring large-volume, high-dimensional fund-manager portfolio data.

A survey of the techniques available for analysing portfolio data should also include more general capital market visualisations. To date, most systems for the visualisation of stock market data

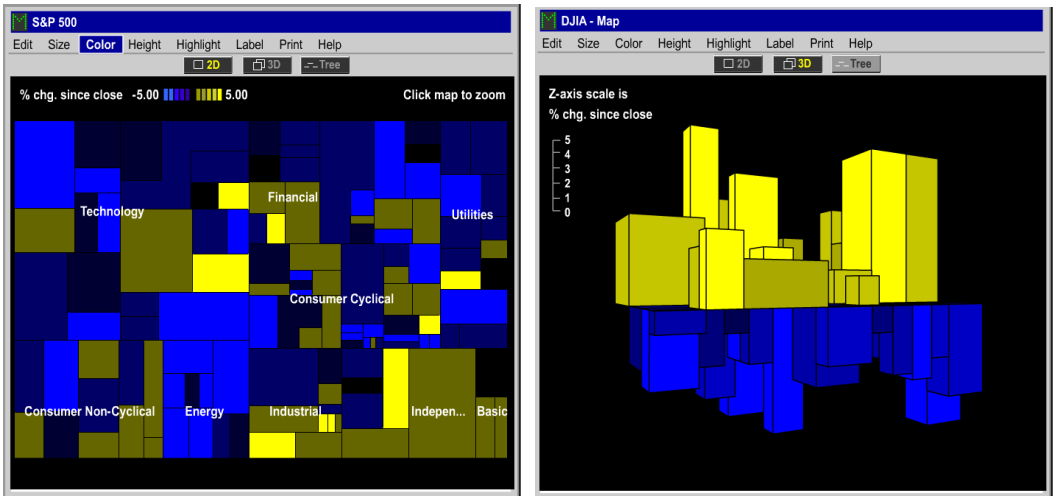
Figure 1.5: A typical share price and volume time-series chart. Courtesy Commonwealth Securites.

have concentrated on effective methods for showing time-series data, particularly trying to capture share-price fluctuations and market share. This has traditionally been achieved through simple charts and histograms: for example, Figure 1.5 is a typical chart showing fluctuations in share price and trading volume over time[4]. A relatively early example is from Varshney and Kaufman [190] who present a tool for exploring financial data, such as stock and option prices, in a visual-spreadsheet-type format. The only 3D visualisation included in their tool is a 3D histogram demonstrated with stock bid prices. Another early report on the visualisation of capital markets data using 3D graphics is demonstrated by Wright [203]. Again, these are essentially 3D histograms but are notable as Wright also stresses the importance of animation and interaction in supporting users' understanding of a 3D scene.

In studying ways to extend visualisations of foreign currency exchange options into three dimensions, Gresh et al. [85] showed how these multidimensional entities could be displayed as surfaces in 3D space. Although these are at the cutting edge in business graphics, such techniques have been used in engineering and scientific visualisation for many years.

Tegarden [184] gives a comprehensive overview of applications of state-of-the-art visualisation techniques to business data. Included is a dense representation of stock price and volume data using a "floor and walls" metaphor. This involves the use of 3D histograms displayed on the floor of a virtual room with additional 2D charts mapped onto the walls of the room. Again, although visually quite arresting, the techniques used are fairly trivial extensions of common chart types into a 3D space.

[4] See http://www.comsec.com.au.



(a) In 2D, colour intensity is used to indicate changing share price.

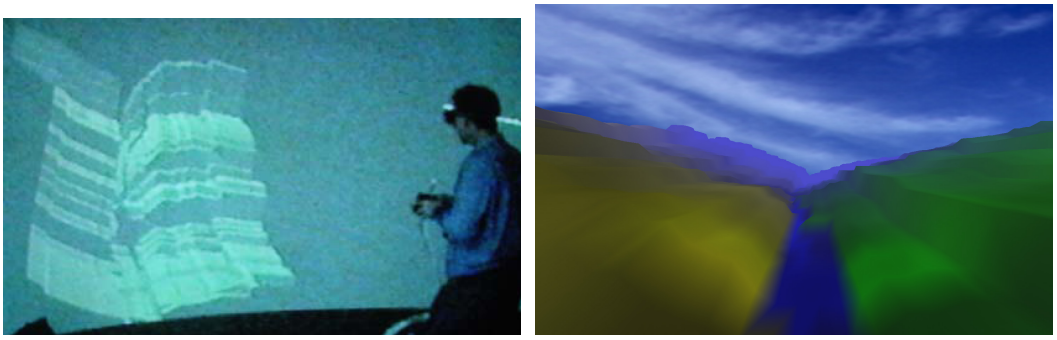
(b) In 3D, height replaces colour.

Figure 1.6: The SmartMoney treemap based portfolio representation.

One innovative technique which deviates somewhat from the time-series chart type, is the SmartMoney market map [175], see Figure 1.6. Based on the TREEMAP system by Shneiderman [173], the market map divides a 2D area into smaller rectangular regions representing industries. These industry rectangles are divided again into still-smaller rectangles representing individual stocks. The area of each region corresponds to the total value of the sector or stock, and colour is used to indicate change over time. A 3D extension extrudes each rectangle into a box where height, above or below the reference plane, instead of colour, is used to indicate change in value. Since they use relative area to represent a percentage, TREEMAPs could be thought of as descendants of pie charts. However, they improve on readability over pie charts and provide a capacity for representing hierarchies — such as the industry/stock hierarchy of this application.

A more-abstract metaphor is explored by Nesbitt [146]. The Bid-Ask Landscape (see Figure 1.7) uses a virtual-reality environment involving auralisation to display a dynamically updated representation of the order book for a particular share. The order book is presented as a valley and a river with volumes of bids (buy orders) indicated by the height of the left bank and asks (sell orders) represented similarly by the right bank. The position of the river indicates the price at which an actual trade occurs. He reports that experiments testing the effectiveness of the visualisation in helping users to predict price changes gave very positive results.

Some more-recent methods for visually analysing financial data have utilised graph-visualisation



(a) The Bid-Ask Landscape visualisation in a virtual reality environment. Courtesy Keith Nesbitt.

(b) A literal interpretation of the Bid-Ask Landscape metaphor. Courtesy Damian Merrick.

Figure 1.7: Two versions of the Bid-Ask Landscape metaphor

techniques to show the relationships between more-abstract concepts. The term *graph* is intended here, and in the sequel, in the mathematical sense of a 'network of connected concepts'[5] as distinct from the charts and histograms described above. The most common examples have tried to capture share price correlation based on various similarity metrics by introducing extra geometric parameters. For example, Brodbeck et al. [25] use such metrics to create a large graph which is then arranged in the plane using a force-directed placement algorithm (see Section 3.2.1). In this way, more highly correlated commodities are positioned geometrically closely together so that clustering is clearly visible. Gross et al. [86] use a similar technique to compare economic data using three-dimensional layout.

1.4.2 Metabolic Pathways

Biochemistry is the study of chemical substances and chemical processes occurring in living beings, and is becoming increasingly important to innumerable applications — from the development of pharmaceuticals to genetic engineering. In biochemistry, biological processes are often represented as complex networks such as metabolic pathways [137], signal transduction pathways [168], and protein-protein interaction networks [132].

The second case study considered in this thesis (Chapter 6) involves the visualisation of metabolic pathways to support biochemists' understanding of these complex networks. *Metabolic reactions* are transformations of chemical substances occurring in living organisms. A reaction involves the transformation of one or more chemical substances (*reactants*) into one or more different chemical

[5] Sometimes called a relational network.

substances (*products*) and this transformation is usually assisted by a catalyst called an *enzyme*. Specific metabolic processes occurring in living beings involve many such reactions forming large and often complex networks, where each node in the network represents a chemical substance and the links between them indicate a reaction transforming one substance into another. The networks representing specific processes, or those defined by functional boundaries such as the network between particular initial and final substances, are called *metabolic pathways*. Metabolic pathways are subnetworks of what biochemists think of as the much larger, complete network of metabolic reactions.

The most common visual representations of metabolic pathways are static drawings of these networks on wall charts or in textbooks, such as [137]. Attempts to render all the known metabolic pathways — or even just the most commonly studied pathways — in a single reference chart have yielded very large and complex posters. For example, Figure 1.8 shows a poster by Michal [136] of important metabolic pathways as they were understood in 1993. Clearly, such manually produced, static representations have a number of drawbacks:

- Even printed as a large poster, the detail is only visible by very close inspection.
- It is very difficult to update such a drawing as new information becomes available. Since 1993 many new pathways have been identified and biochemists' understanding of existing processes have been updated.
- Finding subnetworks for specific metabolic processes requires a tedious scan across the entire poster.

Efforts to create systems with pre-drawn pathway diagrams, cross-referenced and stored in online, searchable databases represent a significant advance on static posters and books. The first such attempt, ExPASy [4] is an annotated image database made up of scanned segments of the Biochemical Pathways poster. The *KEGG: Kyoto Encyclopedia of Genes and Genomes* [110] is a similar system, but it contains separate diagrams (again manually produced) for pathways of different processes (see Figure 1.9). Another — although much less extensive — online collection of pathway diagrams, the *Metabolic Pathways of Biochemistry* website[6], is notable in this survey for its limited use of 3D graphics. It uses the MDL CHIME plug-in[7] to draw 3D molecular representations of the substances *in-situ* in the diagram (see Figure 1.10).

[6] See http://www.gwu.edu/~mpb/.
[7] See http://www.mdli.com/products/framework/chemscape/.

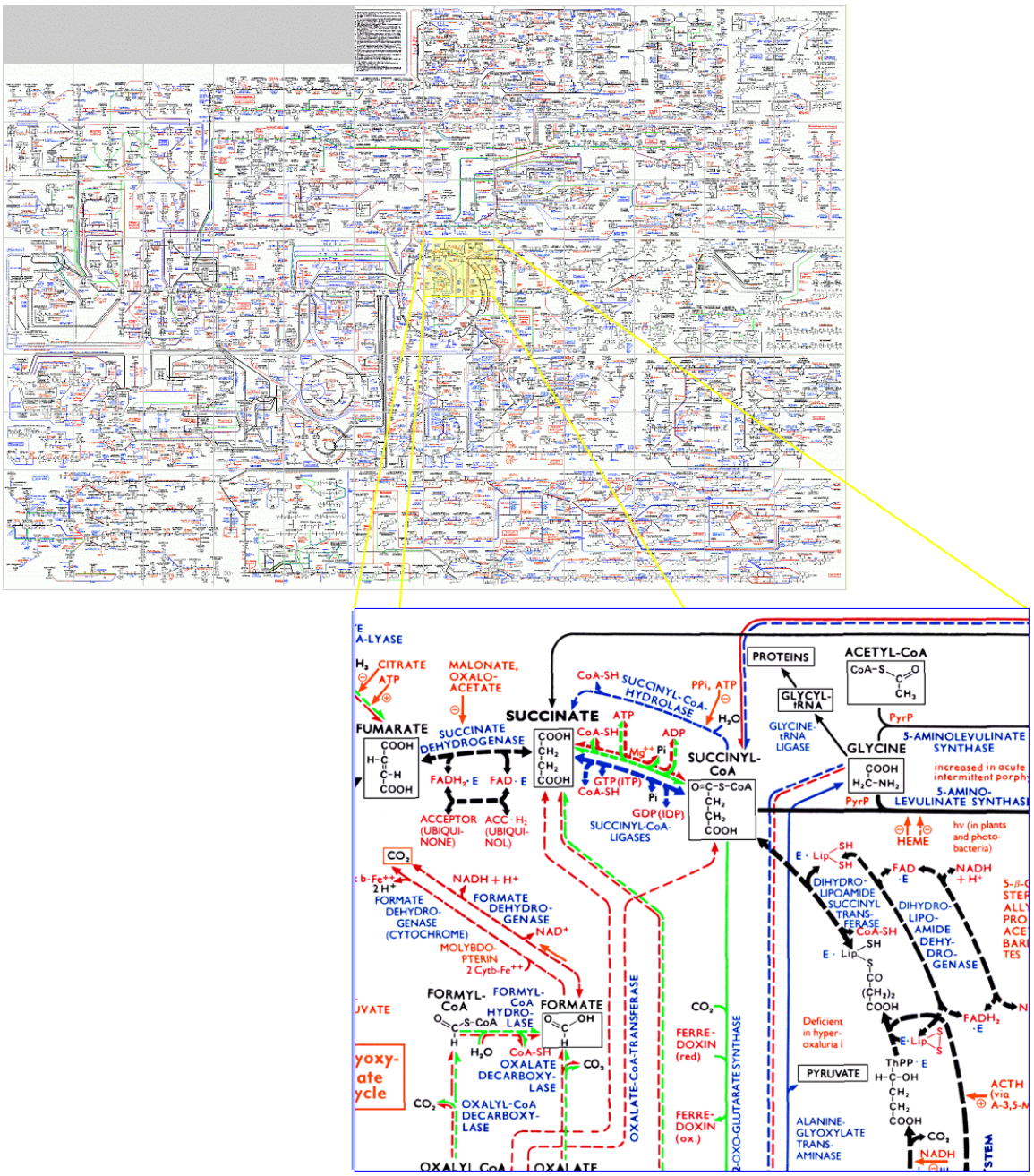


Figure 1.8: The metabolic pathways part of the "Biochemical Pathways Poster" by Michal [136] with detail shown in inset.

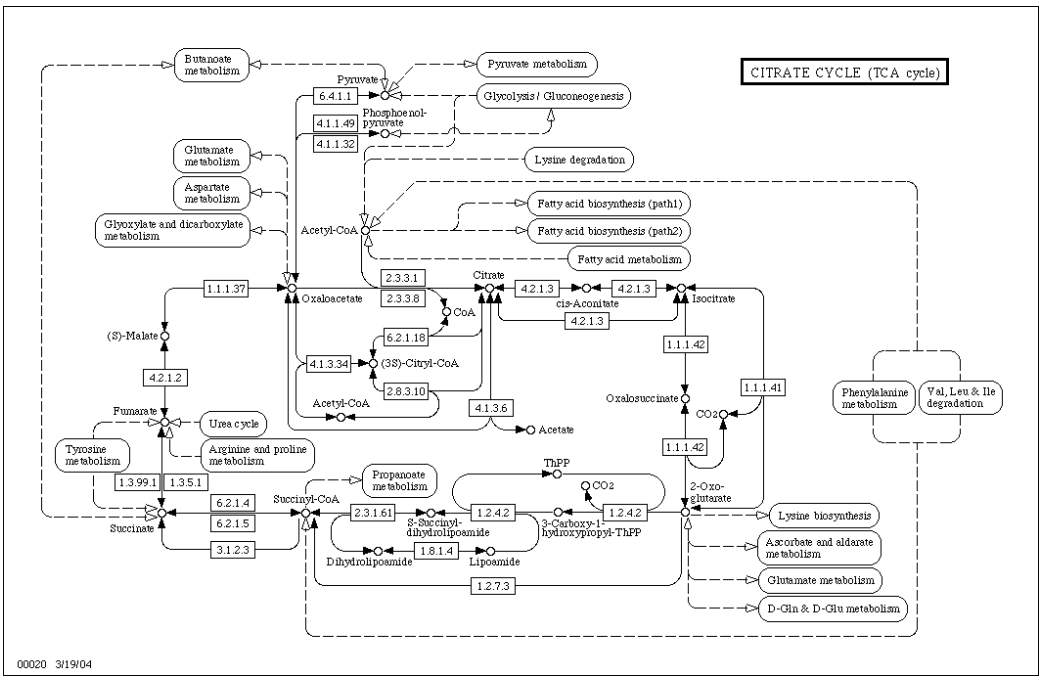


Figure 1.9: A representation of the Citrate Cycle from the KEGG database.

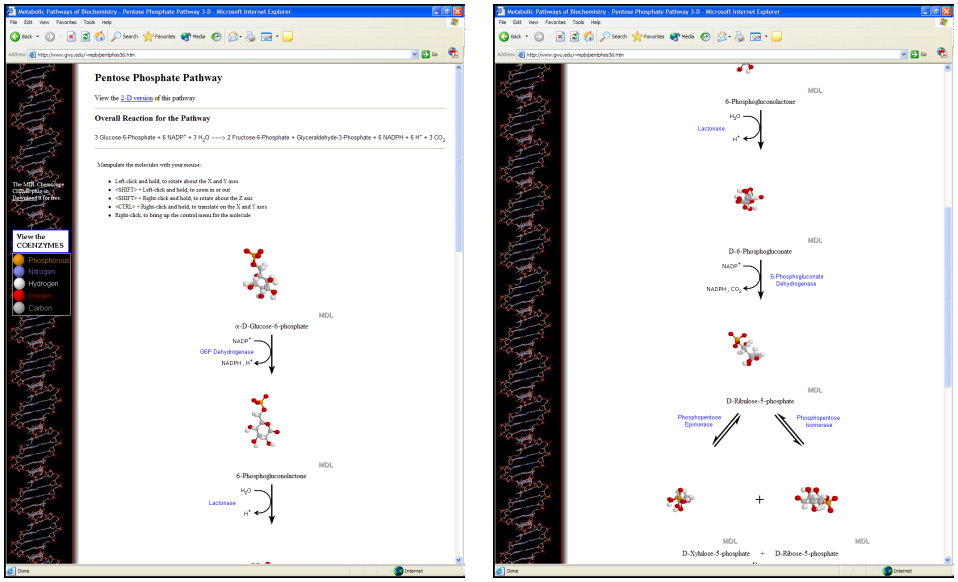


Figure 1.10: Screen-shots of the “Metabolic Pathways of Biochemistry” website showing 3D molecular representation of substances using the MDL CHIME plug-in.

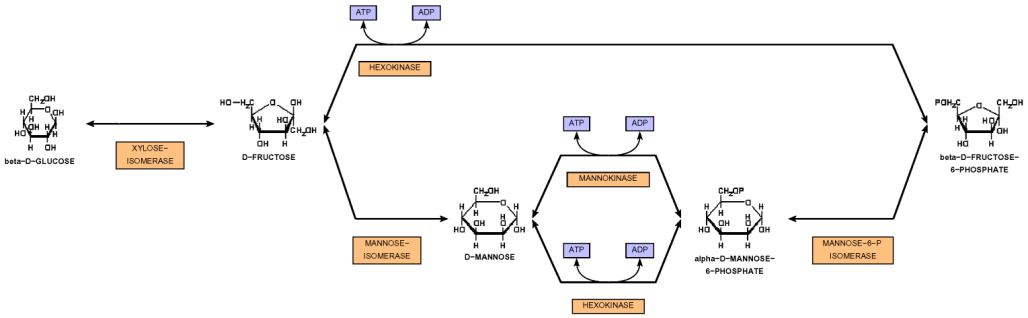


Figure 1.11: A metabolic pathway automatically drawn by BIOPATH.

The images stored in each of these on-line databases are static: that is, each diagram is manually produced and cross-referenced in the database. Although this addresses the search-ability issue, the time consuming process of creating new diagrams as new knowledge becomes available, means that databases such as KEGG are difficult to keep up to date. The solution to this problem lies in generating the diagrams automatically using graph-drawing techniques. The current state of the art in this endeavour is the BIOPATH system [69, 169]. Figure 1.11 shows a high quality metabolic-pathway drawing automatically generated by BIOPATH.

1.4.3 Phylogenetic Trees

The final case study in Chapter 7 involves the use of 3D graphics to study *Phylogenetic trees*. Phylogenetic trees (also called *evolutionary trees*) show the ancestral relationships among a set of species. The leaves of these trees represent species, the internal nodes refer to (hypothetical) ancestors. Phylogenetic trees play an important role in life sciences and are used for many applications, such as understanding evolutionary processes, designing new drugs, measuring genetic variations between species, and predicting the expression of genes.

The trees are inferred from data, describing similarities between species, in a process called *phylogenetic analysis*. These similarities may be determined by expert knowledge about the morphological attributes (physical characteristics) of species or, more recently, by automated methods such as the comparison of nucleotide and protein sequences [61, 67, 1]. There are many algorithms available for tree inference from this similarity data. Generally, most methods recursively match pairs of species, and then families of species, to produce a binary hierarchy.

An example of a conventional drawing of a phylogenetic tree is given in Figure 1.12[8]. Note that,

[8]Data-set from [39]. Used by permission.

by convention, branch lengths give an indication of similarity of the species (and therefore evolutionary time). Note also, that the leaves of the trees are usually drawn with no specific order except to keep the drawing planar[9]. In some applications it makes sense to order the leaves such that the total dissimilarity of consecutive leaves is kept to a minimum, while respecting the neighbourhood constraints defined by the tree such that the tree-drawing remains planar. Such orderings are known as *optimal leaf orderings* and can be computed efficiently for any similarity measure [8, 140].

A number of simple systems are available for producing these standard 2D tree drawings, also called *phenograms*, *dendrograms* or *cladograms*. A well known example system is TREEVIEW [150]. Producing these drawings is relatively straightforward and a number of packages for inferring tree structure from species similarity data also generate their own cladograms, for example PHYLIP [61] and T-REX [127]. More recently the focus of phylogenetic tree visualisation systems has shifted to providing interactive visualisations of very large trees, for example, the TREEWIZ system [164] allows a user to browse 2D drawings of tree structures with tens of thousands of leaves.

The availability of very large phylogenetic tree data-sets and biologists' need to study them, has also helped to motivate the use of 3D graphics in order to show as much of the tree structure as possible in a single display. Figure 1.13 shows a visualisation, prepared by Hughes et al. [103], of the most complete phylogeny of all known cellular organisms available[10]. The display was generated by the WALRUS 3D hyperbolic graph visualisation tool[11] and contains approximately 180,000 nodes. A user can interactively "fly" around the 3D space or zoom-in to sub-trees by choosing focus nodes in the hyperbolic browser.

Another recent thrust in phylogenetic tree visualisation research has focused on the comparison of sets of similar trees. The TREEJUXTAPOSER [142] system allows users to compare two trees using interactive zooming. A system from Klingner et al. [117] allows users to browse a larger set of trees using a *multi-dimensional scaling*[12] technique. An effort to utilise 3D graphics in tackling this visualisation problem comes from Stewart et al. [179]. In this visualisation[13] (see Figure 1.14) a number of traditional 2D arrangements of similar trees are stacked into the third dimension to support comparison of the trees' internal structure. This is an example of what is defined in Chapter

[9] A planar drawing of a tree has no overlapping branches.

[10] From the NCBI Taxonomy database. See http://www.ncbi.nlm.nih.gov/Taxonomy/.

[11] See http://www.caida.org/tools/visualization/walrus/.

[12] Multi-dimensional Scaling (MDS) is the process of reducing the dimensionality of data: for example, by combining correlated variables. MDS is explained in more detail in Chapter 5.

[13] Visualisation produced by TREEVIEWER. See http://www.avl.iu.edu/projects/Tree3D/.

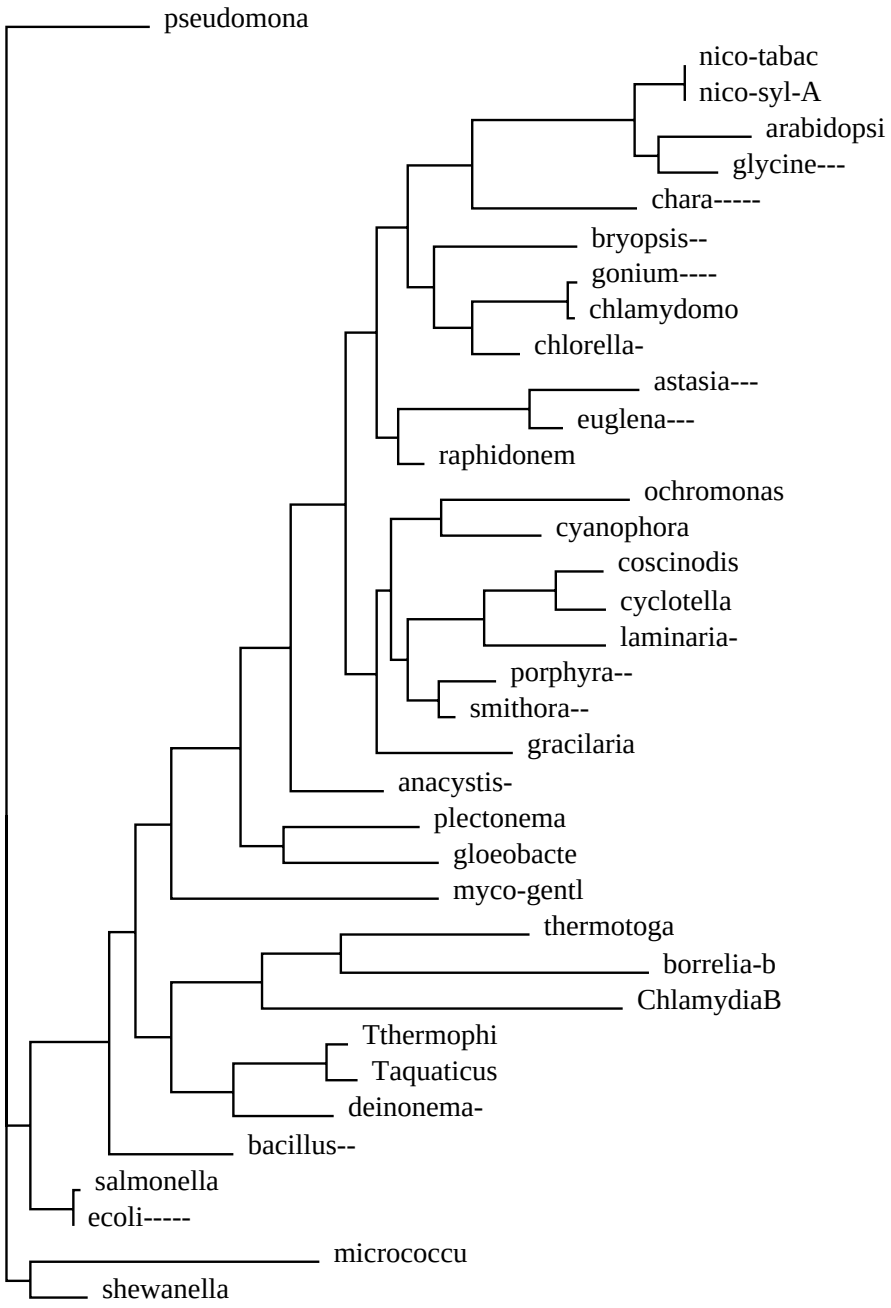


Figure 1.12: An example of a conventional drawing of a phylogenetic tree (a phenogram). The leaves indicate species; the internal nodes represent hypothetical ancestors for the species; and branch lengths give an indication of evolutionary time. The leaf ordering is entirely arbitrary.

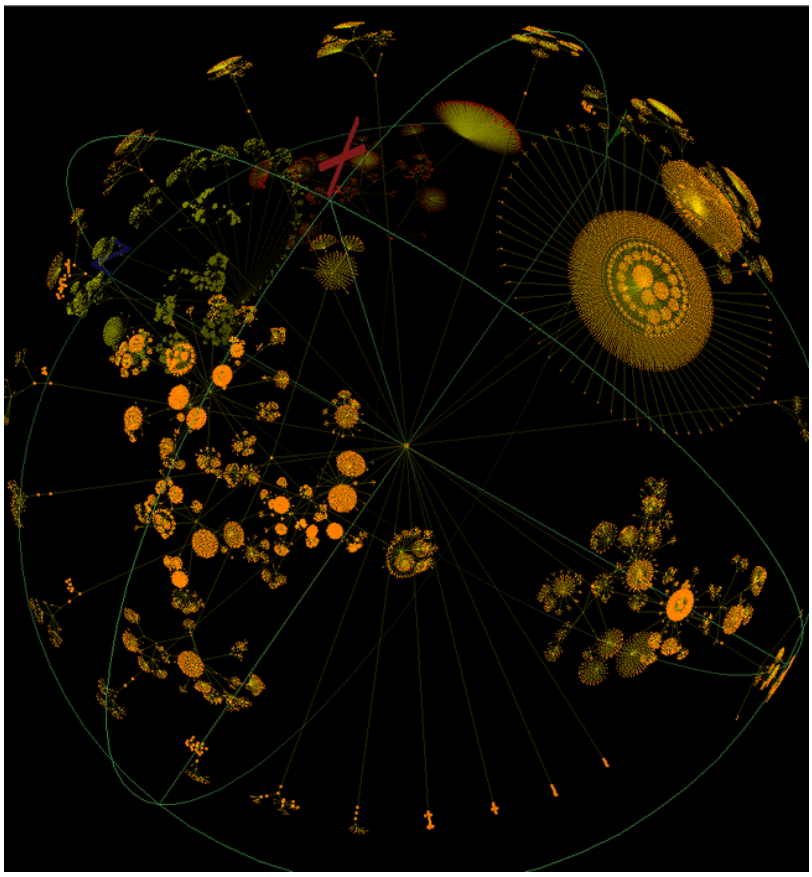


Figure 1.13: A 3D Hypobolic drawing of a very large phylogenetic tree generated with the Walrus tool, courtesy Tim Hughes.

4 as "$2\frac{1}{2}$D stratified graph visualisation" and is discussed again in more detail in the case study in Chapter 7.

1.5 Research Methodology

In this thesis the early literature studies, research and experiments in creating visualisations for various domains led to the formulation of a number of principles for 3D visualisation design. Existing descriptions of so called "$2\frac{1}{2}$D" visualisation were refined to produce a more precise definition of $2\frac{1}{2}$D visualisation and $2\frac{1}{2}$D network (or graph) visualisation. The $2\frac{1}{2}$D visualisation paradigm was explored using a quantitative experimental study. The experimental method used is a standard two-group, post-test-only, randomised experimental design (see [101]). Subjects' performances in various tasks testing their ability to understand a visualisation were analysed.

A number of 3D visualisation tools implementing these $2\frac{1}{2}$D principles were designed. The

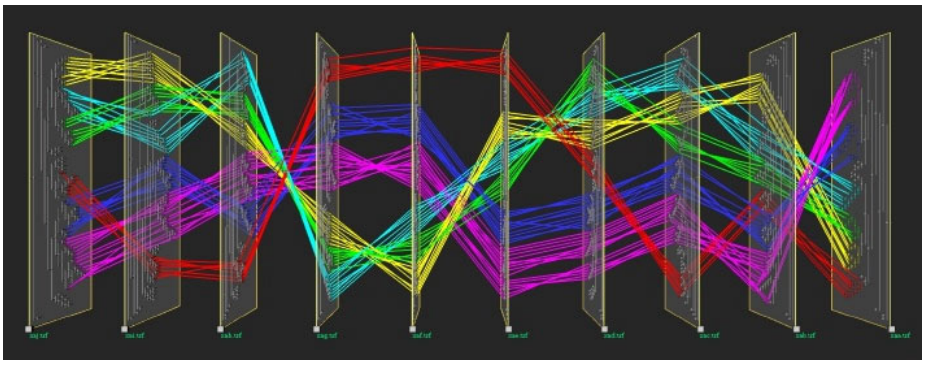


Figure 1.14: The $2\frac{1}{2}$D visualisation of a set of phylogenetic trees proposed by Stewart et al. . Note the large number of crossings between the coloured edges joining similar species in adjacent trees. Used by permission.

evolutionary process that was followed for designing, implementing, evaluating and refining these tools could be considered analogous to the software engineering process model of iterative prototyping [83]. Tools and algorithms were initially designed based on broad requirements gathered by researching the application domains and by seeking suggestions from experts in these domains. Prototype implementations of algorithms were evaluated both analytically and with experiments on a number of sample inputs. Results were tabulated and reviewed. Prototype tools were implemented to test the applicability of $2\frac{1}{2}$D visualisation principles in three different application domains. These were then evaluated, as described below, and refined based on the results of this evaluation.

Initially, the prototypes were little more than technology demonstrations, featuring few interactive features and applicable only to specific data-sets. However, some implementations, particularly the fund-manager portfolio visualisation system, described in Chapter 5, became quite elaborate and functional, although still of prototype quality.

The prototype visualisation systems developed for our various case studies were evaluated in interviews with domain experts. The interview design was based around an informal *cognitive walk-through* [178, 199] style methodology. That is, guided walk-throughs of use-cases of the various visualisation systems were conducted with domain experts and their feedback was gathered. Originally, cognitive walk-through methodology was conceived for evaluating user interfaces for software developed according to a clear set of requirements. The walk-through would step through use-cases designed as part of the requirements-gathering process and the experts would be asked to evaluate the system according to a narrow set of usability criteria. For example (from Wharton et al. [199]):

- Will the user try to achieve the right effect?

- Will the user notice that the correct action is available?
- Will the user associate the correct action with the effect sought?
- If the correct action is performed, will the user see that progress is being made toward the solution of the task?

When evaluating the *potential* of novel visualisation techniques, however, the above questions were thought to be too focused on implementation specifics. That is, such low-level questions are not constructive when an evaluation of the visualisation paradigm, rather than an evaluation of the details of the user interface, is needed. Therefore the interviews were permitted to flow freely in order to capture as much feedback from the experts as possible.

Such a qualitative evaluation is the most reasonable approach because of the high-level nature of the task and a lack of similarly ambitious tools with which to compare. As well as providing a feel for the potential and utility of the visualisation paradigm this qualitative evaluation also provided feedback enabling improvements to the systems' usability to be made. Transcriptions of typical interviews are given in Appendix A.

1.6 Contributions

First, a formal definition for $2\frac{1}{2}$D visualisation is given and its advantages and disadvantages are investigated through a quantitative study.

Secondly, this $2\frac{1}{2}$D visualisation definition is refined specifically for the visualisation of graphs. A $2\frac{1}{2}$D "stratified" graph-visualisation paradigm is introduced for applications involving evolving graphs and comparing sets of related graphs.

Geometrically arranging these stratified graphs so that their structure is clearly visible is a graph visualisation problem. This thesis shows that the most popular 2D graph-visualisation algorithms, based on force-directed layout and hierarchical layout, can be applied to stratified graphs with acceptable results. Further, modifications to these algorithms are introduced that make better use of the third dimension in stratified graph visualisation.

The above techniques are evaluated with respect to three important application domains in a series of case studies. Specifically, the first case study provides two novel, and complementary, $2\frac{1}{2}$D methods for visualising movement in fund-manager portfolios. The second case study introduces novel $2\frac{1}{2}$D graph visualisation techniques in three metabolic pathway visualisation applications.

The final case study introduces an algorithm for improving the readability of a $2\frac{1}{2}$D phylogenetic tree visualisation.

1.7 Organisation of this Thesis

This thesis is structured in such a manner that all research results are reported in chapters 2 and 4–7 while the bulk of the background material is given in chapters 1 and 3. However, in order to preserve the logical progression of the definitions, some background material specific to certain topics is presented closer to the research results where it is most relevant.

This introductory chapter has stated the motivation for the research and has provided some general background into 3D visualisation — especially in terms of the three application domains explored in later case studies.

Some general definitions that are used throughout the thesis are found in Chapter 2, together with further background and some experimental research supporting those definitions. More specifically, *$2\frac{1}{2}$D visualisation* is defined and the history of the term "$2\frac{1}{2}$D" in various application domains is explored. A report on a quantitative experiment exploring the utility of the $2\frac{1}{2}$D-visualisation guidelines is also presented.

Chapter 3 provides an introduction to the field of *graph visualisation*: that is, visualisation of relational networks. The most popular 3D graph-visualisation methods, both those that are still only of interest to researchers and those that are actually used in practical applications, are surveyed. When introducing these graph visualisation methods they are appraised in terms of the $2\frac{1}{2}$D design principles introduced in Chapter 2, particularly concentrating on the 3D graph-visualisation techniques that are extended in Chapter 4.

In Chapter 4 *$2\frac{1}{2}$D graph visualisation* is defined and a number of graph visualisation methods and applications that broadly fit this definition are identified. Then, a more specific class of $2\frac{1}{2}$D graph visualisation — *stratified graph visualisation* — is introduced and its applicability to visualising evolutionary graphs and comparing sets of related graphs is discussed. Some algorithmic considerations in arranging these graphs are also discussed.

These concepts of $2\frac{1}{2}$D visualisation are investigated in the light of three application domains. The first of these, in Chapter 5, is capital markets visualisation, for which two complementary prototype visualisation systems for overview and detailed study of fund managers' portfolio holdings are demonstrated. In Chapter 6 the application of several $2\frac{1}{2}$D stratified graph visualisation tech-

niques in bio-informatics are shown, specifically in the visualisation of metabolic pathways. The third case study, in Chapter 7, investigates another bio-informatics application of $2\frac{1}{2}$D stratified graph visualisation — a method for comparing sets of phylogenetic trees. Rather than a system demonstration, this last case study presents an algorithm that seeks an optimal arrangement of a set of trees for stratified graph visualisation.

Finally, in Chapter 8 conclusions are set out together with some directions for future research.

CHAPTER 2

Two-and-a-half Dimensional Visualisation

> "All communication between the readers of an image and the makers of an image must now take place on a two-dimensional surface. *Escaping this flatland is the essential task of envisioning information – for all the interesting worlds (physical, biological, imaginary, human) that we seek to understand are inevitably and happily multivariate in nature. Not flatlands.*" — Edward R. Tufte [186, p. 12]

Chapter 1 provides a number of examples of supposedly 3D visualisations which do not achieve any practical advantage over 2D counterparts. This chapter explores *$2\frac{1}{2}$D design principles* — a way of utilising 3D graphics for information visualisation that is well suited to human spatial perception. It is argued that visualisations based on $2\frac{1}{2}$D design principles provide concrete benefits over comparable 2D visualisations.

More specifically, Section 2.1 introduces the concept of *$2\frac{1}{2}$D visualisation* as 3D visualisation based on $2\frac{1}{2}$D design principles. A general background for the term is given, as well as the more precise definition that is used throughout this thesis. Section 2.3 provides a simple example of application of these design principles to extend a famous 2D visualisation into the third dimension. The effectiveness of this redesign is evaluated by an experimental study which is detailed in Section 2.4.

2.1 Background

The term "$2\frac{1}{2}$D" is used inconsistently in various types of literature.

This section surveys these uses and provides a more concrete definition for the purpose of this exposition. The term "$2\frac{1}{2}$D" was introduced into psychology by Marr [129, 130] to describe the imperfect human cognitive process of spatial perception. Humans detect edges around shapes and then infer surfaces inside the edges based on depth perception. The result is that the human ability to resolve depths through stereo and motion viewing is much more limited than our ability to resolve horizontal and vertical distances.

Ware [194] outlines some guidelines for effective use of 3D visualisation that take into account the limitations of human depth perception. He calls the application of these guidelines a "$2\frac{1}{2}$D Design Attitude". The four guidelines Ware proposes are:

WG1 *Use 3D Objects to represent data entities.* He supplies some evidence from cognitive psychology and his own experiments suggesting that renderings of 3D objects provide more recognisable *glyphs*[1] in an information visualisation than 2D symbols.

WG2 *Emphasise 2D Layout.* He claims that important "object structures" should be arranged in the plane. He does not strictly define the term but we can assume that, in visualisation, it refers to more complex glyphs and also to the focus of this thesis: relational networks or graphs.

WG3 *Use depth cues selectively as needed.* Here he suggests that there are situations where full blown 3D perspective rendering is not necessary to achieve some of the advantages of 3D display. Simpler depth cues, such as occlusion, may be adequate for applications which allow users to navigate through an information space too large to fit into a single screen.

WG4 *Use 2D layout to support navigation in 3D spaces.* This point relates to the coupling of 2D navigational aides such as overview maps or cross-section viewers to the 3D view.

Ware's use of the term $2\frac{1}{2}$D is clearly, though informally, defined. Ware's guidelines are referred to frequently in the sequel. However, the term also often arises in the literature in different contexts and with ambiguous intent. Surveying a few such uses helps to motivate the more formal definition offered in Section 2.2 for $2\frac{1}{2}$D in information visualisation.

Although it is difficult to know when the term was first applied to information visualisation, the occasional references to $2\frac{1}{2}$D in the literature are usually used to describe a *pseudo-3D* visualisation. That is, not a true 3D space, but perhaps a rendering making the scene look 3D. For example, in an evaluation of the effectiveness of spatial memory in 2D and 3D environments, Cockburn and McKenzie [35] test a *data mountain* style document retrieval interface [158] which they call "$2\frac{1}{2}$D".

(a) “2D” (b) “$2\frac{1}{2}$D” (c) “3D”

Figure 2.1: Three different styles of Data Mountain visualisation by Cockburn and McKenzie. The “2D” display has occlusion, the supposedly “$2\frac{1}{2}$D” display only allows documents to be positioned on a plane.

In the $2\frac{1}{2}$D data mountain (see Figure 2.1(b)), a plane, tilted away from the viewer, is rendered in perspective. Users are able to move thumbnail images representing documents — in this case web pages — forward and back, or left and right on the surface of the plane. The authors compare this view with a “2D” view (see Figure 2.1(a)) and a true 3D view (see Figure 2.1(c)). The 2D view shows an orthographic, top-down view of the plane, similar to a Windows desktop, in that it supports occlusion between overlapping thumbnails. In the “3D” view, users are able to move the thumbnails with three degrees of freedom.

The use of the term “$2\frac{1}{2}$D” here seems rather arbitrary, since their “2D” visualisation also exhibits the 3D property of occlusion and the “$2\frac{1}{2}$D” visualisation is simply a perspective rendering of a 2D coordinate system. Their “$2\frac{1}{2}$D” visualisation does not display any of Ware’s $2\frac{1}{2}$D traits and the real difference between the “2D” and the “$2\frac{1}{2}$D” interfaces is perspective rendering. Not surprisingly, their results showed little difference in performance between users of these two visualisations.

Though it predates Ware’s article, a 3D graph visualisation system by Dodson [44] shows a strongly $2\frac{1}{2}$D design by making extensive use of depth cues (**WG3**) and restricting the layout of the graphs to 2D planes (**WG2**). He uses the term $2\frac{3}{4}$D to describe his use of 3D space, but regardless of the 10% dimensional inflation in his casually adopted catchphrase, the intention seems the same. This visualisation is discussed in more detail in Section 3.2.1.

The term also appears in a number of other domains, although with differing meanings. In Computer Aided Design, essentially 2D shapes or designs which have been extruded into the third dimension, have long been described as $2\frac{1}{2}$D. For example, in manufacturing and mechanical en-

[1] In information visualisation *glyph* is the generic term used to describe either a 2D symbol or 3D object representing a data entity.

gineering many simple engine parts, such as cogs, can be described in this way [105]. In the same way, an architect can easily convert a 2D floor plan into a simple $2\frac{1}{2}$D model as a method of rapid prototyping — allowing allowing a rudimentary virtual walk-through. The term may have been made more popular in 1986 when, by coincidence, the first version of AutoCAD to provide this functionality was released with a version number of 2.5 [201].

Although he does not explicitly use the term, Brooks [26] demonstrates an understanding of the $2\frac{1}{2}$D design attitude in designing scientific visualisations. He advocates the use of depth cues (**WG3**) and declares:

> "Perspective is very effective when parallel lines and right angles abound in the model"

The term does appear in scientific visualisation publications, but again, it is difficult to find any formal definition. Typically, visualisations in which data is mapped onto a 2D surface (or manifold) are known as $2\frac{1}{2}$D. For example, in geographical information systems, visualising some attribute by mapping it to an artificial height dimension, is a common practice [80].

The ubiquitous windowing desktop environments, familiar to personal computer users, could be thought of as $2\frac{1}{2}$D in that a stacking into the third dimension is implied by occlusion, a clear application of **WG3**. There are a number of efforts to create a more complete 3D desktop interface. For example, the LOOKING GLASS system from Sun [138] takes advantage of the increased graphics power of modern desktop systems to allow users to browse open windows by viewing the stack from the side. However, LOOKING GLASS still observes $2\frac{1}{2}$D design constraints, such as **WG3** and **WG4**, by maintaining a default orientation of the stack and a uniform spacing between layers in the stack. That is, the third dimension is still utilised in a limited way to support users' understanding of the virtual space.

The term "$2\frac{1}{2}$D" has also been used to describe simplified projections from 3D models to a 2D surface [121]. For example, a distinctive feature of classic Chinese and Japanese painting is isometric or axonometric projection. Axonometric drawings have no vanishing point and parallel lines in the 3D model are also drawn in parallel. The viewpoint is typically a birds'-eye view looking down at the scene at an angle of about 30 degrees. Unlike pre-Renaissance Western art, in which perspective distortions were most likely due to a poor understanding of optics, the Chinese used axonometric projections[2] quite deliberately. It allowed them to paint long story telling scenes

[2] Such projections are referred to in Mandarin as *dengjiao toushi*, which translates as: "equal-angle see-through" [121].

on scrolls. The viewer could unroll the scroll to follow the story with a continuous 3D effect. Additionally, observers such as Bragdon [15], hail axonometric projection as a simplification of a 3D scene that more closely corresponds to the observers' mental model of the environment. This $2\frac{1}{2}$D design attitude fits well with **WG3** and **WG4**.

Possibly, the most successful and commercially competitive applications of 3D computer graphics, are computer games. In the light of this competition, it is not surprising that game developers demonstrated an early understanding of the $2\frac{1}{2}$D design attitude. Most notably, the first widely acknowledged "first-person shooter" — WOLFENSTEIN 3D, released by ID software in 1991 [104] — provided the player with a convincing illusion of navigating a 3D world. However, internally the world model was only a 2D map and the player could move with only three degrees of freedom (2D location and orientation). This design achieved two things: first, it allowed the game engine to produce reasonable graphics performance on the limited hardware of the time; secondly, players could easily navigate the simple maps using only keyboard controls with minimal disorientation. Subsequent games became more sophisticated, gradually adding more degrees of freedom to player navigation.

In 1994 ID software released WOLFENSTEIN 3D's successor, called DOOM. The new graphics engine could render a terrain with an apparent 3^{rd} dimension featuring ramps and balconies. However, this was really just a rendering trick. Again the underlying map was two dimensional. The third generation of first-person shooter games was ushered in by QUAKE in 1996. QUAKE used a true 3D OPEN-GL [6] graphics engine that allowed players five degrees of freedom. As well as 2D location, players can look in all directions and control their position in the *z*-axis in a limited way. They can jump, crouch and walk directly underneath other players or monsters on bridges or elevated levels. QUAKE's first-person shooter successors have all followed this interaction model and have become the most popular gaming genre. By contrast, games such as DESCENT (another ID offering) which feature full six-degree of freedom control, never really reached a wide market.

Although games are meant to be recreational, the vast size and highly competitive nature of the gaming market makes it a potent software usability lab. New interaction features are regularly introduced and their success or failure is measured quantitatively in terms of revenue for the publishers. The lessons learnt in this domain therefore deserve to be taken seriously by the visualisation community.

2.2 Definitions

Such disparate uses of the term suggest that a much more precise notion, of exactly what "$2\frac{1}{2}$D information visualisation" means, is needed. In all of these domains, the term "$2\frac{1}{2}$D" is employed to indicate that the third spatial dimension is used in a fundamentally different — and also more restricted — way from the other two. In this section a class of visualisations that can be grouped informally under a "$2\frac{1}{2}$D" heading, is defined. The definition basically involves treating the 3^{rd} dimension (which, for convenience, is called the z-dimension) as a separate channel for conveying information. Some guidelines for restricting the mapping from the information space to the visual space for this channel, are also given. These guidelines fit well with Marr and Ware's caveats (see Section 2.1) for human spatial perception. The guidelines fall under three headings: *directness*, *independence* and *discreteness*.

2.2.1 A Separate Visual Channel

An information space S of k dimensions consists of k-tuples of attributes. That is, $S = A_1 \times A_2 \times \ldots \times A_k$ where A_i is the set of values for the i^{th} attribute. For example, in Chapter 5, a data-set is considered, in which fund managers' portfolios are weighted in stocks spread across a number of market sectors. That is, each tuple is a portfolio and the attributes are weightings across market sectors.

For general 3D information visualisation, given an information space of k dimensions, a visual mapping is sought, such that:

$$\sigma : \prod_{i=1}^{k} A_i \mapsto \mathbb{R}^3 \times \Theta \tag{2.1}$$

where $\mathbb{R}^3$ is the spatial mapping in the visual space and:

$$\Theta = \prod_{i=1}^{j} \theta_i$$

is the range of graphical attributes available, including colour, glyph shape, texture and so on. The set θ_i consists of values available for a particular graphical attribute (range of colours, number of shapes available, etc).

To restrict this broad definition to $2\frac{1}{2}$D information visualisation, two separate mappings f and

g from the information space to the visual space, are defined:

$$\begin{aligned} f &: A_j \mapsto \mathbb{R} \\ g &: \prod_{i \neq j} A_i \mapsto \mathbb{R}^2 \times \Theta \end{aligned} \tag{2.2}$$

Here the visual mapping σ is (f, g). The mapping f is solely spatial, while g is spatial and graphical. Effectively, the mappings f and g provide two separate channels for conveying information visually.

2.2.2 Constraining the z-Dimension

As discussed in Section 2.1, Marr noted that human depth perception was limited in resolution compared to the dimensions in the plane parallel to the optic array. Following this notion, **WG3** advocates careful use of depth cues. Therefore, the hypothesis postulated is that a mapping to 3D space that conforms to human spatial perception will make only limited use of the third dimension.

A 3D visualisation is most useful when the viewer is free to change the view-point relative to the scene, to peer around occlusions or to view the scene from a different perspective. Still, there must always be a default view, or starting eye/camera position, in any visualisation. The z-dimension or axis is therefore defined to be the axis parallel to the viewer's line of sight in the default view. That is, the depth axis. Therefore, f is referred to as a *depth mapping*.

Such a model also works well as a metaphor for physical environments and conforms to the ideas of Gibson [78] concerning ecological geographic spatial perception. Our interaction with the world around us is much more limited in the third "up/down" dimension than our ability to move around on the surface of a landscape. We live and work in buildings divided up into levels. It makes sense, therefore, that our mental mapping ability is better at coping with $2\frac{1}{2}$D environments than fully 3D spaces.

To summarise these observations of human spatial perception: we have two, not incompatible, arguments for limiting use of the third dimension in 3D visualisation. Marr's model of $2\frac{1}{2}$D vision gives us a case for limiting resolution of the depth axis relative to the viewpoint. Gibson's model of geographic spatial cognition suggests that human understanding of 3D space is limited regardless of direction of view. In this thesis we use both these models to argue for a restricted use of one of the three spatial dimensions (thus conforming to limited geographic spatial cognition), with

preference for limiting the depth access relative to the viewpoint (thus conforming to limited $2\frac{1}{2}$D visual acuity).

The following points are guidelines for restricting the mapping of information to the third dimension in an effective way.

Directness

The first restriction proposed here for the depth mapping f, is that it be the simplest possible mapping from the information to the visual space. This property is referred to as *directness*. In the definition for the two channels (Definition 2.2), f is already restricted to be a mapping of only one variable. That is, f is a "one-to-one" mapping from the semantics of the information space to geometry in the z-dimension. For example, in a traditional 2D scatterplot of two variables, the value of the variables is mapped linearly to the geometry of the axes. A log mapping is somewhat less direct.

An example of a mapping that is not direct is a multidimensional scaling in which high dimensional data is mapped to a single visual dimension in such a way that the geometric distance of data points in the visual dimension relates as closely as possible to distance in the underlying multidimensional data space. This is indirect in two ways. First, it violates the restriction that f be a mapping from a single variable as defined above. Secondly, a single position in the visual space may correspond to more than one value in the information space: that is, it is not a one-to-one mapping.

The distinction between direct semantic-based mappings and indirect mappings becomes more important when graph visualisation is introduced in Section 3.2 and the concept of an *aesthetic* mapping is brought forward.

Independence

The variable mapped to depth should not be dependent on, or derived from, any other aspect of the system and preferably it should be independent of the domain. When statisticians analyse data, they commonly construct a model in which one variable is a function of another. For example, $y = F(x)$. Here, x is known as the *explanatory variable* and y is the *response variable*. The variables x and y are also commonly referred to as *independent* and *dependent*, respectively [59, Page 161]. When plotting such data in 2D, it is standard practice to map the independent variable to the horizontal axis.

In experimental psychology, independent variables are easily defined as: 'variables whose values are chosen by the experimenter'. In physics, the x-axis in a chart is routinely mapped to a variable, such as time, that is independent of the particular phenomenon being studied. An informal "fuzzy" definition for what makes a variable independent in this and other contexts, might be that it represents a primary concept that makes sense in its own right. That might mean that the variable is not derived from other variables or that it is the least domain specific of the variables involved. Consider again the example of *time* which is a concept easily grasped by non-physicists.

In keeping the depth mapping as simple as possible, it makes sense to choose a similarly independent variable as its domain.

Discreteness

If the human vision system's depth field is limited, it makes sense to require that the depth function maps to discrete levels or parallel planes, ie:

$$f : A_j \mapsto \mathbb{N}$$

The discrete, single-dimensional channel f is clearly more limited in the type of information which can be represented than g, the continuous, two-dimensional channel. The choice of variable that is mapped to the discrete channel is therefore going to have to be carefully made.

2.2.3 Choosing the variable for the depth mapping

In [28, Page 20], Card et al. categorise variables, from the point of view of information visualisation, into three basic types:

Nominal values can only be tested for equality.

Ordinal values can be compared with the "less than" operator ($<$).

Quantitative values can be used in arithmetic operations.

For the *z*-dimension the most appropriate variable types are nominal or ordinal types. Nominal variables can also be thought of as categorical classes. For example, biological samples might come from different organisms, statistical data might come from different countries, etc. Nominal variables can also be thought of as ordinal if an ordering across the classes is self evident (for

example, classes in a high-school might be logically ordered by year-level) or can be arbitrarily defined if there is a direct relationship to some other attribute. Quantitative variables can also be converted to Ordinal variables. Some examples are:

Time — rounded to convenient units: eg. seconds, weeks, months, etc.

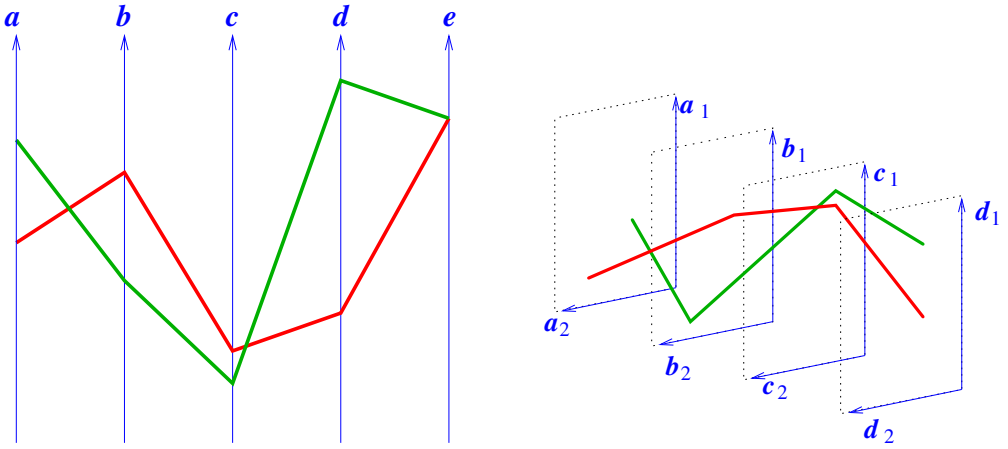
Quantiles — for example, one might class students participating in a study based on their coursework marks.

Arbitrary ordering of nominal variables in mapping to the z-dimension is discussed further in Chapter 4.

A popular 2D method for visualising data with a number of categorical classes (and therefore a spatial mapping of a nominal variable), is the *parallel coordinates* paradigm. In a parallel coordinates system the various categories or attributes of a set of data tuples are represented as a set of axes drawn as (usually vertical and equally spaced) parallel lines. A tuple is then drawn as a connected set of line-segments between axes as shown in Figure 2.2(a). The point where the tuple's line intersects a particular axis indicates the value for the attribute associated with that axis. In terms of the definitions above, a parallel coordinates visualisation could be thought of as a discrete, direct and independent mapping of a nominal variable to the x-dimension: that is, a $1\frac{1}{2}$D mapping. Similarly, an extension of parallel coordinates to 3D by visualising tuples as lines intersecting a set of parallel planes (as described by Wegenkittle et al. [198]), could be thought of as a $2\frac{1}{2}$D mapping. An example of such an extension to 3D is shown in Figure 2.2(b).

2.3 Example

By way of example, the $2\frac{1}{2}$D model described in Section 2.2 is applied to extending a well known 2D visualisation into the third dimension. Figure 2.3 shows a set of charts investigating the relationship between inflation, i.e. increase in Consumer Price Index or CPI, and male unemployment rate in major OECD countries over two economically turbulent decades. The charts originally appeared in a 1977 OECD report by McCracken et al. [134, Page 106] but were made famous by Tufte in [188] who cited them as an excellent example of a visual exploration of a supposed relationship between inflation and unemployment. The "Philips" curve predicts that there should be an inverse relationship between these variables. However, this set of charts show that for the decades



(a) Traditional parallel coordinates plot of two data tuples across five variables — a $1\frac{1}{2}$D mapping

(b) 3D parallel coordinates of two data tuples across four pairs of variables — a $2\frac{1}{2}$D mapping

Figure 2.2: Traditional 2D parallel coordinates and a possible extension to 3D.

in question no such clear relationship could be seen. McCracken offered explanations such as increased government intervention, changing international markets and effects that are now referred to as globalisation.

This set of 2D charts is designed to explore relationships amongst four variables: *time*, *country*, *increase in CPI* and *unemployment rate*. It is postulated that, with careful use of the $2\frac{1}{2}$D principles defined in Section 2.2, a 3D version of Figure 2.3 can be designed, that makes it easier for observers to understand these four dimensional relationships.

To extrude the charts into 3D, the crucial step is choosing a variable to map to the z-axis according to the principles of *directness*, *independence* and *discreteness*. Although these principles help us to narrow down the choice of appropriate mapping, there is not always a single obvious candidate variable. In the case of this example, inflation and unemployment can be rejected on two counts each. First, neither is discrete. Secondly, they are both response variables in this domain: that is, they are not independent. This leaves two possibilities for the depth mapping: *time* and *country*. Both are independent. *Country* is a nominal, discrete variable. *Time* is ordinal and has already been reduced to discrete, yearly granularity.

It is possible to argue that mapping either of these variables to the third dimension will improve on the 2D version. To make a final decision, prototype visualisations based on both mappings are created and the relative benefits of each are considered.

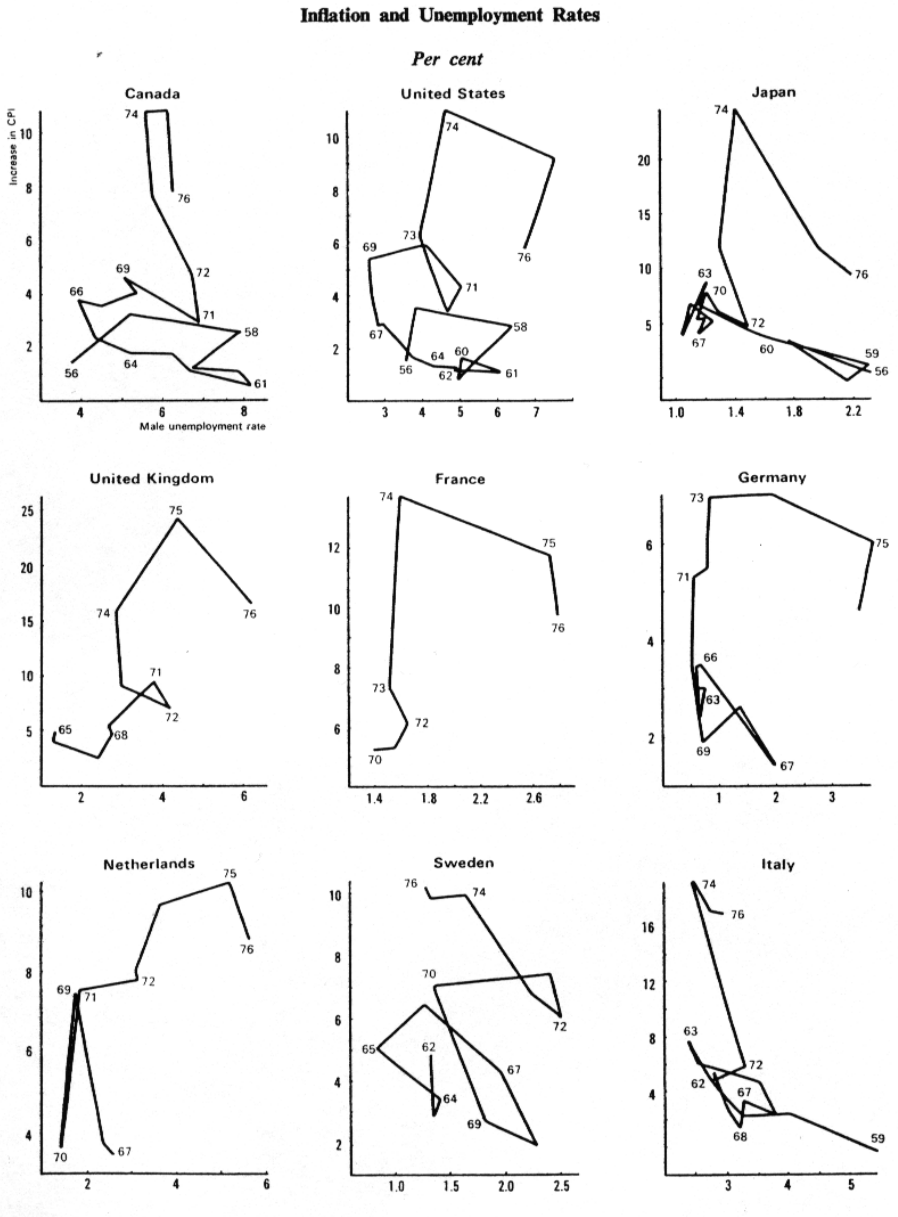
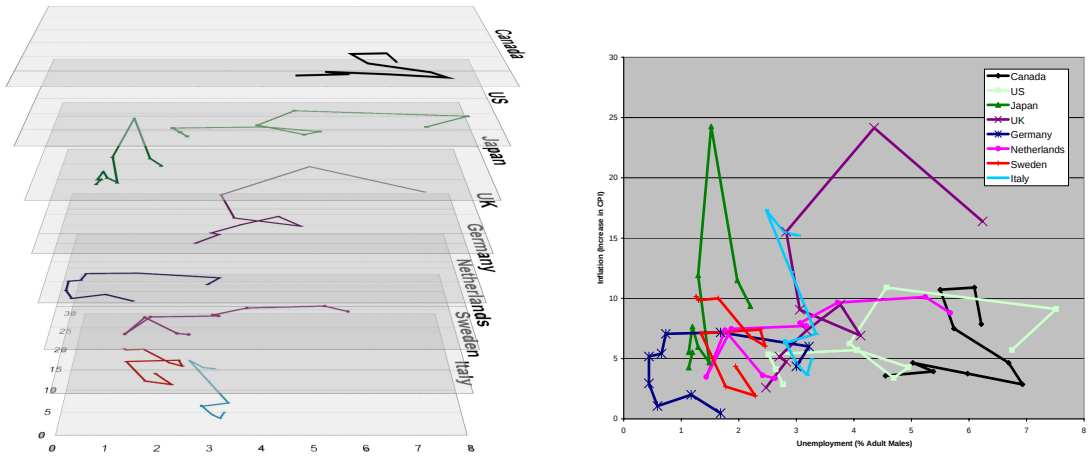


Figure 2.3: A set of charts exploring relationships between inflation and unemployment in OECD countries from McCracken et al. [134, Page 106]



(a) In $2\frac{1}{2}$D with the height dimension mapped to *country*

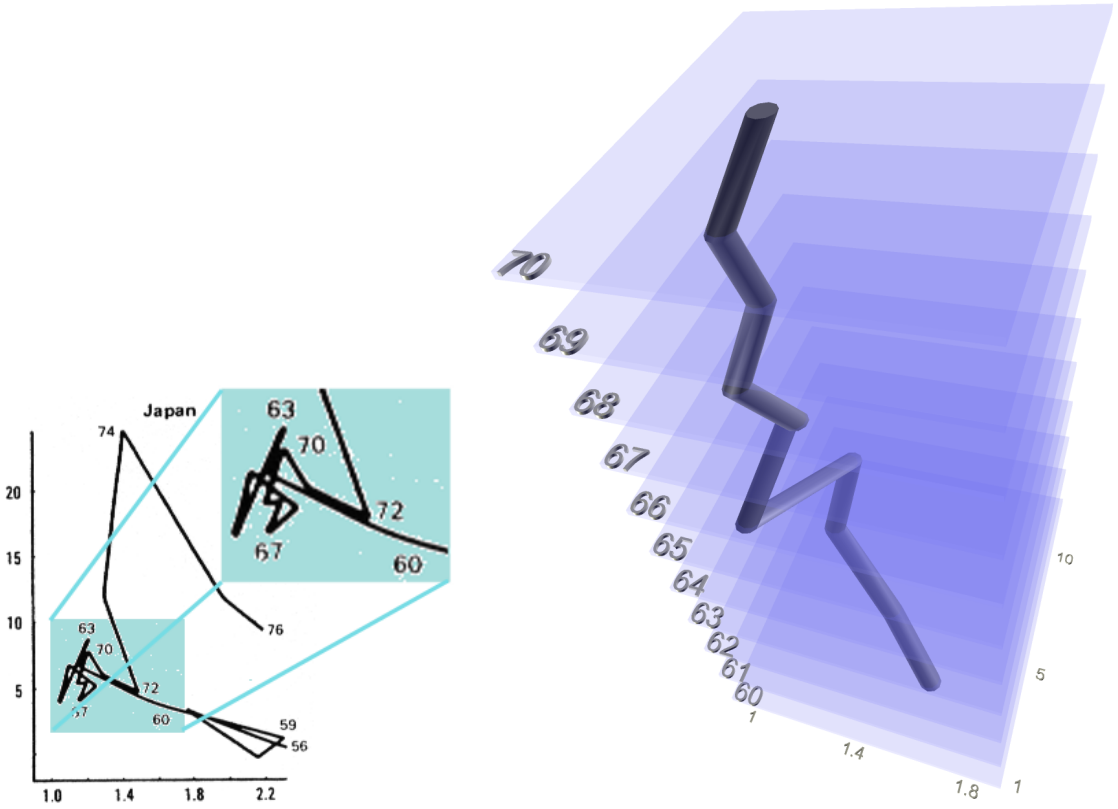
(b) Compounded into a single 2D chart

Figure 2.4: Inflation vs Unemployment 1967-1976

Mapping *country* to the third dimension is equivalent to arranging each of the charts as layers in a 3D stack as in Figure 2.4(a). The individual charts have been redrawn such that they share a common scale[3] Arguably, comparison of the different curves is made easier in this overlaid configuration than in charts placed side-by-side. However, as can be seen in Figure 2.4(b) it is also possible to provide for *in-situ* comparison in 2D by collapsing the stack. When comparing fewer curves this might be quite effective, but with eight it is confusing and tests our ability to find contrasting colours. Lifting the stack into 3D is not a solution to this problem and only adds an extra degree of awkwardness due to parallax and perspective. *Country* is therefore discarded as a possible variable for mapping to the third dimension since such a mapping offers no significant improvement over the original.

If the curves from the original figures are to be preserved, then the alternative mapping for the third dimension, *time*, requires that the curves be lifted into "worms" extending through the spatial dimension mapped to time. In the original figures, the year corresponding to each point on the curve could only be determined by reading the label. Providing a direct spatial mapping for this variable clarifies regions of the curves that were highly ambiguous in the original 2D charts, for example see Figure 2.5. This is because our ability to parse labels is limited by what Ware [193, Pages 162–170] called our *Iconic Buffer* – our limited working memory for symbol processing. Mapping to a spatial

[3]Note that most of the original charts show data from 1967 to 1976 so only these ten years are shown. Also, note that the chart for France is omitted since it did not include complete data for this period.



(a) The original chart from Figure 2.3, ambiguous region enlarged

(b) Time mapped to the 3^{rd} dimension

Figure 2.5: Two versions of the Inflation versus Unemployment chart for Japan

dimension allows for the task to be performed pre-attentively. This suggests that mapping *time* to the third dimension for this set of charts leads to concrete benefits in clearly conveying relationships between the four variables to the observer.

2.4 Experimental Study

In Section 2.3 it is suggested that mapping *time* to the third dimension for the McCracken "Phillips" curve charts has benefits in allowing an analyst to understand relationships between the four variables involved. This proposition is tested in a simple experiment.

2.4.1 Experimental Goals

An experiment was conducted that explores the question of which types of analysis benefit from extruding a 2D visualisation into $2\frac{1}{2}$D. Participants were asked a number of questions designed to make them study relationships between different types of variables. The effect of the type of visualisation presented to the participants, 2D or 3D, on their understanding of relationships between data dimensions, was measured.

This study is not intended to be a rigourous psychological experiment or a definitive answer to the question of whether 3D is better than 2D. As discussed earlier, it is believed that such a question misses the point that there are certain types of data and analysis that are better suited to 3D and that, to take advantage of the extra spatial dimension effectively, a $2\frac{1}{2}$D design approach should be used.

To be more precise, each task, or question, posed to the experimental subjects, tests the operational hypothesis H_1 that:

> *Presenting the data to participants in a 3D model, rather than in the original 2D charts, has a positive effect on their ability to understand relationships between the underlying variables.*

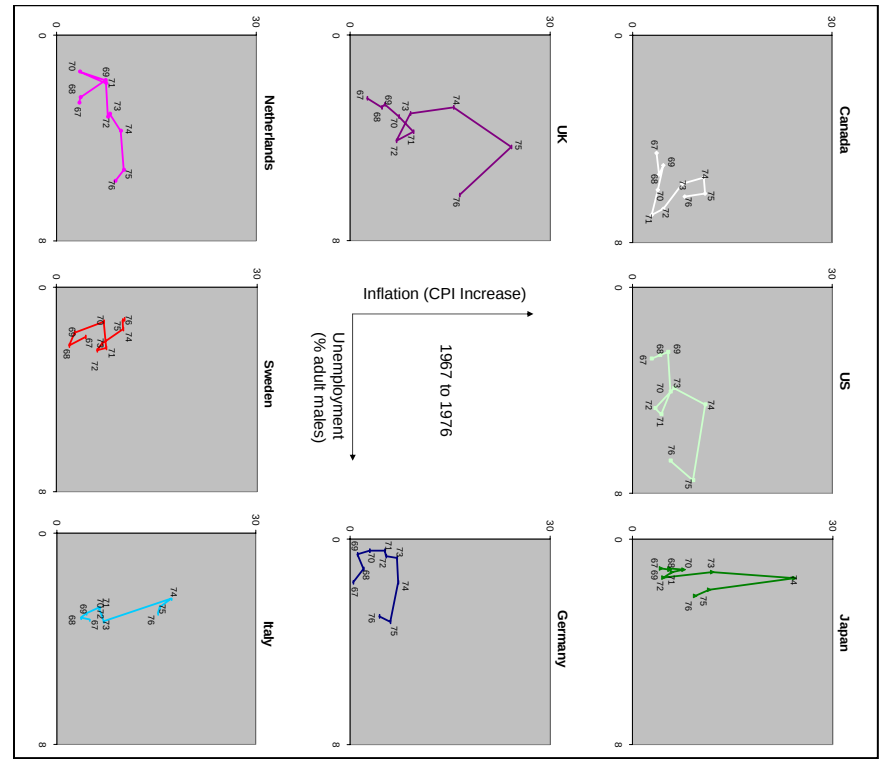
The null hypothesis H_0 is:

> *The 3D model does not have a positive effect on participants ability to understand relationships between the underlying variables.*

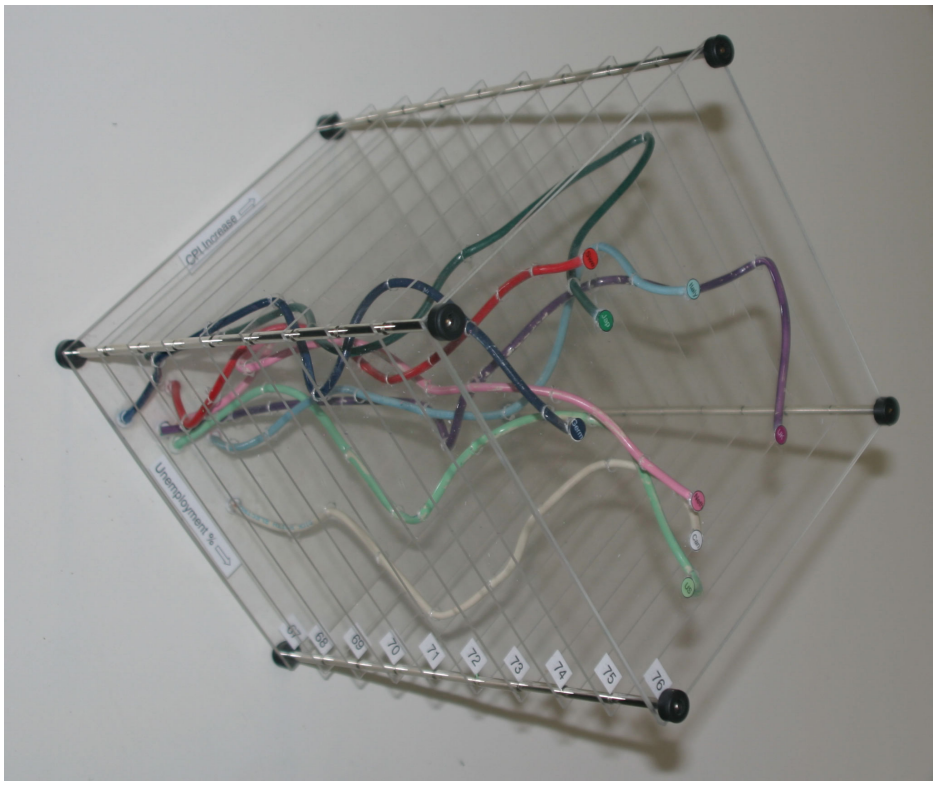
2.4.2 Experimental Method

Forty subjects, aged between 20 and 30, were arbitrarily selected from students and staff of the School of Information Technologies at the University of Sydney. These subjects were randomly assigned to two groups of twenty. Participants in the first group were given a 2D lattice array similar to the original published by McCracken et al. , see Figure 2.6(a). Participants in the second group were given a physical 3D representation of a $2\frac{1}{2}$D version of the same data, with *time* mapped to the third dimension, see Figure 2.6(b).

The experimental method used is a standard two-group, post-test-only, randomised experimental design, see [101]. Examples of similarly designed usability tests of visualisation techniques are given by Kasmarik and Thurbon [112] and Hendrix et al. [91], both studying the effect of software visualisation on users' understanding of program fragments.



(a) 2D Chart Lattice



(b) Physical $2\frac{1}{2}$D stack

Figure 2.6: The 2D and $2\frac{1}{2}$D models shown to subjects

The data-set used in both representations has been filtered leaving eight countries with curves spanning equal time periods of ten years. The original curves covered a longer time period for some countries than for others. France was left out entirely since only seven years worth of data was available. In the original representation each chart was shown at a different scale, so that the area covered by the data points was maximised. In the representations here, the charts are all shown at the same scale in order to allow individual points, for different countries, to be compared.

The new 2D lattice for the filtered data was generated with a popular charting application and then arranged manually on an A3 size page using a page-layout package. Year labels for each data-point were added by hand.

To test the $2\frac{1}{2}$D design for the charts a physical 3D model was built. The $2\frac{1}{2}$D stack was constructed with transparent perspex plates, each plate corresponding to one year. Holes were drilled through the perspex at positions corresponding to the data-points for that year. Coloured, flexible cable was threaded through holes in the plates to create a physical model of the type of "worms" shown in Figure 2.5.

Using a physical model, rather than using 3D computer graphics to create a virtual model, provides a number of benefits. Specifically, the physical model:

Requires less user training — Subjects can interact with the model directly rather than having to learn how to navigate with a mouse or other device[4].

Simulates idealised virtual/augmented reality conditions — *real* versus *realistic*: that is, the physical model offers the ultimate in stereo optics, haptic feedback, interaction, etc. and simulates ideal technology that might be available in the future[5].

Eliminates variables due to 3D rendering choices — Parker et al. [151] found that the choice of rendering environment, for example, stereo, head coupled motion, etc, had a significant effect on users' ability to understand a 3D visualisation.

Of course there are also some disadvantages. For example:

[4]For example, such difficulties with interaction were suggested as a possible confounding factor in a study by Swan et al. [182]

[5]Hopefully this will overcome limitations of current technology that have limited other studies. For example, Sebrechts et al. [171] gives a comparison between 2D and 3D systems where the 3D system used in the experiment was found to be fundamentally flawed by the illegibility of labels. This seems to be largely due to the limitations of the display hardware used.

- With a physical model it is impossible to automatically generate unlimited models based on random data.
- Virtual environments can offer types of interaction that are difficult or impossible in a real model — for example, allowing a user to view cross-sections of the model.
- The physical model is more time consuming to build and adjust.

These advantages and disadvantages are assessed further, based on feedback from the participants, in Section 2.4.5.

Since colour-coding was used in the physical model, a simple check for colour blindness was given to participants. The examiner pointed in random order to each worm in the stack and participants were required to name the colour. Those who could not differentiate between the differently coloured worms were not tested.

Participants were given a small reward for participating in the task[6].

2.4.3 Questions

The study consists of twelve questions intended to cover three categories of analysis: *single country*, *comparison between countries involving three variables* and *comparison between countries involving all four variables*. It had been expected that questions within a category would pose a similar challenge to participants and therefore their performance within each category would be similar. However, as discussed in 2.4.5, the relationships between questions turned out to be a little more complicated.

A trial study involving six additional participants, three for each type of visualisation, was completed before the main study in order to resolve problems such as ambiguously worded questions. The final set of questions posed in the actual study is as follows:

Single country

1. For the UK, from 1967 to 1971, how would you describe the relationship between inflation and unemployment?

(Answer: when inflation increases unemployment also increases)

[6]The rewards were flashing “super” balls which happened to be in good supply.

2. For Canada from 1971 to 1974, how would you describe the relationship between inflation and unemployment?

(Answer: when inflation increases, unemployment decreases)

3. In which year did Italy experience the highest unemployment rate?

(Answer: 1973)

4. Did Japan experience its highest unemployment rate at the same time it experienced its highest inflation?

(Answer: no)

Comparison between countries — Relationships between three variables

5. In which of the following years do the 8 countries experience the most similar inflation rates?

(Answer: 1968)

6. Between 1968 and 1975, what is the only year in which the Canadian employment rate moved in the opposite direction to the US employment rate?

(Answer: 73-74)

7. In what year was the maximum unemployment rate recorded and in which country did this occur?

(Answer: 1975, US)

8. Which year has the greatest difference between maximum and minimum inflation rates across all countries?

(Answer: 1975)

Comparison between countries — Relationships between all variables

9. Which country has the greatest variation in inflation together with the smallest variation in unemployment between 1972 and 1976?

(Answer: Japan)

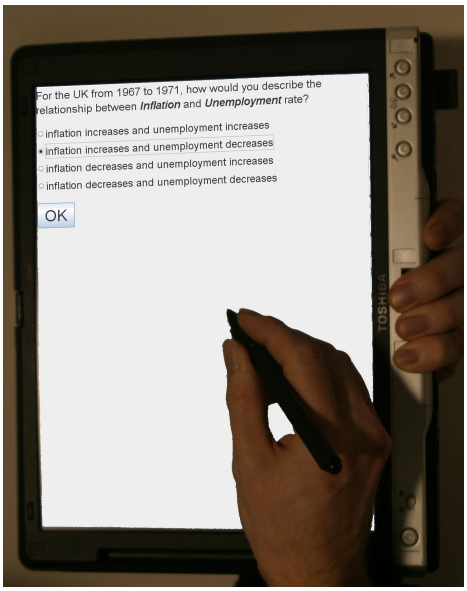


Figure 2.7: The questionnaire program in action.

10. Across all countries, is there more variation in inflation and unemployment before 1971 or after 1971?

(Answer: after)

11. Between 1973 and 1974, which countries show an increase in inflation and a decrease in unemployment?

(Answer: Canada, UK, Sweden, Italy)

12. Which of the following countries tracks the US Inflation vs Unemployment most closely throughout the entire period (i.e. on average, which country is closest to the US in terms of both inflation and unemployment at each time period)?

(Answer: Netherlands)

2.4.4 Presentation of Questions

Questions and a choice of answers were presented to participants using a small Java program running on a tablet PC. The program prompted the subject for answers and recorded results and response times. Figure 2.8 shows a simulated test in process (not a real subject).

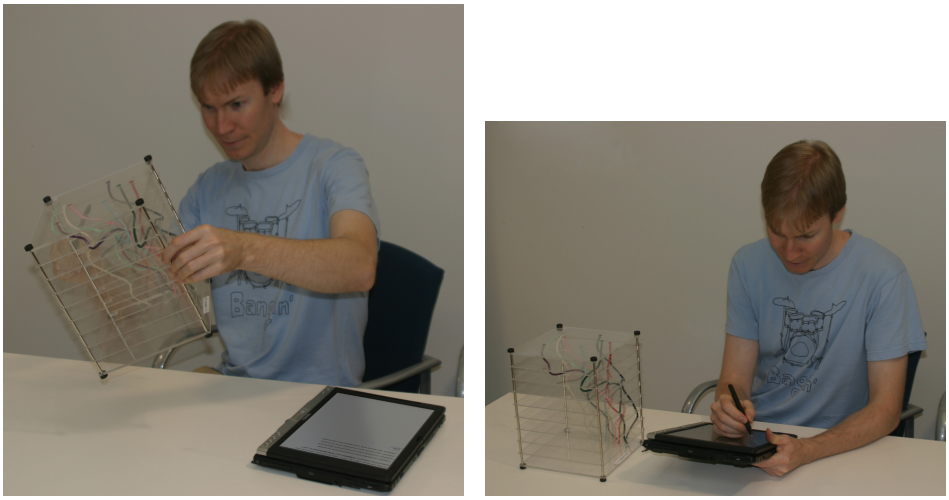


Figure 2.8: A re-enactment of the test in process, with the author posing as subject.

2.4.5 Results

Responses were collected from 40 participants divided into two groups of $N = 20$ participants. One group was required to complete the tasks using the physical 3D model while the other group used the 2D printed charts. Following [112] and [91], for each question and group the following statistics are considered:

Correctness (n_c/N) — Ratio of correct responses n_c to correct and incorrect responses.

Efficiency (n_c/T) — Correct responses per minute spent by all participants in the group on the question.

Response Time ($\bar{X}_T$) — Mean time in seconds for the participant to respond to the question.

Correct Response Time ($\bar{X}_c$) — Mean time in seconds for a correct response (excluding participants who responded incorrectly).

Looking first at the *correctness* data from Table 2.1 and shown in Figure 2.9, little appreciable difference between the two groups is seen for most questions. Across all questions, the mean correctness scores for the 2D and 3D groups, are 85% and 84% respectively.

The two largest differences in correctness occurred in questions 4 and 6, in which the 2D group scored 15% higher than the 3D group. For these two questions the decrease in accuracy for the

Qu.	Correct (n_c)		Total Time in minutes (T)		Correctness ($n_c/20$)		Efficiency (n_c/T)	
	2d	3d	2d	3d	2d	3d	2d	3d
1	19	20	23.66	22.64	0.95	1.00	0.80	0.88
2	18	19	13.28	13.48	0.90	0.95	1.36	1.41
3	20	20	10.49	13.03	1.00	1.00	1.91	1.54
4	20	17	8.64	14.35	1.00	0.85	2.31	1.18
5	12	14	28.65	26.51	0.60	0.70	0.42	0.53
6	17	14	25.44	31.69	0.85	0.70	0.67	0.44
7	18	17	12.33	12.52	0.90	0.85	1.46	1.36
8	12	13	35.30	19.66	0.60	0.65	0.34	0.66
9	17	17	20.31	23.86	0.85	0.85	0.84	0.71
10	19	19	15.28	10.49	0.95	0.95	1.24	1.81
11	19	17	23.81	28.81	0.95	0.85	0.80	0.59
12	12	14	25.50	21.28	0.60	0.70	0.47	0.66

Table 2.1: Number of correct responses, total time, correctness and efficiency by question for all participants.

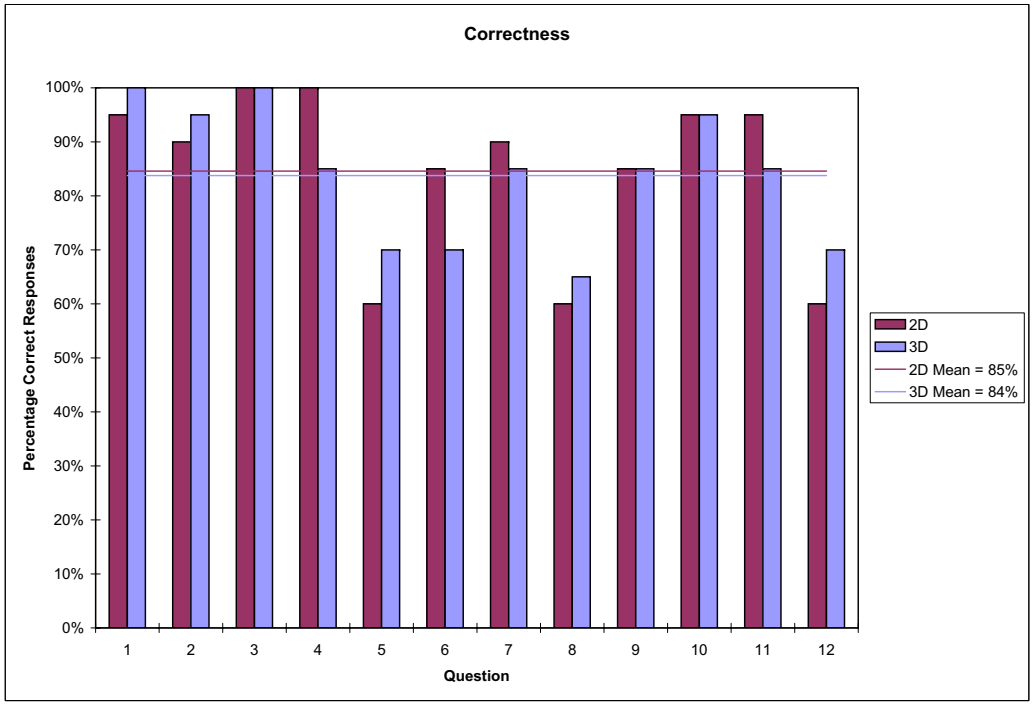


Figure 2.9: Percentage of total responses that were correct by question.

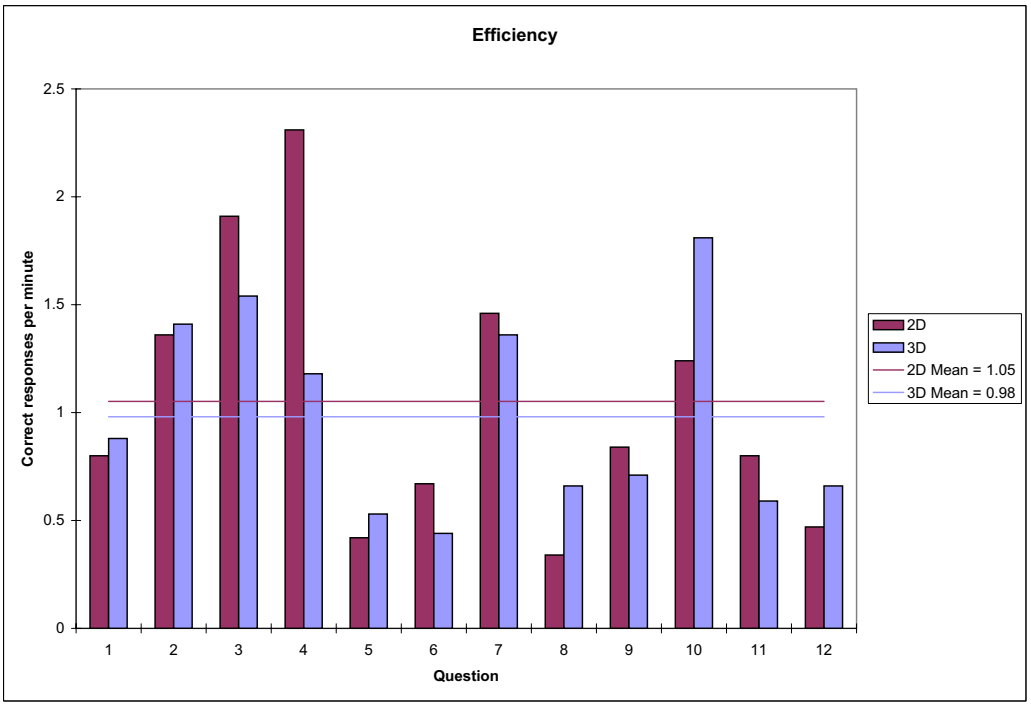


Figure 2.10: Efficiency in terms of correct responses per minute.

3D group could be due to the 3D model compacting all eight countries into a single display. Thus, noticing details for a single country, as in Question 4, becomes harder in the 3D model than in the 2D lattice where the countries are separated. Of the four questions concerning individual countries, Question 4 seems most affected by this confusion, since the curve for Japan tends to weave in and out of other curves. In hindsight, a fairer test for questions about a single country might involve comparison of a 2D chart for one country against a 3D view of a single country, similar to that shown in Figure 2.5(b). Similarly, in Question 6, which required a comparison between two countries, the benefit of being able to represent both countries in a single 3D model is somewhat outweighed by the distraction of the curves for the six other countries not relevant to the question. An ideal visualisation system would allow the user to filter data such that only the relevant details, in this case the two countries in question, are shown.

By contrast, the questions which required the subject to analyse all eight curves had higher rates of correctness for the 3D model than for the 2D model. Particularly, questions 5 and 12 showed a 10% improvement in correctness for the 3D model against the 2D lattice. Question 8, also requiring comparison across all curves, showed a 5% improvement in correctness for the 3D model over the 2D model.

Efficiency, which is of course related to correctness, is charted in Figure 2.10. Again, mean efficiency across all questions is not significantly different between the two groups. One question, notable for a significant improvement in efficiency from the 2D to the 3D representation with no difference in correctness, is Question 10. Thus, although nearly all participants in both groups were eventually able to answer the question correctly, those with the 3D model responded 46% more quickly. This question, as well as being a comparison between all eight countries, also requires the subject to consider change over time. Possibly, this is a confirmation of the point discussed in Section 2.3, that people are able to more rapidly discern differences in values mapped spatially, than in values which must be interpreted from labels.

The study of mean response times, shown in Figures 2.11 and 2.12, is more conclusive. Since there are response times for each participant in the two sample groups, it is possible to calculate confidence intervals around the sample means estimating the true population mean. For both sets of response times — all responses and correct responses only — standard deviation was calculated for each group using the sample standard deviation:

$$s = \sqrt{\frac{\sum (X - \bar{X})^2}{N - 1}}$$

For total response times, degrees of freedom $df = N - 1 = 19$ and the corresponding two-tailed critical t value at the 0.05 level is 2.093 [101]. This critical t-value is then used to estimate a range for the population mean with 95% confidence: $\mu = \bar{X} \pm t \times s/\sqrt{N}$. For correct response times N varies for each question, i.e. $N = n_c$. Thus, the $t_{0.05}(n_c - 1)$ values used to calculate confidence intervals for each group and question are listed in the last columns of Table 2.3.

For both mean response times and mean correct response times only two questions returned results without overlapping confidence intervals at the 95% level. These are questions 4 and 8. Thus, for question 4, the operational hypothesis is rejected, H_0 is accepted. That is, for this particular question the 3D model had a negative effect on the participants' ability to understand relationships between the variables involved. Note that in this question the participants did not have to compare different countries, so there was no benefit from the 3D model's compaction of all curves into a single display. Indeed, this probably just added to the confusion. Neither did the participants have to measure change of the variables over time.

By contrast, Question 8 showed an improvement in response time for participants using the 3D model that was significant at the 95% confidence level and so H_0 is rejected and the operational

Qu.	mean ($\bar{X}_T$)		Std. Dev. (s_T)	
	2d	3d	2d	3d
1	70.99±17.23	67.93±16.26	36.82	34.75
2	39.84±7.01	40.45±7.55	14.98	16.12
3	31.46±5.40	39.09±12.47	11.55	26.65
4	25.93±7.09	43.04±8.26	15.14	17.65
5	85.94±16.92	79.54±22.23	36.14	47.50
6	76.32±15.57	95.08±19.09	33.28	40.80
7	36.98±6.56	37.56±10.80	14.01	23.08
8	105.91±17.47	58.97±14.33	37.32	30.62
9	60.93±16.01	71.59±15.55	34.21	33.22
10	45.85±9.55	31.48±6.49	20.41	13.86
11	71.44±13.47	86.42±15.08	28.78	32.21
12	76.49±11.76	63.85±13.34	25.14	28.51

Table 2.2: Response time means with 95% confidence intervals and standard deviations.

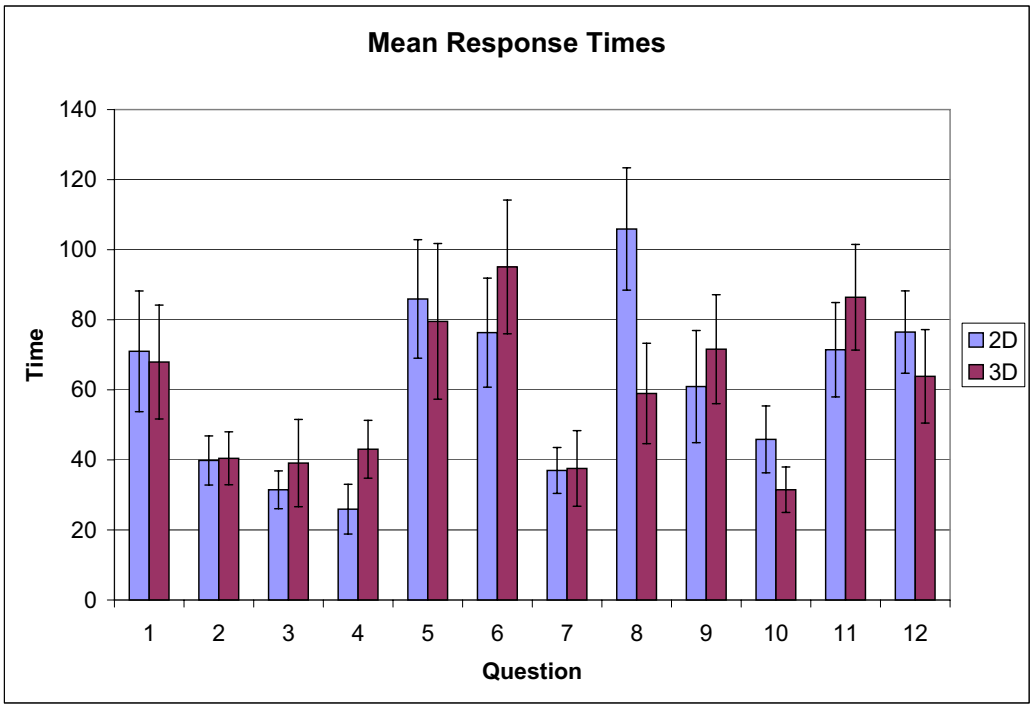


Figure 2.11: Mean total response times ($\bar{X}_T$) from Table 2.2 for each question. Error bars indicate 95% confidence intervals.

Qu.	mean ($\bar{X}_c$)		Std. Dev. (s_c)		$t_{0.05}(n_c - 1)$	
	2d	3d	2d	3d	2d	3d
1	71.00±18.23	67.93±16.26	37.83	34.75	2.101	2.093
2	40.60±7.76	40.60±7.98	15.61	16.55	2.110	2.101
3	31.46±5.40	39.09±12.47	11.55	26.65	2.093	2.093
4	25.93±7.09	44.02±9.75	15.14	18.95	2.093	2.120
5	83.39±24.23	73.94±22.88	38.13	39.63	2.201	2.160
6	79.96±17.60	86.22±19.28	34.23	33.40	2.120	2.160
7	35.62±6.97	36.18±11.90	14.01	23.14	2.110	2.120
8	110.12±24.61	63.12±18.24	38.74	30.18	2.201	2.179
9	61.57±19.12	65.48±11.31	37.18	21.99	2.120	2.120
10	46.93±9.82	31.22±6.84	20.37	14.19	2.101	2.101
11	67.46±11.19	85.08±17.07	23.22	33.19	2.101	2.120
12	76.32±19.17	68.76±16.70	30.18	28.92	2.201	2.160

Table 2.3: Correct response time means, standard deviations and critical $t_{0.05}$ values used to determine 95% confidence intervals for population means.

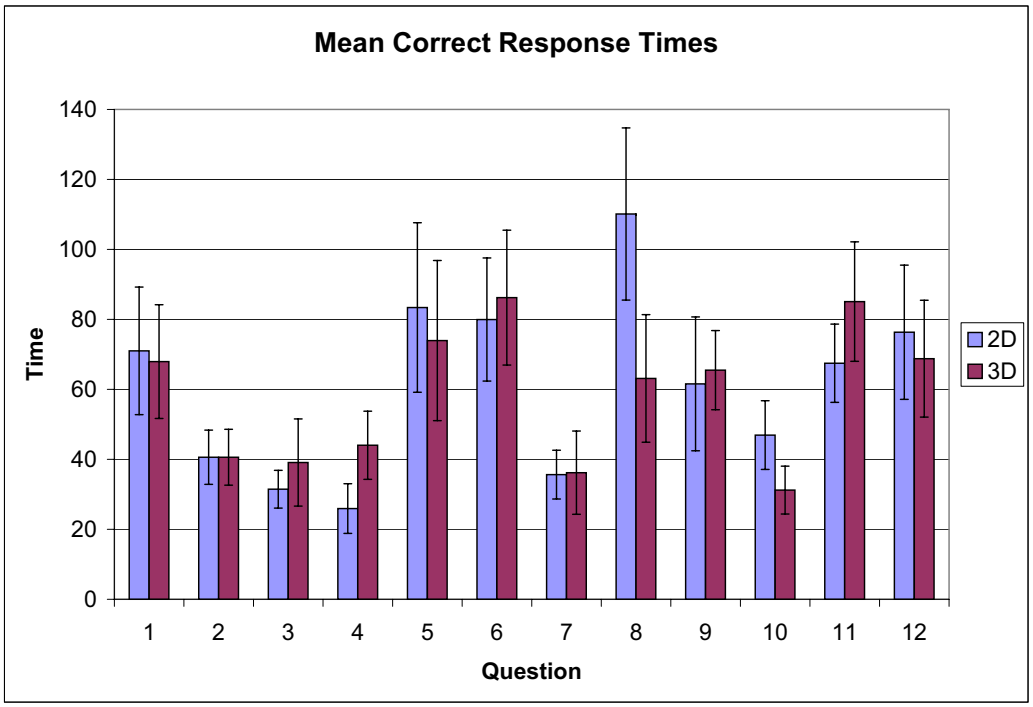


Figure 2.12: Mean correct response times ($\bar{X}_c$) from Table 2.3 for each question. Error bars indicate 95% confidence intervals.

hypothesis is accepted for this question. Note that in this question participants were required to compare all countries and to study change over time.

Though the estimated population mean confidence intervals overlap, the difference in mean correct response times for question 10 is still quite large. In fact, a two-tailed heteroscedastic t-test reveals that the probability that the difference occurred by chance is only 0.0095. Thus, H_0 is again rejected and the operational hypothesis is accepted for question 10. As discussed above, this is probably due to the spatial mapping of time in the 3D model as opposed to time shown only with labels on the 2D curves.

Results for other questions are less conclusive, and certainly H_0 cannot be formally rejected. Of the questions which require a comparison between two or more countries across time it is possible to suggest, based on feedback from participants, some explanations for why some did not experience the significant improvement observed in Questions 8 and 10. Question 5 showed a slight improvement in response time, as well as an improvement in correctness, with the use of the 3D model. However, the presentation of the multiple choice answers — a selection of years — turned out not to require a scan across time at all. Rather, participants would find the particular years listed by label on both the 2D charts and 3D model. Thus, no benefit was attained from the spatial mapping of time. A similar situation seems evident for questions 6 and 7, both of which showed a slight worsening in performance on the 3D model. Question 6, which required a comparison between two countries, was also relatively easy for the 2D group, since in the 2D chart lattice the countries in question — the US and Canada — were adjacent to each other in the top-left corner of the lattice. Had they been placed at opposite corners of the lattice the comparison might have been more difficult.

Question 8, however, as a search for maxima and minima, seems to have been completed as a scan across time, thus favouring the spatial mapping. Question 10, which required participants to review a range of time periods, seems to have required a similar scan. Question 11 only considered two specific time periods, again requiring subjects in both groups to read labels.

Although Question 9 was intended as an exercise requiring comparison across countries, feedback suggests that participants examined the choice of countries provided by the available answers and completed the task by a process of elimination. In other words, they examined each country individually and the task was effectively no different than that of questions 1 to 4.

Question 12 involved consideration of the entire range of times, and did show an improvement in accuracy and response time with the use of the 3D model, though not sufficient to reject H_0.

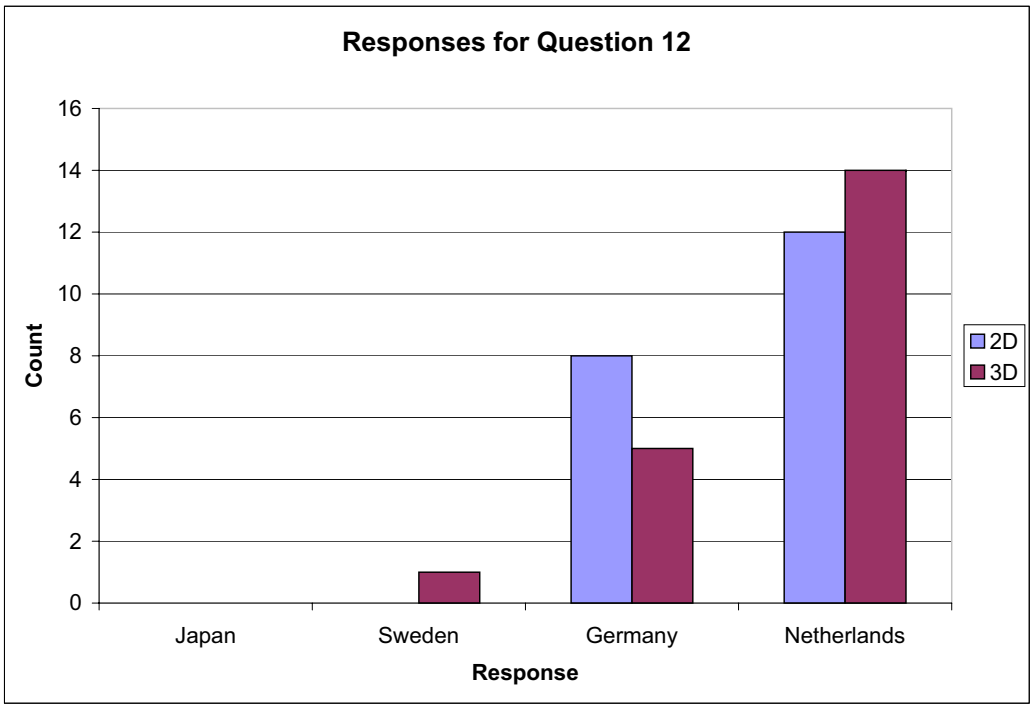


Figure 2.13: Response frequencies for Question 12. Netherlands was the best match for the US curve by distance while Germany was the best match by shape.

Interestingly, feedback from participants seems to suggest that regardless of the group, the question was consistently misinterpreted. Although the question was carefully worded with the intention of requiring the subject to look for the country curve with minimal total distance from the US curve, the subjects tended to answer by matching the shape of the curve — possibly an easier task for subjects using the 2D model. Figure 2.13 seems to bear this out.

The distribution of the underlying data for total response times and correct response times is reflected in the maxima, minima and quartiles listed in tables 2.4 and 2.5 and displayed graphically by box-plots in figures 2.14 and 2.15.

2.4.6 Remarks

This experiment has identified a number of tasks where the subjects' performance was improved by the 3D model but has also identified situations where there was a negative impact on performance.

As with most experiments that seek to assess subjects' ability to perform high level tasks based on a seemingly straightforward variable, confounding factors became evident in conducting the experiment. Most notably, the design of the 3D model was not a simple extrusion into the third

Qu.	Minimum		1st Quartile		Median		3rd Quartile		Maximum	
	2d	3d	2d	3d	2d	3d	2d	3d	2d	3d
1	22.09	26.15	39.30	41.76	59.46	56.20	101.87	98.40	149.35	147.07
2	21.88	13.06	25.73	29.25	36.01	38.99	50.34	46.83	77.42	77.88
3	19.41	11.35	21.26	21.98	28.45	31.99	38.57	41.77	55.45	113.40
4	11.77	16.30	19.41	29.89	22.15	40.56	26.83	51.33	85.49	86.17
5	32.08	19.35	53.06	46.64	81.19	64.39	112.39	95.49	166.17	203.43
6	26.82	36.67	48.19	54.24	75.00	99.03	96.32	125.53	167.45	176.49
7	16.17	13.36	24.61	22.98	33.27	31.57	48.80	42.91	67.17	112.85
8	15.90	13.68	75.04	34.24	115.58	52.92	131.37	88.65	174.34	114.75
9	18.28	32.84	48.96	48.87	56.71	68.24	67.10	84.33	178.35	181.09
10	22.95	12.05	30.16	19.35	41.64	27.84	50.85	40.79	96.28	71.61
11	34.19	46.45	46.16	57.26	67.32	78.57	83.92	118.15	147.15	141.19
12	27.09	3.44	57.25	48.36	76.00	66.82	99.00	72.89	118.07	119.88

Table 2.4: Maxima, minima and quartiles for all response times.

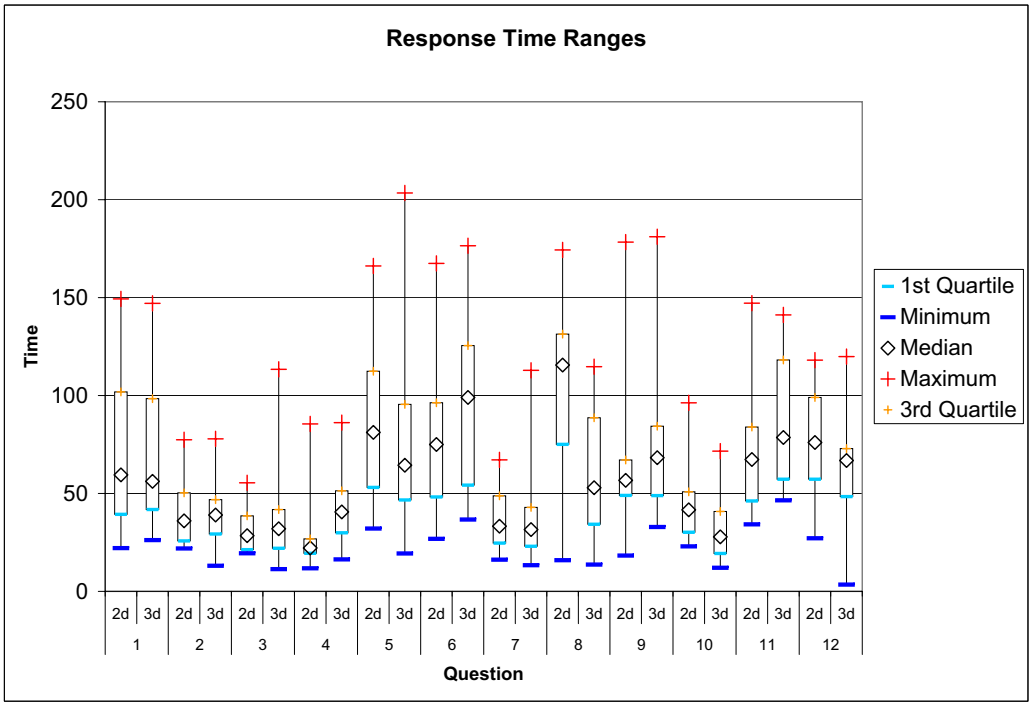


Figure 2.14: Boxplots for each question showing the range of total response times.

Qu.	Minimum		1st Quartile		Median		3rd Quartile		Maximum	
	2d	3d	2d	3d	2d	3d	2d	3d	2d	3d
1	22.09	26.15	39.30	41.76	57.88	56.20	101.87	98.40	149.35	147.07
2	21.88	13.06	25.73	29.25	37.73	39.81	50.34	46.83	77.42	77.88
3	19.41	11.35	21.26	21.98	28.45	31.99	38.57	41.77	55.45	113.40
4	11.77	16.30	19.41	29.89	22.15	41.64	26.83	51.33	85.49	86.17
5	32.08	19.35	53.06	53.55	77.77	58.33	105.53	95.32	166.17	183.11
6	26.82	36.67	59.10	54.24	81.34	90.11	96.32	115.40	167.45	127.08
7	16.17	13.36	24.61	23.71	30.79	30.95	47.51	41.61	67.17	112.85
8	60.88	22.19	74.95	35.90	118.94	61.73	147.71	88.65	174.34	110.26
9	18.28	32.84	48.96	48.87	56.45	66.36	67.10	81.96	178.35	117.63
10	22.95	12.05	34.02	19.35	42.45	26.65	50.85	40.79	96.28	71.61
11	34.19	46.45	46.16	57.26	64.53	78.24	80.08	108.77	119.14	141.19
12	27.09	21.12	52.12	51.42	79.06	67.93	107.21	77.59	118.07	119.88

Table 2.5: Maxima, minima and quartiles for correct response times.

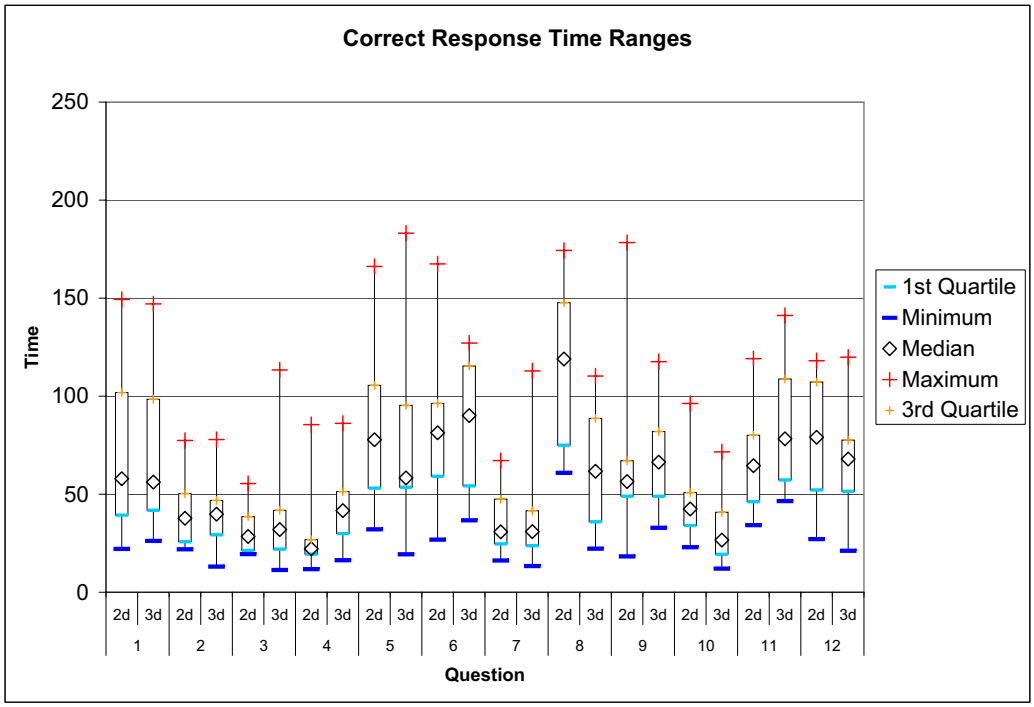


Figure 2.15: Boxplots for each question showing the range of correct response times.

dimension of the original 2D charts. The data for eight different countries was compounded from the lattice of 2D charts into a single model or "scene". A fairer comparison might be between the 3D model and a similarly compounded 2D chart — such as the one shown in Figure 2.4(b) or between the chart lattice and a set of 3D models similar to that shown in Figure 2.5(b), each showing the inflation versus unemployment data for a single country.

Section 2.3 raised two situations where mapping to 3D following the $2\frac{1}{2}$D design guidelines could be advantageous:

1. When it enables direct comparison of different sets.

2. When it facilitates mapping a variable to a spatial dimension.

In hindsight, the $2\frac{1}{2}$D mapping to 3D, though primarily aimed at addressing the first situation also enabled direct comparison by compounding the sets of data for different countries into a single model. This duality makes the study richer than either of the more ideal scenarios outlined above but also makes it more difficult to fairly assess the results.

In summary, situations were found which did indeed seem to demonstrate the above advantages for the $2\frac{1}{2}$D mapping but circumstances were also discovered where the added complexity of the compound visualisation made it difficult to see a benefit and yet other situations where it was a clear disadvantage. The task which showed a significant reduction in performance for the 3D model, reminds us that care needs to be taken in mapping to 3D and that comparing only two or three variables can probably be accomplished best in 2D. Though mapping to 3D provides greater flexibility when comparing more than three variables, facilities for viewing any two or three of the available variables in 2D should still be provided.

2.5 Conclusion

This chapter introduces $2\frac{1}{2}$D visualisation as 3D visualisation in which the third dimension is treated in a fundamentally different way to the other two. It is defined more formally as a mapping from an information space to two separate visual channels. One channel maps the information space to position in 2D and visual attributes such as glyphs, colour, texture and so on. The other channel is a *direct* and *discrete* mapping to position in the third dimension from an *independent* variable in the information space.

The extended example of an application of $2\frac{1}{2}$D visualisation and the associated experiment demonstrated various advantages of this type of visual mapping. Generally, $2\frac{1}{2}$D mapping is useful when it provides direct comparison between different categories of data or when it allows a third variable to be mapped to spatial position. Mapping to spatial position supports pre-attentive processing and is easier for a viewer to interpret than, for example, labels. Other possible mappings also support pre-attentive processing: for example [193], colour, texture or symbols; however, a spatial mapping allows for these visual mappings to be reused for other variables (such as the mapping of colour to country in the example). Thus, a $2\frac{1}{2}$D mapping is useful when studying relationships between three or more variables. The most significant negative impact of the $2\frac{1}{2}$D mapping is that it makes simple relationships between two variables more difficult to see.

An interactive, computer-generated visualisation system can provide a range of 2D and 3D visual mappings of data. Ideally, the user is free to choose the most appropriate style of visualisation for the types of variables being studied. Thus, a 3D view can provide an overview of relationships between several variables; but when analysts are more interested in relationships between a particular pair of variables they should be able to switch to a 2D view. For example, in the application systems described in Chapters 5 and 6, cross-sectional views of the 3D model are provided. The cross-sectional viewer allows the user to examine the detailed state of the model for each integer value of the independent variable. That is to say, the cross-sectional viewer allows users to isolate individual planes in the $2\frac{1}{2}$D model.

CHAPTER 3

Three-dimensional Graph Visualisation

> "Well I think we've sorted all that out now. If you'd like to know, I can tell you that in your Universe you move freely in three dimensions that you call space. You move in a straight line in a fourth, which you call time, and stay rooted to one place in a fifth, which is the first fundamental of probability. After that it gets a bit complicated, and there's all sorts of stuff going on in dimensions 13 to 22 that you really wouldn't want to know about. All you really need to know for the moment is that the Universe is a lot more complicated then you might think." — Douglas Adams, *Mostly Harmless*

In this chapter some topics concerning relational-network (graph) visualisation are introduced. These topics are central to the rest of this thesis. Section 3.1 defines some general graph concepts and Section 3.2 introduces the state of the art in 3D graph visualisation; particularly looking at how well various common 3D graph layout methods utilise $2\frac{1}{2}$D design principles.

3.1 Graphs

In this section some terminology from graph theory [42] is defined as it is used throughout the remainder of this thesis.

A graph is a mathematical construct typically modelling a network of relationships between entities such as people, places, organisations, abstract concepts and so on. The standard formal definition for a general graph is: a graph $G = (V, E)$ is made up of a set of nodes V and a set of edges $E = \{\{u, v\} | u, v \in V\}$ connecting pairs of nodes.

A *hyper-graph* is a graph $G_H(V, E_H)$ in which edges $e \in E_H$ (called *hyper-edges*) connect a set of nodes $\{u_1, \ldots, u_n\} \subseteq V$. That is, each hyper-edge may connect more than just two nodes.

A *bipartite graph* is a graph $G_B = (V_1, V_2, E_B)$ in which the nodes are partitioned into two logical sets V_1 and V_2. Each edge $e \in E_B$ connects a pair of nodes u, v where $u \in V_1$ and $v \in V_2$. A hyper-graph $G_H = (V, E_H)$ can be represented as a bipartite graph G_B, such that nodes in the hyper-graph are mapped to nodes in one of the bipartite node sets, i.e. $V_1 = V$ and for each hyper-edge $f \in E_H$ there exists a node $v \in V_2$ and a set of edges $E_v \subseteq E_B$ where $E_v = \{u, v | u \in V_1\}$.

A *directed graph* or *digraph* is a graph made up of *directed edges*, that is, each edge (u, v) is an ordered pair. Directed edges are usually drawn with an arrow from the first node to the second.

A *directed cycle* is a path or sequence of nodes $(v_1, v_2, \ldots, v_n)$ in the graph in which for every pair of neighbouring nodes in the sequence, v_i and v_{i+1}, there exists a directed edge (v_i, v_{i+1}); the first node in the sequence is also the last node, i.e. $v_1 = v_n$; and all other nodes $v_2 \ldots v_{n-1}$ in the sequence are unique.

Directed acyclic graphs , that is, directed graphs with no directed cycles, may be drawn with nodes arranged such that all edges are monotonic in the direction of the arrow. For example, trees, being a type of directed acyclic graph, are often drawn with the arrows omitted but the hierarchy indicated by the ordering of the nodes, i.e. with all edges directed away from the root.

A *clustered graph* $C = (G, T)$ is a graph G over which an hierarchical grouping (clustering) T of the nodes is defined. The tree T is usually defined such that the leaves of the tree are the nodes of G [62]; however, directed acyclic graph definitions for T are also possible. The latter definition allows for nodes to be shared between disjoint clusters. In this thesis the tree definition is used.

3.2 3D Graph Visualisation

The process of producing visualisations of graphs is a fairly mature field of study within the larger domain of information visualisation. The field is traditionally called *Graph Drawing*. The term "drawing" is arguably a little out of date, since it tends to imply static pictures of graphs on a

printed two dimensional page. However, to date the field has grown to include animated displays of graphs [72, 71], interactive graph visualisation [165, 43, 135] and immersive 3D graph visualisation [196, 151]. Therefore, the more generic term *Graph Visualisation* is used in this work.

Di Battista et al. [11, Page 6] and Kamada [108, Page 29] give formal definitions for the mapping of graph elements to spatial positions. They define a *drawing* Γ of a graph $G = (V, E)$ as the combination of a function mapping nodes to positions $\Gamma(V)$ and a function mapping edges to simple open Jordan curves $\Gamma(E)$ with end points $\Gamma(u)$ and $\Gamma(v)$ where $E = (u, v)$. Note that in this definition Γ is purely a spatial mapping, that is, nodes are infinitesimal points and edges are infinitely thin curves.

In practical applications, graphs often have attributes associated with nodes (A_V) and edges (A_E). For example, edges may be associated with distances or flow in a graph representing a transport problem. In another example, nodes in a graph modelling a computer network may need to show the load on a server. These attributes are typically mapped to colour or size of the graphical elements in the final drawing but may also have some importance in layout. For example, sets of nodes grouped according to some attribute may be drawn on the same layer in layered drawings of digraphs or placed close together in clustered graph drawings.

The mapping Γ can be rewritten to include mappings for A_V and A_E to visual attributes, in terms of the definition for information visualisation given in Equation 2.1:

$$\begin{aligned} \Gamma &= (\Gamma_V, \Gamma_E) \\ \Gamma_V &: V \times A_V \mapsto \mathbb{R}^k \times \Theta \\ \Gamma_E &: E \times A_E \mapsto C^k \times \Theta \end{aligned} \tag{3.1}$$

Here Γ_V maps each node $v \in V$ and its attributes $A_v \in A_V$ to a position in $k(\leq 3)$ dimensional space with an appropriate graphical representation in Θ. Similarly, Γ_E maps each edge $e \in E$ with attributes $A_e \in A_E$ to a Jordan curve $c \in C^k$ in k-dimensions with the attributes A_e mapped to visual parameters Θ such as width or colour[1].

Classical graph drawing methods are typically characterised as follows:

- They are primarily concerned with the spatial mapping $V \times E \mapsto R^k \times C^k$ induced from Γ — commonly referred to as *graph layout*.

[1] Drawings of very large graphs may omit edges altogether, implying edges through inter-node proximity.

- The graph layout problem is usually based purely on the graph-theoretic structure of the data-set and ignores the underlying semantics of the data domain.
- Layout methods typically aim for *aesthetics* and *readability* — concepts that are generally based on personal preference, intuition or poorly understood cognitive models of the way humans perceive groupings and structure in pictures.

In graph drawing algorithm design, a number of aesthetics have been cited as influencing the readability of graph drawings. The most commonly cited aesthetics, for example [11, Pages 14–16] and [113, Page 19] are:

Edge bends — edges should be drawn as straight as possible.

Edge crossings — where possible, edges should not intersect.

Edge length — edges should be drawn with similar length.

Node-Node/node-edge overlaps — nodes that overlap look like a single node and should therefore be avoided. Similarly, a node overlapping an edge looks as though it is connected to two edges.

Minimum edge angle — where a node is connected to more than one edge the angle between those edges should be maximised.

Orthogonality — edges should be drawn parallel to the coordinate axes.

Area minimisation — a graph may be more readable if the nodes are evenly distributed throughout the available space.

Clustering — drawing a graph such that cohesive groups of nodes are spatially grouped while low-degree, isolated nodes are shown as outliers helps to show graph structure.

Symmetry — reflectional and rotational symmetries should be maximised.

Many of these aesthetics are based on anecdotal evidence and the efficacy some, such as orthogonality, have been called into question by Purchase et al. [153] based on the results of experimental studies. Note also that some aesthetics conflict with others — for example, clustering and area minimisation — and some may only be relevant in certain application domains. More recent work,

such as that of Ware et al. [197] and Nesbitt et al. [147], has suggested links between readability of graph drawings, gestalt principles, and neurophysiology. However, graph understanding is such a high-level task that it is difficult to draw definitive conclusions.

Section 2.2 discusses a *directness* property for mappings of data attributes to spatial position in a visualisation. A direct mapping is a one-to-one transformation of the semantics of the information space to geometry; for example, the mapping of an integer value to an integer coordinate in one spatial dimension. The mapping from graph theoretic structure to geometric layout, based on aesthetics such as those described above, are much more abstract and complex. The spatial mapping in Γ is therefore considered an indirect *aesthetic mapping*.

3.2.1 3D Graph Layout Methods

The aesthetics for graph visualisation discussed above were originally intended for 2D drawings. However, since 3D graph visualisation involves understanding a 2D projection of a graph arranged in a 3D space, similar aesthetics apply. It should be noted, however, that problems such as edge crossings in non-planar graphs can be resolved through interactive interfaces allowing 3D rotation.

There are many possible methods for finding arrangements of graphs in a 3D space (i.e. 3D graph layout). Next, some of the most popular methods are reviewed with particular reference to the way that they use the third dimension and how this usage relates to the theme of $2\frac{1}{2}$D visualisation.

Force-directed Layout

The most commonly used method for arranging general graphs — not including simple tree layouts — is *force-directed* layout. The general approach involves modelling the graph as a physical entity subject to forces similar to those found in nature. Forces and the properties of nodes and edges are chosen such that an arrangement minimising the total energy of the system also satisfies aesthetics such as those described in Section 3.2. Although originally conceived for arranging graphs on a 2D screen or page, the methods are usually easily extended to 3D. Figure 3.1 shows an example of a graph arranged in 3D using a force-directed technique. Note the characteristic resemblance to a molecular structure.

Force-directed layout methods have been invented a number of times, under different guises, in various research communities. To the author's knowledge, Fisk and Isett [66] give the earliest suggestion of a technique for arranging network diagrams with an iterative method that simulates

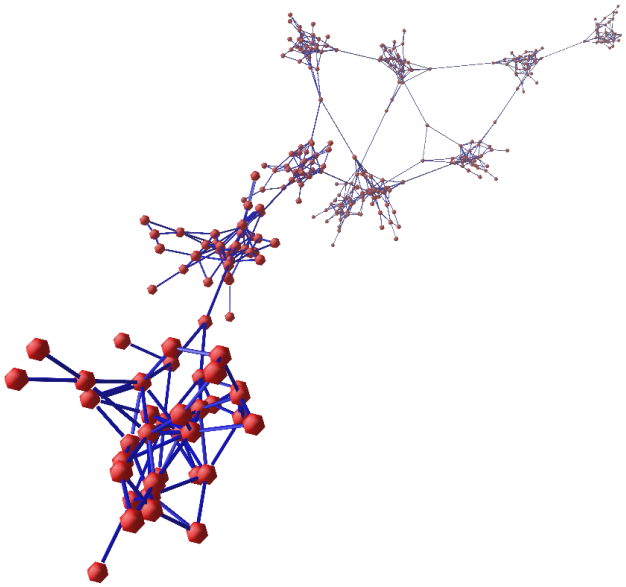


Figure 3.1: A graph arranged in 3D using a force-directed method.

physical forces. Although their method defines the repulsive and attractive forces that are characteristics of force-directed graph drawing techniques, their work is aimed squarely at designers of integrated circuit boards. As such it was overlooked by the graph-drawing community for many years.

The earliest suggestion of such a technique in the graph-drawing literature is Eades' spring embedder [52] which finds a pleasing layout for small graphs by balancing a constant repulsive force between all nodes against a logarithmic attractive force between nodes connected by an edge. The method proceeds iteratively, each iteration moving nodes by a small amount according to a force vector calculated by summing the attractive and repulsive forces. Usually, the arrangement eventually settles to a stable layout.

Kamada and Kawai [109] model the forces slightly differently, with springs following Hooke's law, between every pair of nodes. A preprocessing step sets the strength of each spring proportional to the graph theoretic distance between its end nodes. From this they derive an objective function for optimal placement of a node based on the combined springs from all the other nodes. They solve for each node in turn using a Newton-Raphson method and move the node to the optimal position found. Of course moving a node changes the forces affecting nodes already considered, so this algorithm must also be run iteratively until the layout converges to a stable state.

Fruchterman and Reingold [73] modified Eades' algorithm to more closely approximate the

physical analogy of electrostatic repulsive force between any two nodes u and v, at positions p_u and p_v respectively:

$$f_{repulsive}(u,v) = \frac{k^2}{|p_u - p_v|} \cdot p_u\vec{p}_v \tag{3.2}$$

and attractive forces between any two nodes u and v connected by an edge:

$$f_{attractive}(u,v) = \frac{|p_u - p_v|^2}{k} \cdot p_u\vec{p}_v \tag{3.3}$$

Frick et al. [70] introduced a number of refinements to this basic model. They added an extra force applied to all nodes attracting them to the centre of the visual space, thereby preventing disconnected components from repelling each other out of the viewable area. They also added heuristics aimed at reducing the number of iterations required to reach a stable state by avoiding unproductive oscillations or rotations of nodes thus.

Although such measures reduce the total number of iterations by a significant factor, all these methods share the same basic computational complexity for each iteration as Eades' original algorithm. That is, for a graph with n nodes each iteration is dominated by the computation of $f_{repulsive}$ between all pairs of nodes, i.e. $O(n^2)$ complexity. Fruchterman and Reingold tried to reduce this complexity by introducing a simple grid-based spatial decomposition such that repulsive forces on each node were only considered for nodes in adjacent cells in the grid. Although this method successfully reduces the complexity of each iteration to $O(n)$, the algorithm often fails to untangle graphs larger than a few nodes.

More recently a number of solutions to this problem have been devised. Quigley [154] extended Fruchterman and Reingold's grid method to a recursive spatial decomposition, using quad-trees in 2D or oct-trees in 3D. Other methods [192, 89] used a graph-theoretic recursive clustering to find a coarse layout of the graph before proceeding to more detailed refinement.

A close cousin of force-directed layout is *multi-dimensional scaling* (MDS). MDS has long been used by statisticians as a method for plotting multivariate data in a visualisable space such that the Euclidean distances between points in the plot preserve — as much as possible — the relative distances between points in the underlying high-dimensional data space.

More formally, if P is a set of points in m dimensions, then Multidimensional scaling computes a point $g(p)$ in k dimensions, where $m >> k$. A function g is required, such that for all pairs (p,q) of points in P, $c \cdot d_k(g(p), g(q))$ is approximately the same as $d_m(p,q)$ for some scale factor c.

Here d_k and d_m are, most commonly, both Euclidean distance functions in k and m dimensions respectively — that is:

$$d(p,q) = \sqrt{\sum_{\alpha=1}^{m}(p_\alpha - q_\alpha)^2}$$

However, other proximity metrics are sometimes used for d_m, for example, the *Minkowski metrics* [122] are defined as:

$$d(p,q) = [\sum_{\alpha=1}^{m}(p_\alpha - q_\alpha)^r]^{\frac{1}{r}}$$

for some r, where the well known case $r = 1$ is commonly referred to as the "city block metric".

Typically, multidimensional scaling is achieved by minimising a stress function for the entire data-set. The first step is to compute the covariance matrix $X = (X_{ij})$ where, if $P = \{P_i : 1 \leq i \leq n\}$, then $X_{ij} = d_m(P_i, P_j)$. Then $g(p_i)$ is computed to minimise the stress function:

$$\sigma = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}(X_{ij} - Y_{ij})^2$$

where $Y_{ij} = d_k(g(p_i), g(p_j))$.

An early observation of the applicability of MDS techniques to finding embeddings for graphs was given by Kruskal and Seery [123]. They define the covariance matrix X based on graph-theoretic distances between all nodes in the graph and then find an embedding for this graph by minimising the stress function using a "classical" MDS approach based on principal coordinates analysis [14]. Note the similarity between this graph layout technique and the independently developed "spring-embedder" technique of Kamada and Kawai (described above). The essential difference being the method used to minimise the "stress function" of the former, versus the iterative Newton-Raphson minimisation of "spring energy" in the latter.

Recently, MDS techniques that were once overlooked by graph drawing researchers have started to make an impact in this field. For example, Koren and Harel [90] use a simple algorithm to find a graph embedding in a very high dimensional space (the authors suggest using around 50 dimensions) and then projecting this high-dimensional embedding down to a visualisable space of two or three dimensions. By using principal components analysis (discussed in more detail in Chapter 5) they are able to achieve this projection in time linear in the number of nodes in the graph. As reported in the paper, the results are similar to the results of force-directed layout — at least for graphs that have a regular mesh structure. A recent study by Hosobe [100] suggests

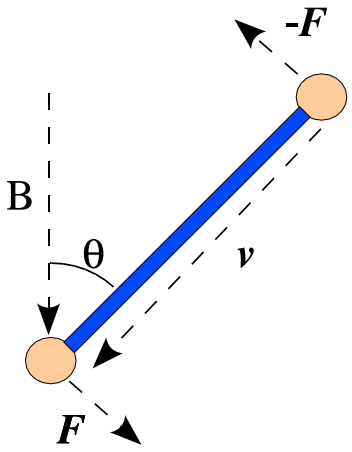


Figure 3.2: The forces exerted on two nodes connected by a directed edge due to the magnetic-field force as defined in Equation 3.4

that the results may be less pleasing for more tree-like graphs. Another paper by Koren et al. [36] combines another long overlooked technique[2] with modern graph coarsening techniques such as those of Walshaw (described above) to produce a scalable technique that seems to work well on a wider range of graph structures.

In most implementations of force-directed layout in 3D all three spatial dimensions are treated equally. That is, the graph is allowed to expand in all directions to minimise the attractive and repulsive forces. Thus, the basic 3D force-directed layout is not at all $2\frac{1}{2}$D in its approach. However, it is not difficult to add forces that constrain the layout in a more $2\frac{1}{2}$D manner. For example, for 2D layout, Sugiyama and Misue suggest a "field force" [180] for arranging directed graphs such that edges are upward pointing. The force causes edges to be rotated to be aligned with one axis. Carmel et al. [29] achieve a similar result with a faster method based on one-dimensional optimisation.

The concept is easily extended to 3D, and is demonstrated in web visualisation by Brandes et al. [20] and in a software visualisation application by Dwyer [45]. For example, in the latter application, the force exerted on an edge $(u, v) \equiv \vec{v} = p_v - p_u$ and a magnetic field of direction and magnitude given by $\vec{B}$ is defined as (see Figure 3.2):

$$\vec{f}_{magneticfield}(\vec{v}) = \frac{\vec{v} \times (\vec{v} \times \vec{B})}{\|\vec{v}\|^2} \tag{3.4}$$

Although such a magnetic-field force has a $2\frac{1}{2}$D flavour since one dimension is treated differ-

[2] In 1970 Hall [87] demonstrated that a graph could be arranged using an eigenvector decomposition of its Laplacian matrix. The details go beyond the scope of this introduction.

ently to the others (see Section 2.2), we can go much further. In Chapter 4 further extensions to force-directed layout for *stratified graph visualisation* are explored; adhering to the $2\frac{1}{2}$D principles of *directness*, *independence* and *discreteness*.

Orthogonal Layout

An orthogonal graph drawing is a drawing with nodes assigned to integer x, y and z coordinates and edges drawn parallel to these axes. Since such drawings are difficult to achieve for many graphs, usually edges are allowed right-angle bends.

Interest in orthogonal graph drawing has often been justified by its potential application to Very Large Scale Integration (VLSI) circuit design. In VLSI the tools used to manufacture circuits are sometimes limited to creating orthogonal connections on layered circuit boards. However, it has also been advocated [113, Page 121] as a form of effective graph visualisation since, by definition, the minimum edge angle in an orthogonal drawing (without hyperedges) is $\frac{\pi}{2}$. Of course, this requires that the maximum node degree is four. A concrete advantage for moving to 3D in this paradigm then, is that the maximum node degree can be increased to six while still maintaining the minimum edge angle of $\frac{\pi}{2}$.

Despite, or perhaps because of, strict satisfaction of the minimum edge angle aesthetic, Purchase [153] found that orthogonal graph visualisations in 2D were less effective than other modes, such as force-directed and layered arrangements, in conveying graph structure to users. It would appear that most of the interest in 3D orthogonal graph drawing algorithms is due to the theorem-proving possibilities they offer combinatorial mathematicians, rather than their potential for visualisation.

Although orthogonal graph drawing restricts layout such that edges are parallel to the axis, the graph is still free to grow in all three dimensions. So, it is by no means a $2\frac{1}{2}$D paradigm. For this reason, and also because of the doubts about its effectiveness in conveying graph semantics, orthogonal graph drawing styles are not considered again in this study.

Symmetric Layout

Although, some of the aesthetics listed in Section 3.2 are contentious, few people argue that it is desirable for a graph drawing algorithm to find an embedding that clearly displays any symmetries or isomorphic subgraphs in graphs. If such an algorithm can be found, then there is a provable benefit in moving from two dimensions to three dimensions. In 2D the only possible types of

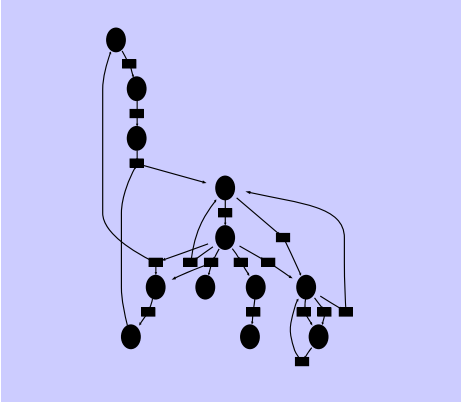


Figure 3.3: A typical hierarchical graph layout — produced with the DOT and WILMASCOPE software as described in Chapter 6

symmetry are rotation about a point and reflection over an axis. In 3D, however, there are four possible types of symmetry: rotation (about an axis), reflection (in a plane), inversion (reflection through a point) and rotary reflection (both rotation and reflection) [131]. Hong [97] demonstrates this with the example of an icosahedron. In 2D an icosahedron can be drawn with at most six symmetries, while in 3D it can be drawn with 120 symmetries.

Most of the work on symmetric graph visualisation cannot be applied to general graphs. The problem of finding whether a general graph has symmetries, and hence the problem of finding largest symmetric subgraphs within an asymmetric graph, is $\mathcal{NP}$-complete. This is true in both 2D [126, 128] and 3D [97]. For this reason, advanced combinatorial methods for symmetric graph drawing are not yet useful in general graph visualisation.

On a positive note, however, Eades and Lin [55] show that standard force directed methods are actually quite effective at displaying symmetries or near symmetries, without explicitly detecting them.

Hierarchical Layout

The hierarchical or Sugiyama [56] layout method aims to arrange the nodes of a directed graph in layers such that sources — nodes with only outgoing edges — are placed on the top layer and sinks — nodes with only incoming edges — are placed on the bottom layer. Edges are allowed only between nodes on different layers and, as much as possible, should be downward pointing. Figure 3.3 gives a small example of a graph arranged using hierarchical graph drawing techniques.

The Sugiyama method proceeds in four distinct stages:

1. Make the graph acyclic by reversing a minimal set of edges.

2. Assign node subsets to layers such that all edges in the acyclic graph are downward pointing. Edges which span more than two layers are split with dummy nodes placed on the intervening layers.

3. Find permutations of the nodes on each layer which minimise the number of edge crossings between layers.

4. Find horizontal positions for the nodes — without changing the ordering found in the previous step — such that edges are as straight as possible.

Each of these steps is an $\mathcal{NP}$-complete problem in its own right; but many, generally adequate, polynomial-time heuristic-based methods have been proposed. Bastert and Matuszewski [113, Pages 87–120] and also Di Battista et al. [11, Pages 265–301] provide in-depth surveys of these techniques.

The discrete space used for arranging the layers or ranks in the graph can be considered a direct analog of the extra half dimension defined for parallel planes in the definition of $2\frac{1}{2}$D visualisation in Chapter 2. In other words, traditional 2D application of hierarchical layout could be thought of as a $1\frac{1}{2}$D mapping. A simple $2\frac{1}{2}$D extension of $1\frac{1}{2}$D hierarchical layout is possible by extending the layers of nodes into parallel planes.

In [44] Dodson discusses a graph visualisation that is almost a force-directed/hierarchical layout hybrid. He draws graphs with the nodes constrained to three parallel planes. Sources are placed on the top plane, sinks are placed on the bottom plane and all other nodes are placed in the middle plane. A force-directed method is then used to arrange the nodes within the planes. Parker et al. demonstrate a similar scheme in [151] but with more conventional layer assignment in which the number of parallel planes is the maximum directed path length.

Such hierarchical-layout methods have a very $2\frac{1}{2}$D feel since the depth mapping for each node is discrete: that is, a mapping from an integer indicating the nodes' path length from a root. However, this is not a very direct mapping since it is dependant on the exact layer assignment algorithm chosen. In Chapter 4 a more direct depth mapping for graph visualisation is described.

Tree Layout

Arranging trees in two or three dimensions is generally an easier problem than arranging general graphs. It is always possible to find planar layouts for trees with relatively simple, linear-time algorithms, as demonstrated most famously by Tutte [189] and Reingold et al. [156]. However, theorists have found many difficult optimisation problems in tree drawing. Most commonly discussed examples are:

- Area and aspect ratio constrained tree drawings.
- Grid drawings (i.e. drawings with nodes placed at integer coordinates).

Surveys discussing a number of theorems concerning these and other tree drawing algorithms are found in textbooks [11, 113].

Most theoretical work on tree drawing algorithms concerns 2D drawings. Some notable work on 3D tree drawing considers symmetric embedding for trees. Hong et al. [98] give a linear time algorithm for drawing trees with a maximum number of symmetries in 3D. This is especially interesting because, as is described in Section 3.2.1, this problem is $\mathcal{NP}$-complete for general graphs.

Other notable work on 3D tree visualisation generally concerns efforts to provide compact representations and interaction metaphors for exploring very large trees of high internal node degree. Possibly the most famous of these is the *Cone Tree* system proposed by Robertson et al. [159]. Cone trees are 3D representations of trees with each level in the tree hierarchy drawn on its own parallel plane. Each internal node in the tree is placed on the appropriate plane and is marked by the tip of a partially transparent cone, with the children of the node placed around the circumference of the base of the cone on the adjacent plane. A user browses through the tree hierarchy by rotating the cones for each internal node to focus on particular children. A disadvantage of cone trees is that half the children of any cone will always be partially obscured, however they are a very compact way to represent trees with high degree internal nodes. According to the definitions in Section 2.2 cone trees can be considered $2\frac{1}{2}$D in their construction. The assignment of nodes to planes is *discrete* since there are an integer number of equally spaced planes; and the mapping of depth in the tree to spatial depth is both *direct* and *independent*.

An alternative geometry for arranging trees in 3D space is suggested by Hong et al. in [99]. Their POLYPLANE system arranges subtrees on planar facets of regular polytopes, see Figure 3.4. The system provides a very compact and aesthetically pleasing representation of large, high-degree

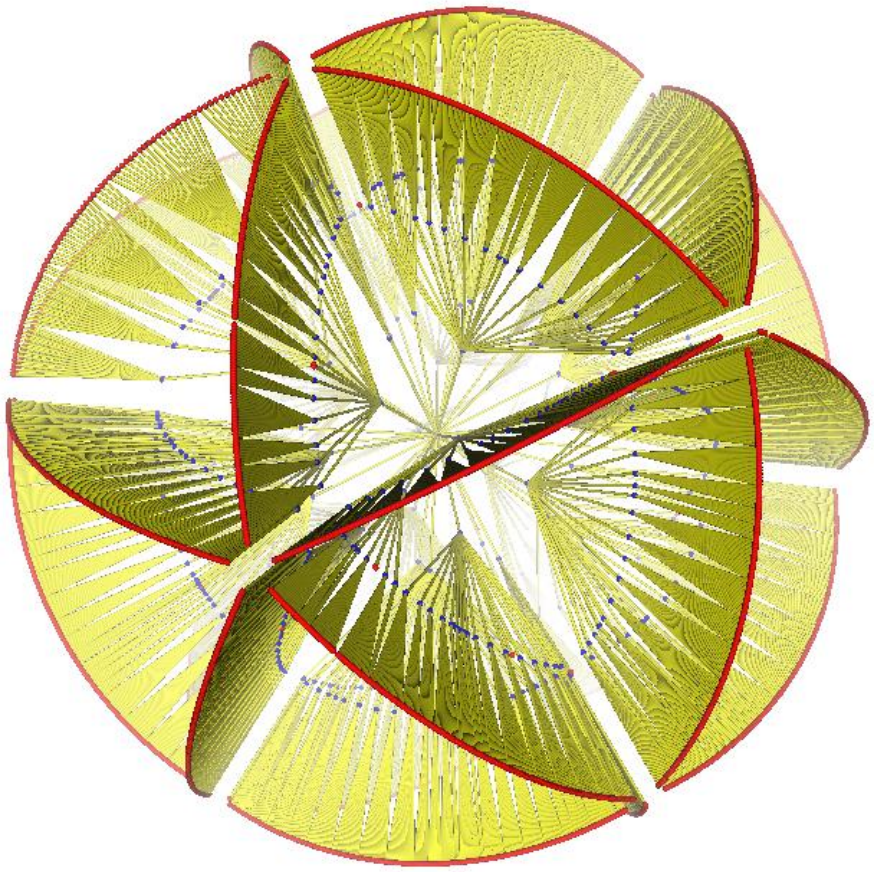


Figure 3.4: A sample tree visualisation generated by the POLYPLANE system. Courtesy Seokhee Hong.

trees; however, internal nodes can be somewhat obscured at the intersection of the planes.

Another effective approach to interactive visualisation of large trees is the 3D hyperbolic representation by Munzner et al. [141]. Although their system also handles directed graphs, it works particularly well with large trees such as the phylogenetic tree visualisation application that is described in Section 1.4.3.

3.3 Conclusion

Graph visualisation is introduced as a mapping from graph structure to a spatial arrangement (or layout) in 2D or 3D. The most common methods for 3D graph visualisation are discussed and their "$2\frac{1}{2}$D-ness" appraised; that is, consideration is given to how well the layout goals of the various methods fit with the $2\frac{1}{2}$D principles, defined in Section 2.2, of *directness*, *independence* and *discreteness*.

CHAPTER 4

Two and a Half Dimensional Stratified Graph Visualisation

> "Many called AutoCAD Version 2.5 (released 10 years ago - June 1986) a 2.5D system because you could produce a 2D profile and give it an elevation and thickness. It was a coincidence that the version number was 2.5; it had nothing to do with the fact that it was a 2.5D system." — Terry Wohlers, 1996 [201]

A method which assigns different semantics to the first pair of orthogonal spatial dimensions (call them *x* and *y*), compared to the third (*z*), is defined in Chapter 2 as "$2\frac{1}{2}$D visualisation". In Chapter 3 graph visualisation is introduced as a mapping for the nodes and edges in a graph to a visualisable space. In this chapter, these two concepts are brought together in defining "$2\frac{1}{2}$D graph visualisation". In Section 4.1 a definition for $2\frac{1}{2}$D graph visualisation is given and existing graph visualisations that can be classed as $2\frac{1}{2}$ dimensional are discussed. Then, in Section 4.2, *stratified graph visualisation* is introduced as a specific type of $2\frac{1}{2}$D graph visualisation for visualising evolving graphs and comparing related subgraphs. In Section 4.3, methods for visualising stratified graphs are examined and finally, in Section 4.4 stratified graph visualisation is compared with other approaches to visualising evolving graphs.

4.1 General Model for $2\frac{1}{2}$D Graph Visualisation

In Chapter 2 $2\frac{1}{2}$D Information Visualisation is defined as a visual mapping σ composed of two separate mappings $f \times g$ (see Equation 2.2) where f maps one dimension in the information space to the z-axis in a visual space and g provides a mapping to the x and y axes and other graphical attributes. In Chapter 3 a mapping Γ_k (see Equation 3.1) is defined for a logical graph model to a visualisable space of k dimensions. A broad definition of $2\frac{1}{2}$D graph visualisation is easily obtained

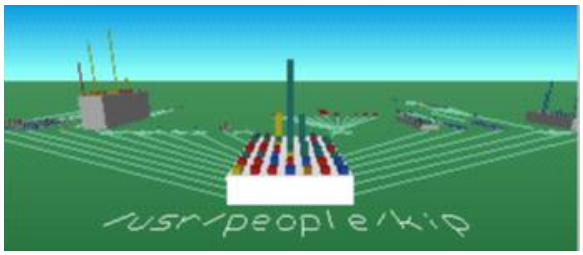


Figure 4.1: "This is a Unix system... I know this!" The SGI file system viewer featured in the movie *Jurassic Park*.

by substituting Γ for g.

$$\begin{aligned} \sigma &= (f, \Gamma) \\ f &: A \times (V \cup E) \mapsto \mathbb{N} \\ \Gamma_2 &: (V \times A_v, E \times A_e) \mapsto ((\mathbb{R}^2 \times \Theta_v), (C^3 \times \Theta_e)) \end{aligned} \tag{4.1}$$

Essentially, this means that two dimensions are used for graph layout, and position in the third dimension — or depth mapping — is used to represent some other attribute A associated with nodes or edges.

A number of different graph visualisations fit loosely into this definition. Possibly the simplest way to extend a 2D graph visualisation into the third dimension is to combine the *cityscape* metaphor with a drawing of a graph in the plane. A classic example of this technique was the SGI File System viewer, FSN[1], made famous by the Jurassic Park movie (see Figure 4.1). The directory hierarchy is represented by a tree drawing on a plane. Nodes are drawn as boxes where the height of the box rising out of the plane indicates the total size of the files in the directory. In terms of the model above, f maps the directory size attribute to the z-axis. Other examples of the cityscape metaphor for $2\frac{1}{2}$D graph visualisation include a visualisation of bibliographic networks by Chen[31] and an interactive system for browsing attributes associated with network nodes by Chuah et al. [32]. Some algorithms and a general overview of the paradigm is given by Keskin [115].

A metaphor closely related to the cityscape graph visualisation involves overlaying a graph drawing with a landscape showing elevations at node points. For example, Brandes and Willhalm [23] used this technique in bibliographic networks (see Figure 4.2) and Davidson et al. [38] visualise microarray expression data using a landscape to emphasise clusters.

Another closely related metaphor uses the elevation of nodes to better show graph structure.

[1] See `http://www.sgi.com/fun/freeware/3d_navigator.html`.

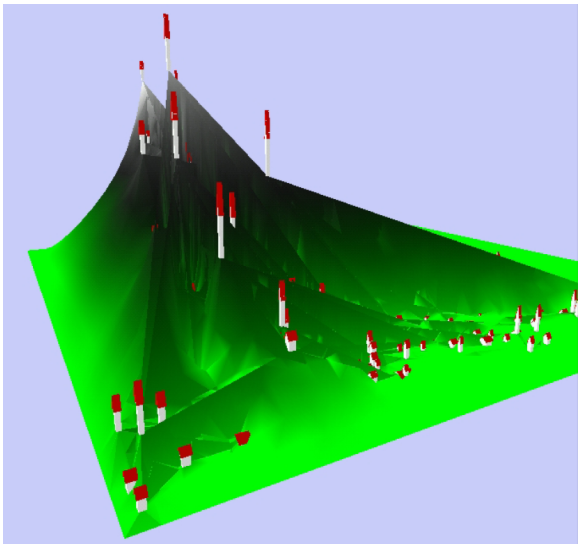


Figure 4.2: The landscape metaphor as used by Brandes and Willhalm

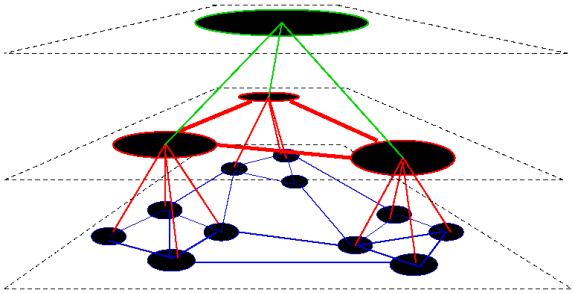


Figure 4.3: Visualisation of a clustered graph by Eades and Feng.

For example, Cone Trees [159], as discussed in 3.2.1, are created by raising the internal nodes in a radial tree drawing to a height proportional to their distance from the root. Similarly, Feng's cluster hierarchies [53] show the clustered structure of a graph by elevating nodes according to their depth in the cluster tree, see Figure 4.3. Another interesting example demonstrates a $2\frac{1}{2}$D drawing of the dependence graph of spreadsheet cells [172]. A spreadsheet has a natural 2D layout but in this visualisation cells are lifted into the third dimension to indicate the data flow.

In a prize winning entry to the 2004 IEEE InfoVis competition [2][2], $2\frac{1}{2}$D principles are used to explore a large citation network. Figure 4.4 shows the citation network with nodes representing papers and edges indicating a citation of one paper by another. Early attempts at visualisation revealed that the citation network was scale-free: that is, the node degree follows a power-law distribution [9]. Scale-free networks are notoriously difficult to visualise such that their structure is

[2]This competition entry represents the author's most recent application of $2\frac{1}{2}$D principles to graph visualisation and was completed in collaboration with Adel Aahmed, Colin Murray, Le Song and Ying Xin Wu.

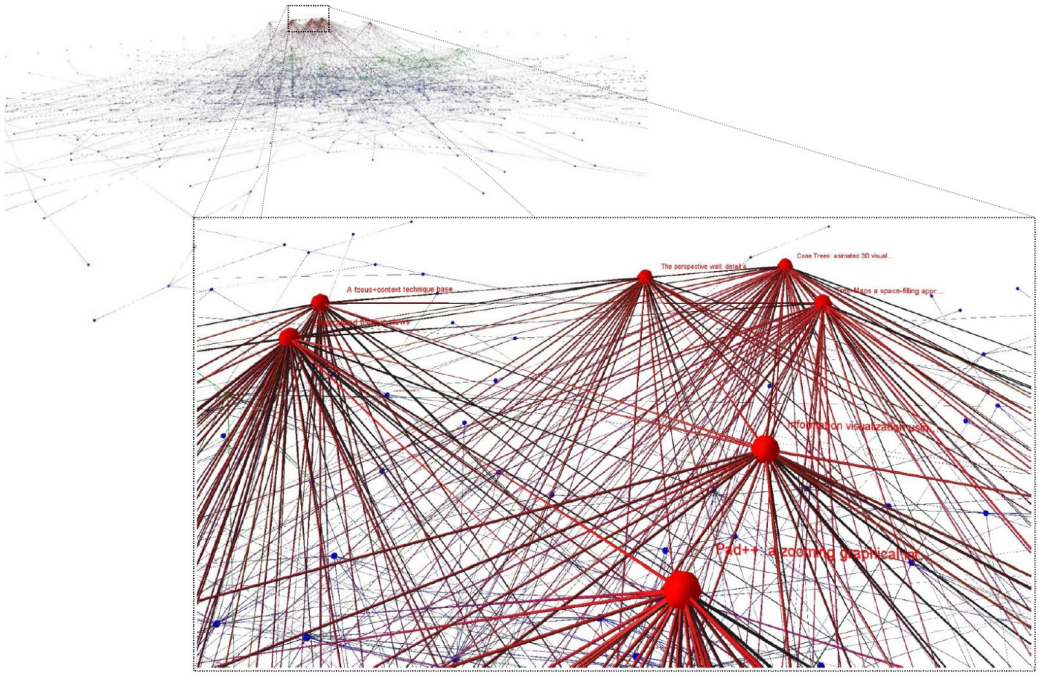


Figure 4.4: A $2\frac{1}{2}$D visualisation of the IEEE Information Visualisation citation network with an enlargement showing the most highly-cited "hub" papers.

clear [7]. The solution involved finding a 2D embedding using a standard force-directed algorithm and then mapping node elevation to discrete levels based on degree. The result shows the hubs clearly at the top of a volcano-shaped structure, with their lower-degree relatives spread out on the layers beneath.

The entry also included an alternative $2\frac{1}{2}$D visualisation in which the data was clustered, based on key-word and abstract information, into 11 research areas. All nodes in the citation network for each year of publication and research area, are replaced with a single node, and then all nodes in a given research area are linked into a chain with a second set of edges. The resulting graph is arranged using a force-directed layout algorithm with the nodes constrained to discrete levels corresponding to year of publication. The results are shown in Figure 4.5. The visualisation helps to elucidate the evolution of the eleven research areas and the changing relationships between them.

4.2 Stratified Graphs

In the above examples no constraints are placed on the way edges are arranged in the three dimensional space. That is, the edges could be any Jordan curve in three-space and could connect nodes with any z-coordinate. In this section a model with extra constraints is explored, the extra

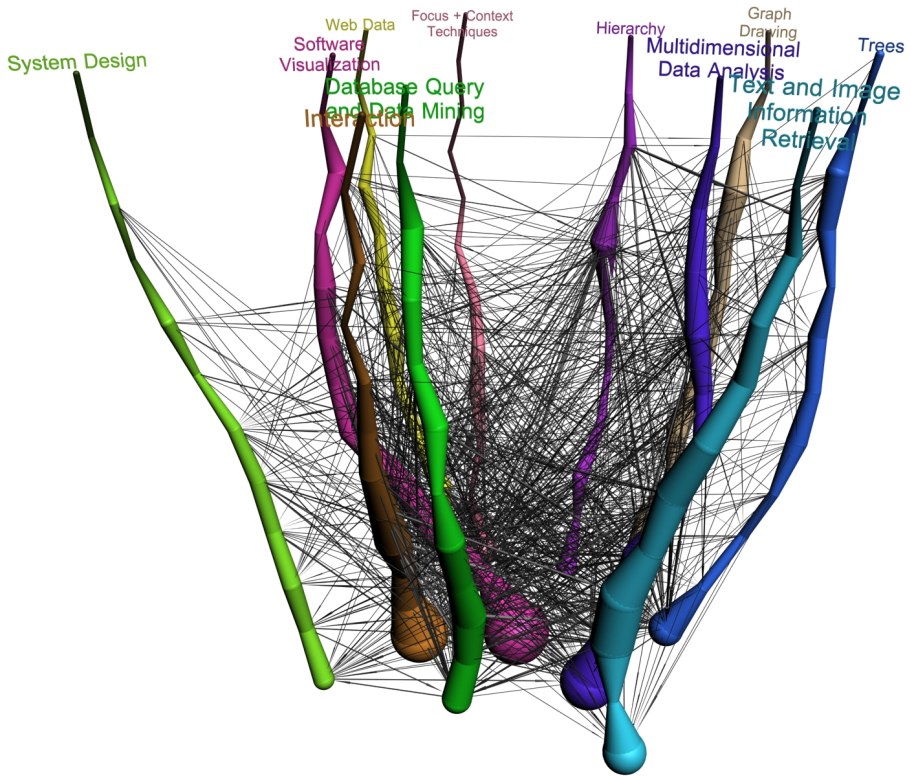


Figure 4.5: A $2\frac{1}{2}$D "worm" visualisation showing the evolution of research areas in the IEEE Information Visualisation citation network.

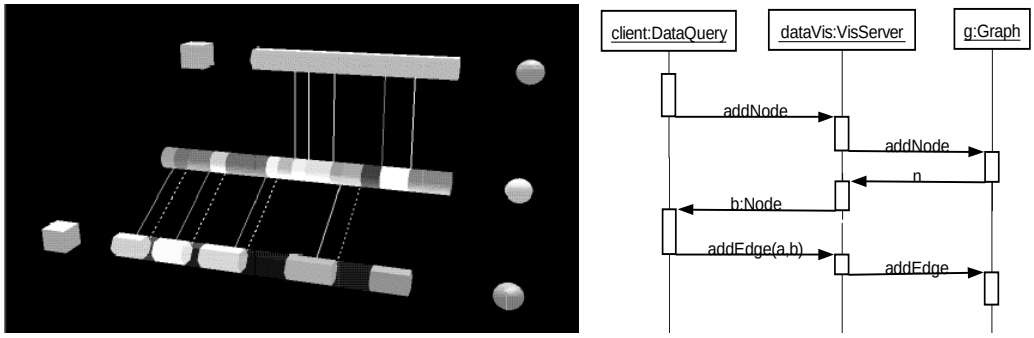
constraints being that:

- Edges must only connect nodes with the same z-coordinate.
- Edges must lie in a plane orthogonal to the z-axis and intersecting the z-axis at the same level as the nodes.

These extra constraints decrease the visual complexity of the visualisation and fit well with the constraints of human depth perception identified by Marr, see Section 2.1. Visualisations adhering to these constraints are called *stratified* graph visualisations, since the parallel planes or levels on which edges must lie resemble geological strata or levels of sediment in rock[3].

An early example of such a stratified graph visualisation is the representation by Koike [118] of parallel program execution, see Figure 4.6. In this representation nodes represent program components and edges represent messages being passed between these components. The graph is arranged in two dimensions and nodes are extruded into the z-dimension forming tubes. Edges are then

[3]Use of the term *stratified* also avoids use of the obvious, but already heavily overloaded, terms *layer* or *level*. Both of these terms are used to refer to the layering used in hierarchical layout, see Chapter 2.



(a) Visualisation of parallel program execution. Courtesy Hideki Koike.

(b) A UML sequence diagram

Figure 4.6: Parallel program visualisation with stratified graph visualisation and a UML sequence diagram.

placed at various levels orthogonal to the z-dimension where the height of the level relates to the time at which the message was sent. In many ways this is a literal extension of Unified Modelling Language (UML)[4] sequence diagrams into the third dimension, see Figure 4.6(b).

The visualisation by Koike is essentially a method for visualising an *evolving graph*: that is, a network with elements that exist only at certain periods in time. More formally, an evolving graph is defined as follows (see [63]):

Definition 4.2.1 *Given a graph $G = (V, E)$ with an ordered sequence of subgraphs $(S_G = G_1, G_2, \ldots, G_{\mathcal{T}})$ such that $\cup_{i=1}^{\mathcal{T}} G_i = G$, let $S_{\mathcal{T}} = t_0, t_1, \ldots, t_{\mathcal{T}}$ be an ordered sequence of time instants. The system $\mathcal{G} = (G, S_G, S_{\mathcal{T}})$, where each G_i is the subgraph in place during $[t_{i-1}, t_i]$, is called an* evolving graph.

Stratified graph visualisation is easy to define in terms of such an evolving graph:

Definition 4.2.2 *Each of the subgraphs $G_i \in S_G$ of an evolving graph G, are mapped to a stratum or plane perpendicular to the z-axis, at a height proportional to the time interval $[t_{i-1}, t_i]$ corresponding to the existence of G_i. In terms of the mappings defined in (4.1), f is the mapping of the interval $[t_{i-1}, t_i]$ to a discrete level on the z-axis and Γ_2 is a mapping of the elements of G_i to positions within the plane on that level.*

An application involving a visualisation of a graph that changes over time — a network of share trades by a fund manager — is presented in detail in Chapter 5.

[4] See http://www.uml.org.

Although evolving graphs are usually considered as evolving over time — that is, f gives a partitioning of G according to a time attribute associated with each element $e \in V \cup E$ — there is no reason why partitions cannot be made according to any other variable associated with e. For example, Chapter 7 considers a set S_G of $\sigma = |S_G|$ graphs — phylogenetic trees — each an approximation of an underlying graph whose exact structure is unknown. Each (sub)graph $G_i \in S_G$ has an associated attribute giving the probability of the approximation, and it is this variable that is used to partition the union graph $\cup_{i=1}^{\sigma} G_i = G$ and map to strata.

A more generic definition of stratified graph visualisation, that does not consider the various strata as necessarily corresponding to time intervals but to any domain specific partitioning of the graph, is as follows:

Definition 4.2.3 *Given a graph $G = (V, E)$, S_G defines a sequence of σ subgraphs $(S_G = G_1, G_2, \ldots, G_\sigma)$ such that $\cup_{i=1}^{\sigma} G_i = G$. A set $\mathcal{S}$ of σ strata is defined along with a mapping f for each of the subgraphs $G_i \in S_G$ to a stratum. The mapping Γ_2 maps the elements of G_i to positions within each planar stratum. Note that an element $e \in V \cup E$ may appear on more than one stratum: that is, for each e there is a set of strata $\mathcal{S}_e \subseteq \mathcal{S}$ in which e is present.*

Also, where there is no inherent ordinal variable available in the underlying data, a mapping to a stratified graph can still be found, but analysts are free to find a partitioning and ordering that suits visualisation for practical or aesthetic reasons relevant to the application domain. In Chapter 6 such an application (visual comparison of metabolic pathways) is considered. In this application a set of similar, but not identical, graphs are arranged into strata and an ordering for the strata is found such that graphs in adjacent strata are as similar as possible.

4.3 Visualisation of Stratified Graphs

The mapping of stratified graphs into a $2\frac{1}{2}$D visualisation is the main concern of the remainder of this thesis. As mentioned above the method is ideal for visual comparison of a set of graphs or following the evolution of a dynamic graph over time. These applications are explored more deeply with case studies in Chapters 5, 6 and 7.

First, however, methods for finding a mapping from the logical graph structure to a geometric structure in $2\frac{1}{2}$D — that is, layout algorithms for graphs in $2\frac{1}{2}$D — are explored. As described above, the mapping of nodes and edges to strata is a direct mapping based on a domain dependent

attribute. However, within each stratum there are any number of 2D layout strategies that can be employed to find an appropriate arrangement for the application.

Since stratified graphs are inherently related to evolving graphs, work on on-line or dynamic graph drawing methods such as [21, 22, 148, 102, 40] are somewhat relevant. However, the key difference between on-line and stratified graph visualisation is that in stratified graph visualisation the structure of all $G_i \in S_G$ is known before the layout process is begun. Brandes and Wagner [21], by contrast, give a generalised model for on-line graph layout inspired by a Bayesian approach to dealing with new graph structure. That is, following Bayes' rule, the layout for each successive graph depends only on its own layout aesthetics and so-called "anchoring-constraints" due to the previous graph layout.

Since a node may appear in multiple strata it would make sense to position the node at the same position in each stratum. Thus, the simplest approach to arranging stratified graphs is to find an arrangement for the entire union graph G in the plane and then simply extrude the plane into 3D, placing a glyph for each element at the same position in each stratum in which it appears. Such a näive approach may be taken with any 2D graph layout algorithm. A similar approach is taken by Diehl et al. [41] in their discussion of what they call "foresighted layout" for evolving graphs. In this discussion they consider the problem of animating a set of graphs by first finding a layout for the union graph. They also consider compacting the union graph by replacing nodes that do not appear in the same time intervals with a single union node. Thus, the same point in the plane may be shared by dissimilar nodes at different points in time. This aesthetic may work well for animations, but is not appropriate for visualising stratified graphs in $2\frac{1}{2}$D since the grouping of such, unrelated nodes, in a static visualisation would (a) make the overlapping nodes difficult to differentiate when viewed from above and (b) seem to imply a semantic grouping. In the following sections several modifications for standard layout methods are recommended, specifically designed for $2\frac{1}{2}$D visualisation of stratified graphs.

The graph visualisations shown in this section were generated with the WILMASCOPE graph visualisation system as described in Appendix B.

4.3.1 Force-Directed Layout

As discussed in Chapter 2, force-directed methods are the most popular tools for general graph layout. The discussion here therefore begins by looking at how force-directed methods can be applied to stratified graph layout.

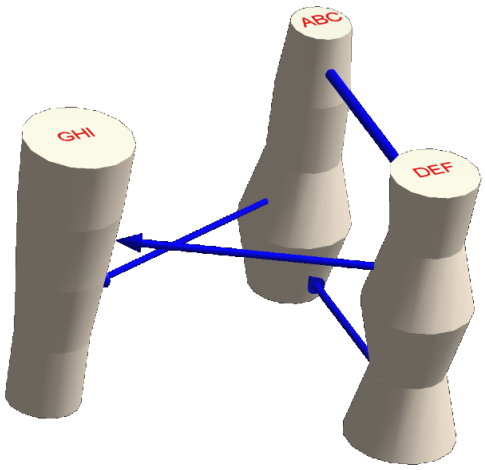


Figure 4.7: A small example demonstrating force-directed column layout with varying column widths to show additional data attributes

As discussed above, the simplest application of layout to a stratified graph is to find a 2D embedding of the union graph, then extrude into 3D, placing edges and nodes at the appropriate depths. The author's earliest work in this area used exactly this approach, see [47]. Contemporaneous work by Brandes and Corman [16] used a similar approach for visualising evolving discourse networks.

Visualisation using straight columns in this way yields some advantages:

- Intuitively, it is easy to track the position of a node across a number of layers if its position does not change.
- Change in column width and colour can be used to display additional attributes associated with each node and to emphasise how these attributes change from stratum to stratum. For example, in Figure 4.7 changing column width across strata is visible.
- As discussed in Chapter 2, force-directed layout displays graph structure such as clustering and outlying nodes extremely well. Using this simplest scheme for applying force-directed layout to stratified graphs shows this structure for the union graph. Figure 4.8 shows a top-down view of a stratified graph arranged in this way with outliers and clusters clearly visible. A detailed view of the central cluster is shown in Figure 4.9.

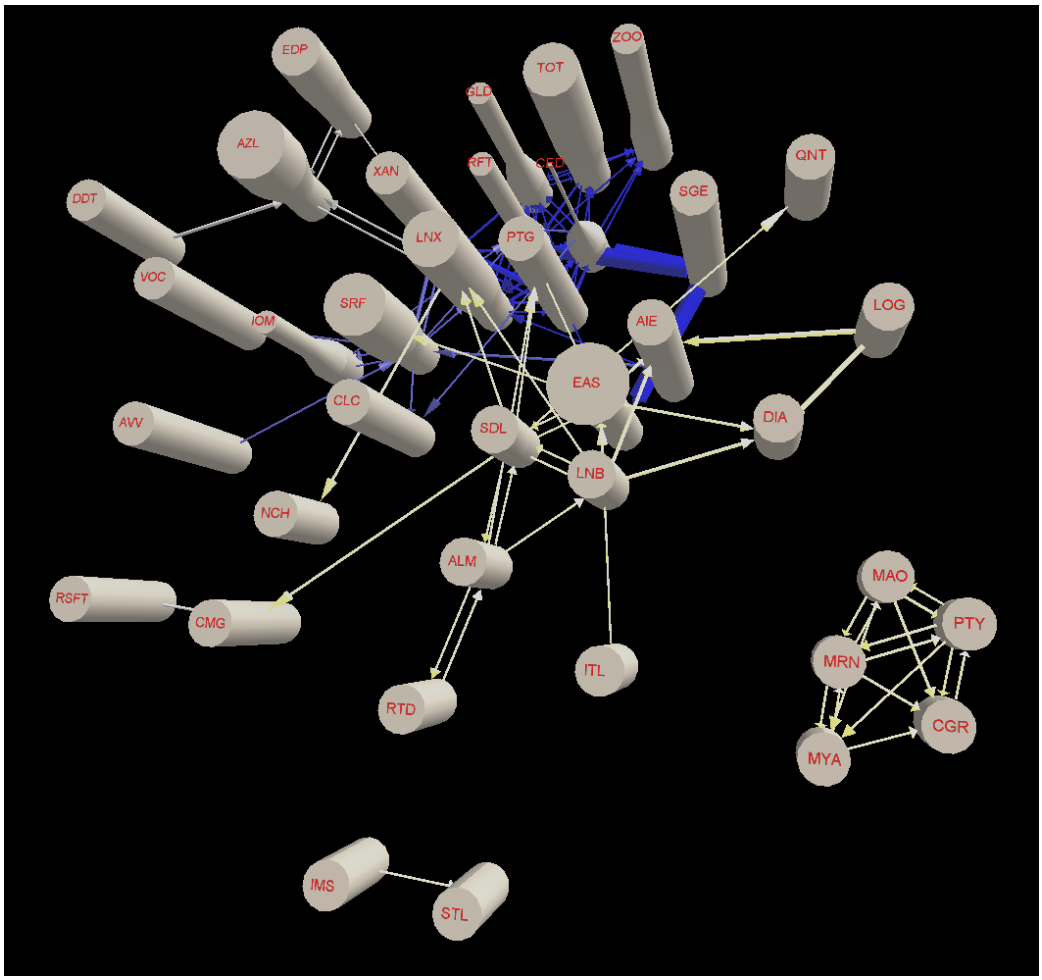


Figure 4.8: A larger example of force-directed column layout

However, the force-directed approach can be modified to further take advantage of the extension into $2\frac{1}{2}$D. Figure 4.10 shows the same small graph from Figure 4.7 but this time the columnar nodes have been allowed to bend to further minimise the stress from the springy edges. Each "worm" is constructed as a chain of nodes connected by virtual edges, see Figure 4.10(a). The position of the nodes is constrained such that nodes connected by a virtual edge must lie on adjacent strata. The final layout is then determined by a 3D force directed method. The worms are rendered by visualising each node in the chain as a sphere and the virtual edges as tubes with end radii matching that of the spheres for each of the end nodes, see Figure 4.10(b).

Allowing the columns to bend in this way allows the layout to accentuate conditions such as changing centrality and clustering over time. However, the danger is that it could become more difficult to see changes in column width and may contribute to general confusion: that is, the bendy columns are generally "busier" and may diminish some of the advantages of $2\frac{1}{2}$D since the

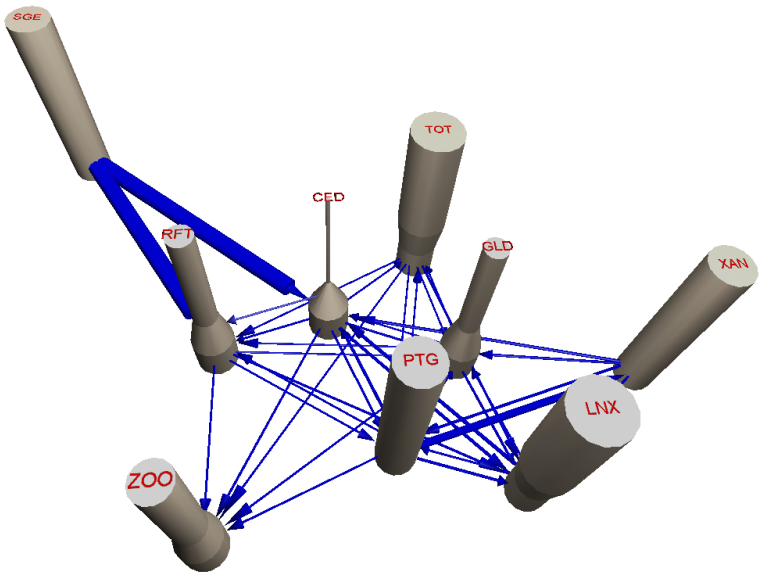


Figure 4.9: A detailed close-up of the clustered portion of the graph shown in Figure 4.8

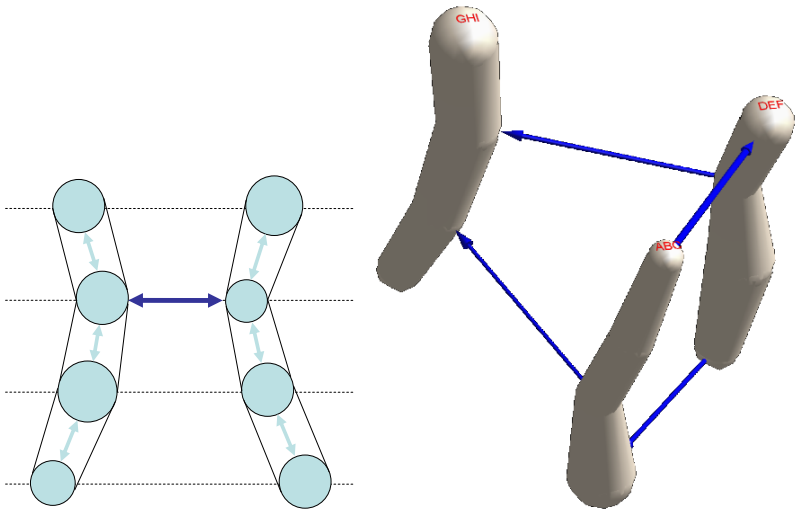
orientation of the z-axis becomes less clear. Figure 4.11, shows a larger example of the "worm" layout where both the advantages (such as temporal clustering) and disadvantages (such as increased complexity) can be seen.

In Chapter 4 an application of these force directed layout strategies to portfolio visualisation is discussed, along with further discussion of the advantages and disadvantages of different layout schemes.

4.3.2 Hierarchical Layout

As with force-directed layout, the easiest way to visualise a stratified graph with an hierarchical layout method (see Section 3.2.1), is to find a 2D embedding of the union graph and then extrude into the 3^{rd} dimension. Again, the extrusion is simply a matter of assigning depths to nodes and edges according to the depth mapping f (Equation 4.1).

However, the extension into 3D may benefit from an hierarchical layout algorithm that is aware of the stratified structure of the graph. For example, as described in Chapter 2, a key step in hierarchical layout methods is finding an arrangement of nodes within each layer that minimises the number of crossings between edges. Figure 4.12 shows how crossings of edges not present in the same stratum can be resolved by 3D rotation or stereo depth perception. It follows that the crossing minimisation step in an hierarchical layout method for stratified graphs can treat crossings that



(a) Construction of worms (b) Result

Figure 4.10: A small example of worm layout

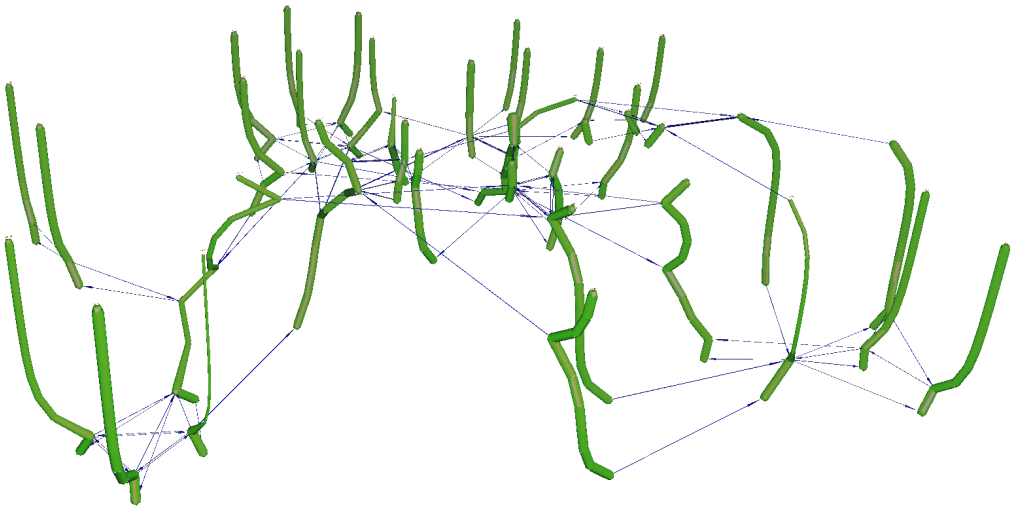
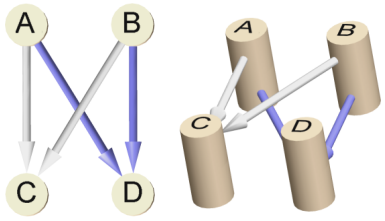


Figure 4.11: A larger example of the worm force-directed layout



(a) Top view (b) Rotated view

Figure 4.12: Crossings between edges in different strata can be resolved by 3D rotation.

occur in the same stratum in a different way to crossings that only occur between edges that do not exist in the same stratum.

Crossing Reduction

As mentioned in Chapter 2 finding the minimum number of crossings in a k-layered graph is an $\mathcal{NP}$-complete problem but heuristics may be applied to find a reasonable solution in good time. A first step in simplifying the problem — used in most practical methods — is to reduce crossings one layer at a time in a layer-by-layer sweep approach. That is, looking first at the top-most layer L_0, the method tries to minimise the crossings between edges from that layer to the next layer L_1 by permuting the node positions in L_1. This is known as the *one-sided crossing minimisation problem*, since only the nodes in one of the two layers being considered are permuted, for example see Figure 4.13. Then, examining the new permutation of L_1 and the next layer L_2, a permutation of L_2 is found that reduces the crossings between edges from L_1 to L_2. This process sweeps through all layers until a new permutation of L_k has been found. A reverse sweep is then applied from L_k back to L_0. The sweep repeats, forward then reverse, until the total number of crossings in all layers converges.

Unfortunately, the one-sided crossing minimisation problem is still $\mathcal{NP}$-complete [57], so heuristics are commonly applied to find approximate solutions. The one-sided crossing minimisation problem, viewed in isolation from the larger multi-layer crossing problem, considers a bipartite graph $G_b = (V_1, V_2, E)$ (see Section 3.1) where V_1 and V_2 are the sets of nodes on adjacent layers. The permutation π_1 of nodes V_1 in the first layer is fixed while the permutation π_2 of the nodes V_2 in the second layer may be changed. Exact and heuristic methods for this one-sided crossing minimisation problem are based on the *crossing matrix* $(c_{ij})_{i,j \in V_2}$, in which an entry c_{ij} corresponds

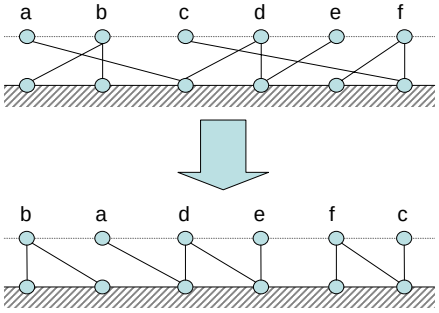


Figure 4.13: The one-sided crossing minimisation problem. The permutation of nodes on the bottom layer is fixed while nodes on the top layer may be reordered to reduce edge crossings.

	a	b	c	d	e	f
a	-	2	0	0	0	0
b	0	-	0	0	0	0
c	1	2	-	2	1	1
d	0	4	2	-	0	0
e	1	2	0	1	-	0
f	2	4	0	4	2	-

Table 4.1: Crossing matrix for the two-layer graph in Figure 4.13. Each entry c_{uv} gives the number of crossings induced by placing the node u corresponding to the row before the node v corresponding to the column.

to the number of crossings between the two layers caused by edges incident to nodes $i, j \in V_1$, if i is placed to the left of j: that is, $\pi_2(i) < \pi_2(j)$. Table 4.1 is an example of such a crossing matrix. Let $\delta_{ij}^k = 1$ if $\pi_k(i) < \pi_k(j)$ or 0 otherwise and let $N(u) \subseteq V_1$ be the set of nodes connected by edges to $u \in V_2$. Then:

$$c_{ij} = \sum_{k \in N(i)} \sum_{l \in N(j)} \delta_{lk}^1 \tag{4.2}$$

A strata-aware crossing matrix is defined by multiplying the contribution of a pair of edges to an entry c_{uv}, with the number of times that these edges are present in a common stratum. Note that this results in a weighted crossing matrix in which crossings are counted individually for each pair of edges, and that this weighting scheme is different from, say, assigning weights to edges and multiplying these if edges cross. More formally, Equation 4.2 is adapted to the new stratified definition:

$$c_{ij} = \sum_{s=1}^{\sigma} \sum_{k \in N_{G_s}(i)} \sum_{l \in N_{G_s}(j)} \delta_{lk}^1 \tag{4.3}$$

A näive algorithm for computing c_{ij} visits every node in $N(j)$ for every $N(i)$ in time $O(|N(i)||N(j)|)$.

A better method for computing c_{ij} in time $O(|N(i)|+|N(j)|)$ is given in Chapter 7. An expression for the number of crossings for an entire two layer problem in terms of c_{ij} — from Jünger and Mutzel [107] — is as follows:

$$\text{cross}(\pi_2) = \sum_{i=1}^{n_{2-1}} \sum_{j=i+1}^{n_2} c_{ij}\delta_{ij}^2 + c_{ji}(1-\delta_{ij}^2) \tag{4.4}$$

Currently the fastest algorithms [10] for counting all the crossings in conventional two-layer graphs do so in time $O(|E|\log|V_{small}|)$ where E is the set of edges and V_{small} is the smaller of the two sets of nodes. Adapting to consider edges from different strata by simply considering σ separate two-layer graphs does not significantly reduce or increase this running time.

Since most crossing reduction methods based on the crossing matrix are oblivious to the definition of its entries, they can also be applied in the stratified case.

There are a number of heuristics for reducing the number of crossings in a two layer problem which may be adapted to stratified graphs. A brief description is given here of those heuristics tested with stratified graphs. For a complete survey see [11, 113]. The *greedy switch* or *adjacent exchange* heuristic is the most straight-forward — a simple scan across all neighbouring nodes $u, v \in V_2$, exchanging their positions if doing so leads to a reduction in crossings (see Algorithm 1).

Algorithm 1 Adjacent exchange crossing reduction

procedure $adjacentExchange(G)$

Require: $G = (V_1, V_2, E)$ is a bipartite graph with node sets V_1 and V_2 arranged on adjacent layers.

```
repeat
    r ← 0 {r accumulates the number of crossings removed in a pass}
    for i ← 1 ... |V2| − 1 do
        j ← i + 1
        if c_ij > c_ji then
            exchange the positions of nodes at i and j
            r ← r + c_ji − c_ij
        end if
    end for
until r = 0
```

The *median heuristic* [57] is conceptually very simple. Each node $u \in V_2$ is placed at the horizontal position of its middle or *median* neighbour: that is, if $v_1, v_2, \ldots, v_j \in N(u)$ is the set of neighbours of u sorted by their position π_1, then set $\pi_2(u) = \pi_1(v_{\lceil \frac{j}{2} \rceil})$. Complexity arises when choosing the exact neighbouring position to use when j is even and there are two middle neighbours

from which to choose: that is, at $\pi_1(\frac{j}{2})$ and $\pi_1(\frac{j}{2}+1)$. The definition above implies defaulting to the left of these two, but more complex strategies are possible. The *weighted median* approach suggested by Gansner et al. [76] is used here. This heuristic chooses central nodes on the more densely packed side.

To compute the stratified graph layouts shown in this thesis, the open-source program DOT[5], which uses a median heuristic coupled with an adjacent-exchange post-processing step [76], was adapted. New permutations generated by these heuristics are rejected if they lead to an increase in edge crossings according to the modified crossing matrix, as described above.

Since the median heuristic method does not consider crossings until after a permutation is generated it was felt that it might not be readily compatible with the new definition of c_{uv}. As an alternative, crossing minimisation based on the Integer Linear Programming (ILP) approach suggested in [107], was also implemented. The ILP method directly uses $c_{uv} - c_{vu}$ as the coefficients of the variables of the cost function to find an exact solution for each one-sided crossing problem in the sweep.

Eades and Kelly [54] suggest that the calculation of exact solutions to the one-sided crossing minimisation problem would be useful in evaluating the various heuristic solutions. At the time of their report, however, the computational complexity of the problem seemed prohibitive. Later work by Jünger and Mutzel [107] demonstrated an Integer Linear Programming approach that made such an evaluation possible for small two-layer graphs (< 100 nodes per layer). Their definition of the Integer Linear Program, or Linear Ordering, for the one-sided crossing minimisation problem begins with a restatement of $\text{cross}(\pi_2)$ — from Equation 4.4 — to isolate δ^2_{ij}:

$$\text{cross}(\pi_2) = \sum_{i=1}^{n_{2-1}} \sum_{j=i+1}^{n_2} (c_{ij} - c_{ji})\delta^2_{ij} + \sum_{i=1}^{n_{2-1}} \sum_{j=i+1}^{n_2} c_{ji} \tag{4.5}$$

Setting $n = n_2$, $x_{ij} = \delta^2_{ij}$ and $a_{ij} = c_{ij} - c_{ji}$ the linear ordering problem is then:

[5] Available at http://www.graphviz.org/.

$$\text{minimise} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} a_{ij} x_{ij} \tag{4.6a}$$

$$\text{subject to} : 0 \leq x_{ij} + x_{jk} - x_{ik} \leq 1 \text{ for } 1 \leq i < j < k \leq n \tag{4.6b}$$

$$0 \leq x_{ij} \leq 1 \text{ for } 1 \leq i < j \leq n \tag{4.6c}$$

$$x_{ij} \in \mathbb{Z} \text{ for } 1 \leq i < j \leq n. \tag{4.6d}$$

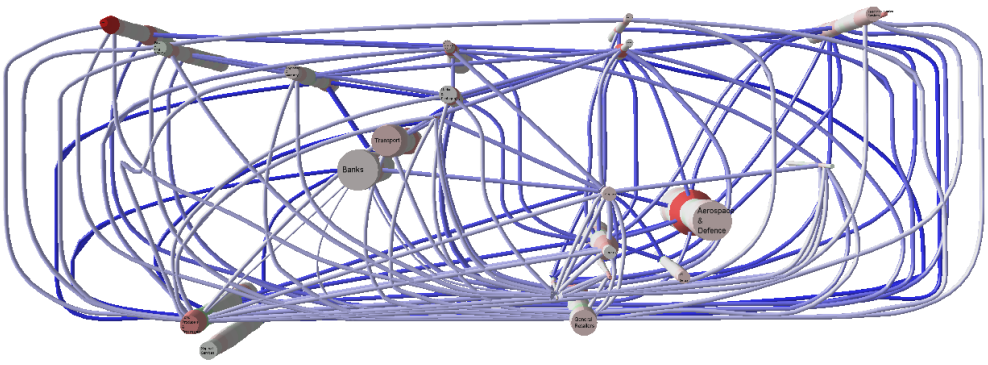
Thus, for each pair of nodes $v_i, v_j \in V_2$, $x_ij = 1$ indicates that $\pi_2(v_i) < \pi_2(v_j)$ and the vector $\vec{x}$ gives a complete ordering for the nodes in V_2. The so-called "3-cycle" constraints in (eq. 4.6b) ensure that $\vec{x}$ corresponds to a legitimate permutation of the nodes in V_2. The adaptation of this linear program to stratified crossing minimisation, by directly substituting the new definition for c_{ij} (eq. 4.3) is trivial. Obviously, the weighted crossing minimisation variant is at least as difficult as standard crossing minimisation.

The addition of this ILP crossing minimisation method to DOT was achieved using the GLPK linear programming libraries[6]. In experiments it was found that the median heuristic outperformed the ILP approach, because it tends to achieve a reasonable global solution which the subsequent adjacent-exchange is able to improve according to the modified definition of c_{uv}. Moreover, it is significantly faster than the *branch and cut* algorithm for solving the ILP.

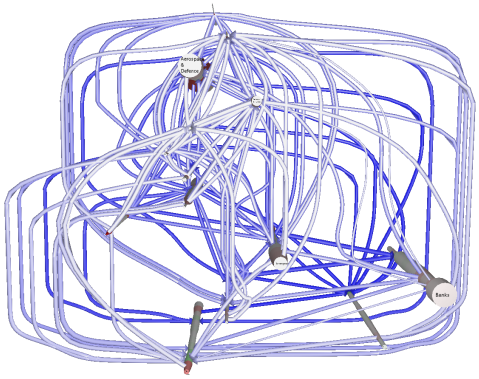
Horizontal Node Positioning

Another refinement to the Sugiyama method for stratified graphs in $2\frac{1}{2}$D is possible in the horizontal positioning step. When routing the edges in $2\frac{1}{2}$D, dummy nodes on different strata can be allowed to overlap. In DOT's implementation, an auxiliary graph is created in which edges of an arbitrary minimum length are inserted between adjacent nodes (and dummy nodes) in each layer to keep them separated. DOT then uses the same network simplex method used in its vertical ranking of the original graph to find an optimal horizontal arrangement of the nodes in this auxiliary graph — see [76] for details of this method. The length of auxiliary edges between adjacent dummy nodes that do not coexist on the same stratum, is therefore set to zero, thus allowing such nodes, and hence edges, to overlap. This led to a significant improvement in aspect ratio of the final layout. For example in the test described below the improvement in width/height was from 2.3 to 1.3 (or

[6] GNU Linear Programming Kit — http://www.gnu.org/software/glpk/glpk.html



(a) without



(b) with

Figure 4.14: Fund manager flow graph used in tests viewed from above with and without strata-aware placement.

approximately 40%) in the densest test cases.

Results

Table 4.2 summarises the results of running the original and modified DOT program on a reasonably complex $2\frac{1}{2}$D graph with 12 strata, 14 nodes and 72 edges. The example graph models fund manager movements between market sectors, as described in 5, and is much more dense than would probably be usable in practice, but it is a useful extreme test — see Figures 4.14 and 4.15. The various layout methods tested are:

Heuristic No Strata — The original DOT median/adjacent-exchange crossing reduction and horizontal positioning

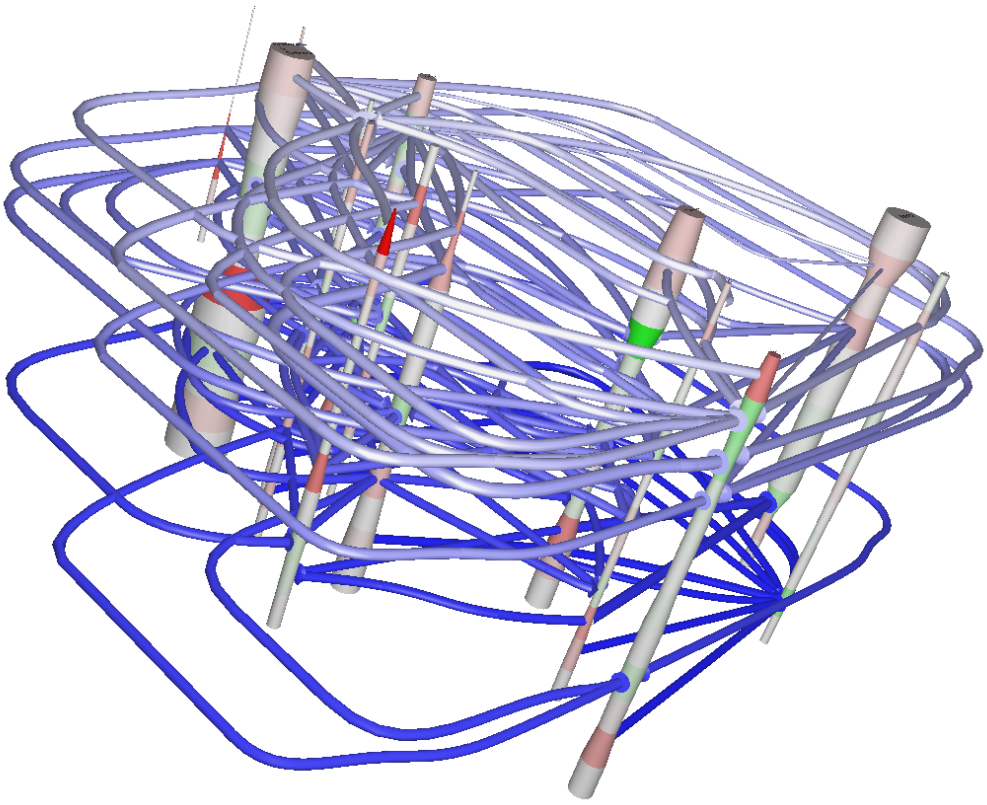


Figure 4.15: An oblique view of the test graph arranged with strata-aware placement.

Heuristic Sep. Strata — The modified median/adjacent-exchange crossing reduction method using the stratified calculation for c_{uv} and using the stratified horizontal node positioning as described above

ILP No Strata — ILP crossing reduction as described in [] with the standard DOT horizontal positioning

ILP Sep. Strata — ILP crossing reduction using the stratified calculation for c_{uv} and stratified horizontal positioning

There are three different crossing counts listed:

No Strata — Duplicate edges on different strata are counted as a single edge with the original definition of c_{uv}

All Strata — Duplicate edges are all counted with the original definition of c_{uv}

Separate Strata — The definition stratified definition of c_{uv} as above.

	Total Crossings			
Method	*No Strata*	*All Strata*	*Separate Strata*	*Width/Height*
Heuristic No Strata	173	346	40	2.35
Heuristic Sep. Strata	238	468	36	1.29
ILP No Strata	202	411	57	2.47
ILP Sep. Strata	210	421	50	1.01

Table 4.2: Results of using the various methods to arrange a $2\frac{1}{2}$D graph with 12 strata, 14 nodes and 72 edges

4.3.3 Tree Layout

In Chapter 7 an application for visualisation of a set of phylogenetic trees is presented. The layout style used is quite specific to the application. Lengthy discussion of stratified tree comparison method, and algorithmic contributions to this method, are therefore presented as part of the case study report in that chapter.

4.4 Conclusion and Comparison with Other Approaches

The method usually considered for visualising evolving graphs (Definition 4.2.1) in most discussions of dynamic or on-line graph layout (as discussed in Section 4.3) is simply to display an animated sequence of the subgraphs corresponding to each time instant in increasing time order. The challenge is to draw the viewer's attention to structural changes in successive graphs. This is achieved either by minimising layout changes in successive graph drawings or alternately, as discussed by Friedrich [72, 71], to animate the transition between successive drawings smoothly, by inserting intermediate frames of animation. In either case, when comparing a pair of successive graphs, the viewer must rely upon memory of the previous drawing to identify differences. The model of the previous graph that the viewer is required to keep in memory is commonly referred to in graph visualisation literature as the *mental map* [139]. The $2\frac{1}{2}$D stratified graph visualisation method has the advantage that it does not rely on the viewer's memory to the same degree as they compare differences in a set of graphs. Rather, the entire set of graphs is displayed concurrently and the user is able to compare and contrast using *external cognition* [167] — a concept that is central to the motivation for data visualisation as discussed at length by Card et al. [28, Pages 1–34].

However, there are also disadvantages to the stratified graph visualisation approach:

1. Dynamic visualisation using animation to step through a sequence of graph drawings can also be applied to a sequence of 3D graph visualisations while the stratified

approach requires the graph in each stratum to be drawn in the plane.

2. A graph on an internal stratum in a stratified graph stack can be obscured by the strata above and below and is generally not as easy to see as an individual graph drawing in a dynamic sequence.

3. The scalability of stratified graph visualisation to sets of many graphs is limited for two reasons:

   - for large sets of graphs, the global layout of the union graph will most likely be far from optimal for individual subgraphs;
   - the problem of outer strata obscuring the view of internal strata, described in (2) above, becomes increasingly serious as more strata are added.

A possible solution to problems (2) and (3) is a hybrid of the dynamic graph visualisation and stratified graph visualisation approaches. In later chapters two such hybrid solutions to these problems are discussed. First, in Chapter 5 a cross-sectional viewer is introduced. This allows users of a stratified graph visualisation system to obtain a 2D view of the graph on an individual stratum. By interactively stepping through each of the strata in the $2\frac{1}{2}$D stratified view, they effectively obtain a conventional animated sequence showing the dynamic graph changes. Secondly, in Chapter 6 an approach is introduced in which browsing a large set of graphs is facilitated with a stratified graph visualisation showing only a subset of the available set of graphs at one time. The user can interactively change the subset of graphs shown in the stratified visualisation, effectively showing a *dynamic sequence of stratified graph visualisations*.

CHAPTER 5

Fund Manager Movement

"The purpose of computing is insight, not numbers." — R. W. Hamming [88]

This chapter presents the first case study using the two-and-a-half-dimensional graph visualisation metaphor. Two related, yet distinct, styles of visualising fund manager holdings as they change over time are introduced. As well as sharing a common application, both the visualisations use a depth mapping for time. However, they have slightly different goals.

The first visualisation style, implemented in the PORTFOLIOSPACE EXPLORER system, is intended to provide an overview of a large number of different fund managers' holdings in a single display. It also provides several interactive features that aid data-mining activities, allowing analysts to "drill down" through the data, filtering to find portfolios that display interesting activity. The technique uses methods from multidimensional scaling for the mapping Γ defined in Eq. 4.1. The aim of multidimensional scaling is to find an embedding of the distance matrix between data points. This is analogous to graph drawing in that the distance matrix defines a weighted complete graph where each of the data points is a node.

The second visualisation style that is described in this chapter, is referred to as a PORTFOLIO COLUMNS visualisation. It is intended to provide a detailed view of the contents of one portfolio (or possibly a small number of aggregated portfolios). To achieve this, a graph representing movement between stocks or market sectors in the portfolio is defined. Then, the stratified graph visualisation techniques defined in Chapter 4 are used to allow analysts to explore these movement graphs.

A demonstration is then given, of how these two techniques can be coupled to provide an *overview-and-detail* style method for visualising a high-dimensional data-set with attributes that change over time. Section 1.4.1 provides some general background for fund manager holdings and capital markets visualisation. Before the new visualisation is described, however, a more in-depth introduction to the topic of fund manager performance analysis is provided. Much of the work

described in this chapter has been published in [47, 46] and in conjunction with David R. Gallagher in [49].

5.1 Introduction

Performance evaluation of fund managers using portfolio holdings data aims to improve understanding of the sources of returns attributable to stock selection and portfolio management. While aggregate fund returns are publicly reported to investors and market analysts, this seldom includes performance attribution from individual stock selection decisions. These can only be determined using portfolio holdings data at a very fine temporal granularity. Therefore, information about fund ownership is an important component in the analysis of investment manager skill, and a visualisation which helps to distill this information may be a useful tool for the analyst or investor.

However, performance analysis represents only one example of why visualisation of fund manager holdings is important. Pension fund trustees have fiduciary responsibilities to fund members (investors in the fund). Where pension fund mandates (or investment contracts) are delegated to fund managers, an understanding of how the portfolio is configured (including the cross-holdings of stocks across managers and how the fund is exposed to individual stocks at an aggregate level) is an important component of portfolio design. Quantitative techniques can be applied as a means of describing the risk attributes of the individual portfolios accessed by pension funds but such techniques may not always be relevant and may not explain complex situations. Visualisation techniques which show portfolio configuration properties more holistically represent an interesting tool for evaluation. They may also provide opportunities in the communication of technical aspects of portfolio management to investors who are not familiar and/or sophisticated in understanding quantitative portfolio analysis. For more information about performance analytics, see [60, 152, 24].

The rest of this chapter is structured as follows. Section 5.2 introduces the philosophy behind the $2\frac{1}{2}$D *overview-and-detail* visualisation strategy. Section 5.3 examines the overview visualisation provided by the PORTFOLIOSPACE EXPLORER and gives some background on multidimensional scaling and the Principal Component Analysis (PCA) method used. In Section 5.4 a novel system for visualising the results of the PCA dimension reduction is introduced. In Section 5.5 the graph-based detail visualisation — or PORTFOLIO COLUMNS view — is discussed and a formal definition of the underlying graph model, is given. Section 5.6 describes the layout algorithm employed to find an embedding of the graph for visualisation. Section 5.7 provides a walk-through of a

scenario demonstrating the use of the system in exploring a real data-set and feedback, received from showing the tool to fund manager research analysts, is discussed. The chapter concludes with Section 5.8 and some directions for future study are discussed.

5.2 Overview and Detail Portfolio Visualisation

The data-set inspiring this research is UK stock market registry data, where the changing portfolio holdings of all registered fund managers are held at a granularity of approximately one month. The data consists of some 3,000 portfolios composed of a selection of over 2,000 different stocks from 54 different market sectors. To obtain a useful picture of the movements within this data-set, at least twelve months' worth of this data must be available. Obviously, presenting such a large body of data to analysts in a visual form is going to challenge their "perceptual bandwidth". The approach presented decomposes this visual challenge into two subproblems, providing a basis for an "overview-and-detail" [28] visualisation design:

- To visualise the changing holdings of all fund manager portfolios in the data-set in order to obtain a broad picture of the entire market.
- To obtain a detailed view of changes in the holdings of just one, or a small number, of fund managers' portfolios.

The important theme in both of these visualisation tasks is visualising changes occurring through time. In the overview it should be possible to see who are the market leaders and followers. When visualising a small number of portfolios it should be possible to study the way fund managers prioritise their holdings to achieve a particular goal such as tracking or outperforming an index.

Both the overview and detail visualisation solutions presented share a common $2\frac{1}{2}$D aspect, in that the third dimension is used to represent discrete time intervals and the other two dimensions are used to represent more complex dimensional scaling or graph layout.

5.3 Overview Visualisation using Multidimensional-Scaling

The data-set used consists of several thousand equity holdings, each of which contains a selection of stocks from different companies belonging to different market sectors in the UK. A portfolio, or set of holdings in an individual fund, can be thought of as having a weighting in each of the available

stocks in the market. If there are 2,000 different companies in the market universe, one can think of these portfolios as being points in a 2,000-dimensional space. Alternatively, the weightings can be aggregated by market sector into a space of lower dimensionality. For example, Figure 5.1 is a chart showing the weighting of one portfolio across 50 market sectors at one particular point in time: that is, just one high-dimensional data point. The problem faced is to visualise the weightings of many portfolios, and to show how these weightings change over time.

Obviously, to make an intelligible visualisation, this high-dimensional space must be reduced to two or three dimensions. This process is called Multidimensional Scaling [14], the chief aim of which is to find a lower dimensional representation for a high-dimensional data-set. This representation should have relative Euclidean distances between the data points which preserve, as much as possible, relative proximities between the original high dimensional data.

As described in Chapter 3, if P is a set of points in m dimensions, then Multidimensional scaling computes a point $g(p)$ in k dimensions, where $m >> k$. A function g is required, such that for all pairs (p, q) of points in P, $c \cdot d_k(g(p), g(q))$ is approximately the same as $d_m(p, q)$ for some scale factor c. Here d_k and d_m are, most commonly, both Euclidean distance functions in k and m dimensions respectively — that is:

$$d(p,q) = \sqrt{\sum_{\alpha=1}^{m} (p_\alpha - q_\alpha)^2}$$

Typically, multidimensional scaling is achieved by minimising a stress function for the entire data-set. The first step is to compute the covariance matrix $X = (X_{ij})$ where, if $P = \{P_i : 1 \leq i \leq n\}$, then $X_{ij} = d_m(P_i, P_j)$. Then $g(p_i)$ is computed to minimise the stress function:

$$\sigma = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (X_{ij} - Y_{ij})^2$$

where $Y_{ij} = d_k(g(p_i), g(p_j))$.

Usually an iterative approach, such as steepest descent, is used to find Y such that σ is minimised. For n data points such a method requires at least $O(n^2)$ operations per iteration to compute the distances between each pair of points. In the data-set considered in this chapter there are in the order of 3,000 portfolios with a data point for each of at least 12 monthly samples. Therefore, the multidimensional-scaling approach needs to be able to scale up to around 36,000 data points with a processing time that will allow a user to interactively explore the data-set. For such a large data-set,

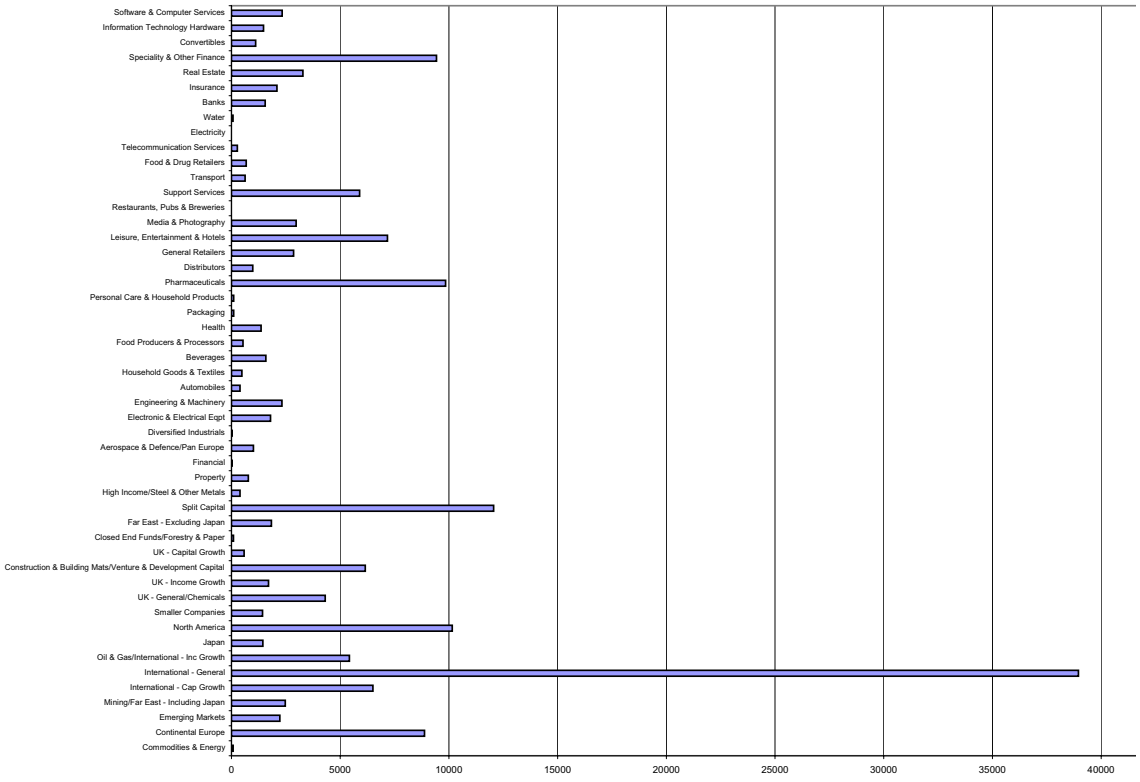


Figure 5.1: The weighting of a portfolio by market sector at one point in time. The horizontal axis is the total value invested.

where the dimensionality of the data is significantly lower than the number of data points, it is more practical to use a classical multidimensional scaling method based on Principal Component Analysis [59]. The complexity of PCA is based on the dimensionality of the data rather than the number of points: that is, $O(m^2 + n)$.

The aim of PCA is to find the axes of greatest variance (*principal components*) through the m-dimensional data. The data can then be projected onto the plane defined by two such axes to obtain a two-dimensional representation which captures this variance. Of course, this does not guarantee to minimise σ, but hopefully an adequate visualisation is obtained in reasonable time. Koren and Carmel explore PCA based multidimensional scaling for visualising high dimensional data in [119].

The approach used to find the principal components in the PORTFOLIOSPACE EXPLORER is fairly standard. There are m possible stocks or market sectors, l different portfolios, t different temporal samples for each portfolio and therefore $n = t \cdot l$ data points. The first step is to place all of this data in an $m \times n$-matrix A such that the columns are the dimensions and the rows are the

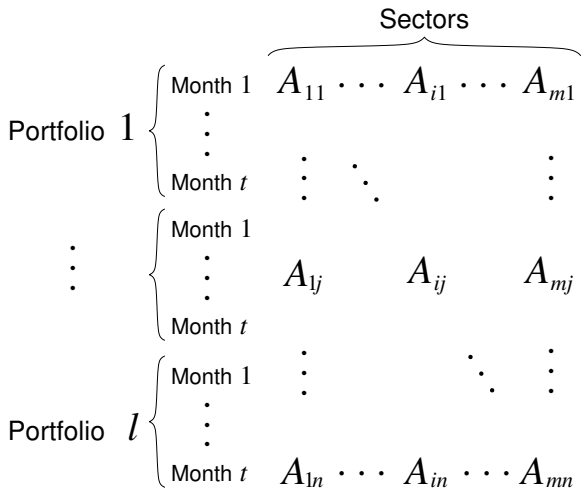


Figure 5.2: The construction of the $m \times n$ data matrix A for m market sectors and l portfolios, each with t monthly samples, where $n = t \cdot l$.

weightings across market sectors where $\sum_{i=1}^{m} A_{ij} = 1$ for all j, see Figure 5.2. The next step is to find the covariance matrix C of the data. This is done by translating the data so that the barycentre of the points is at the origin:

$$B_{ij} = A_{ij} - \frac{1}{n}\sum_{j=1}^{n} A_{ij} \tag{5.1}$$

and multiplying the resulting matrix B by its transpose to find C:

$$C = \frac{1}{n} BB' \tag{5.2}$$

Next, an eigen decomposition is obtained such that $C = Q\Lambda Q'$ where Q contains the eigenvectors $q_{1\ldots m}^{\rightarrow}$ and Λ is a diagonal matrix of the corresponding eigenvalues. The two eigenvectors with the largest eigenvalues, q_1 and q_2 are then the principal components.

Finally, the projection can be performed to find x and y coordinates of the n data points in two dimensions:

$$x_i = p_i' \vec{q_1} \tag{5.3a}$$

$$y_i = p_i' \vec{q_2} \tag{5.3b}$$

where $1 \leq i \leq n$.

5.4 Displaying temporal changes by extruding into 3D

The novel visualisation of the 2D PCA projection described above, involves extruding the data into 3D such that time is represented by the 3^{rd} dimension. The result is a mass of "worms" crawling through time. The direct, independent and discrete mapping of time to the 3^{rd} dimension makes this a $2\frac{1}{2}$D visualisation as defined in Section 2.1.

Each of the data points comes from one of t samples, usually taken at regular intervals in time. Each of the n data points above is now assigned the z coordinate such that $z = c \cdot (i - t/2)$ where $1 \leq i \leq t$ and c is a constant scale factor. To draw the worms, each pair of adjacent points representing the same portfolio is connected with a line segment.

It is worth noting that in any type of multidimensional scaling the position of data points in the reduced (k-)dimensional space relative to the axes is meaningless. The important thing is that clusters and outliers are clearly visible. In the "worm" visualisation an analyst can see how various portfolios move in and out of clusters or towards or away from each other as time progresses. This also allows colour and thickness of the lines to be used to display additional attributes as they change over time.

Figure 5.3 shows the essential elements of the overview visualisation when applied to a small set of dummy data. The data-set contains three sets of holdings, including weightings for the market index, spread across three market sectors and captured at 6 points in time. That is, $m = 3$, $l = 3$ and $t = 6$. Coloured cones have been added, visible in the lower half of the figure. Each of these corresponds to a market sector or dimension in the portfolio space. The cones are placed, at the level of the lowest plane, at the position where a portfolio, fully weighted in the corresponding sector i, would be placed. That is, the x and y coordinates for the cone for sector i are exactly the i^{th} elements of $\vec{q_1}$ and $\vec{q_2}$ (Eq. 5.3) respectively.

For example, one of the two portfolios is fully weighted in Sector 3 in the last time sample (note, that time increases from bottom to top as shown). Thus, its highest point lies directly above the cone. The height of the cones corresponds to the percentage of the total holdings placed in that market sector.

The market index is like a "hypothetical" benchmark portfolio. Market index data is just like a portfolio in that it has weightings across stocks (and therefore market sectors) that change over time. In the figures the market index is distinguishable from other market sectors by its blue colour and is marked by an inverted cone above the top-most time plane.

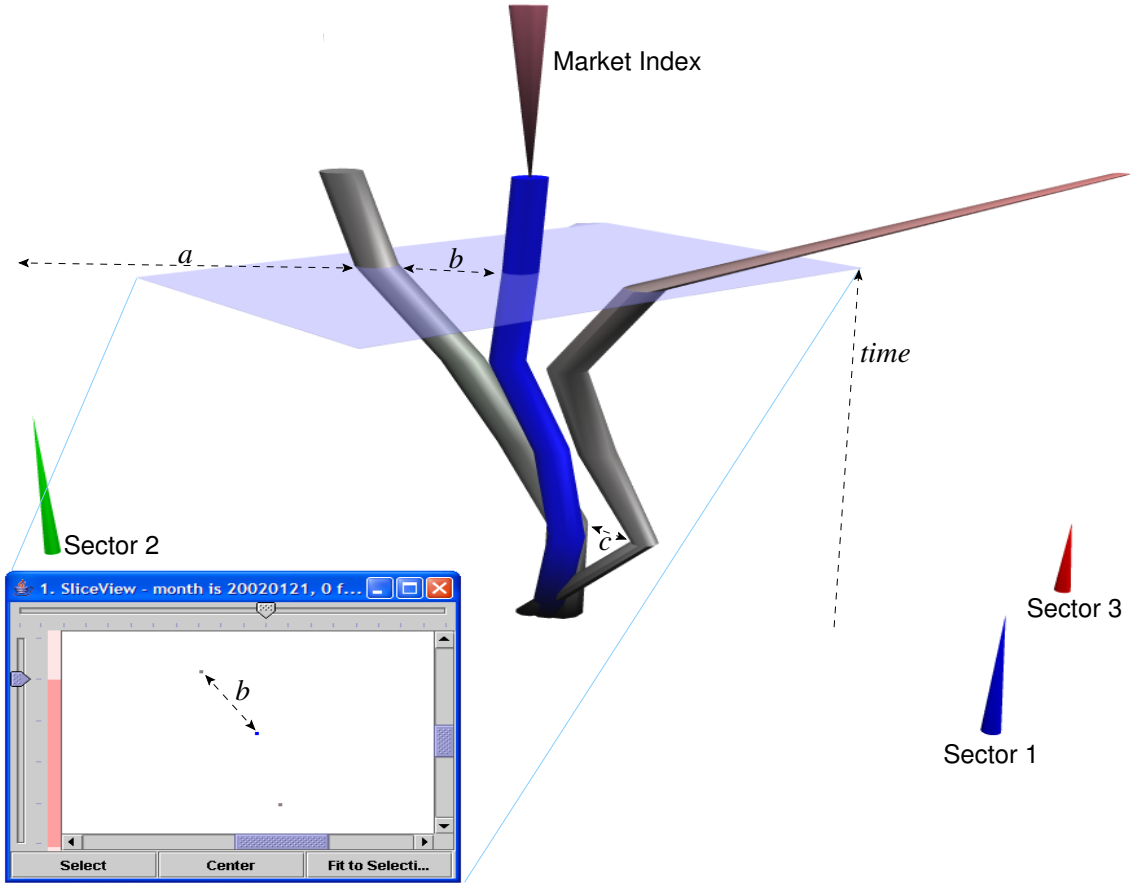


Figure 5.3: A visual explanation of the worm-view used in the PORTFOLIOSPACE EXPLORER using a simple three dimensional data-set.

The distances marked in Figure 5.3 correspond to:

(a) The difference in weighting between an actual portfolio and a hypothetical portfolio fully weighted in a particular market sector.

(b) The difference between a portfolio and the market index.

(c) The difference between two portfolios.

Two other important attributes are captured in this figure. Cross sectional area of the worms corresponds to total value of the portfolio, and colour is used to emphasise change in this value. Grey is the default colour and indicates no change in value from the previous time period, but hues can shift towards green to show increase in value or red to show decrease.

A cross-sectional view of the scene, corresponding to one time period, is captured by the small

window in the foreground. This corresponds to the cross-section at the level of the transparent blue "water-level", which may be repositioned or resized with the controls in the window.

5.4.1 Projection Relative to Index Data

In studying fund manager performance, it is important for an analyst to be able to determine the extent to which the holdings of fund managers — and therefore, the fund managers' performances — deviate from the underlying market portfolio or index. The market index defines the universe of stocks and industry sectors from which investors select securities and manage their portfolios, and these index weights are determined largely as a function of the market capitalisation (or size) of a company listed on an Exchange. The finance literature applied to portfolio management strongly emphasises the importance of the market index in portfolio selection, and studies (such as that of Roll [163]) have identified how critical *tracking error* is in portfolio risk management. Tracking error of a fund is usually defined as the difference between fund returns (the increase in value of the fund) and index returns. A fund is said to *out-perform* the index over a period of time if it increases in value more (or decreases in value less) than the index in that time period. It is said to *under-perform* if it increases in value less (or decreases in value more) than the index. Thus tracking error is evident in the worm visualisation when a fund worm moves independently to the market index worm.

A slight modification to the PORTFOLIOSPACE EXPLORER allows the projection to be recast such that portfolio worms are positioned relative to the index, and tracking error is more easily seen. For the UK market data the FTSE 350 index data across all market sectors has been obtained for the same period. This data is then placed in an $m \times t$ matrix F. By subtracting F from the $m \times t$ sub-matrix of A for each of the l portfolios before calculating B as in Equation 5.1, the projection shown in Figure 5.4 is obtained. In this visualisation, any movement in the fund worms indicates potential tracking error since it indicates a change in weighting in the fund that does not occur in the index. Actual tracking error (difference in returns between net fund value and index value) can be emphasised by colouring the worms to show out-performance or under-performance relative to the index for each time period, rather than total profit or loss. Note that the hypothetical portfolios, fully weighted in the various market sectors, now also move relative to the index and thus, are no longer shown as cones but as bendy worms. Conversely, the market index is now a straight column centred between the sectors.

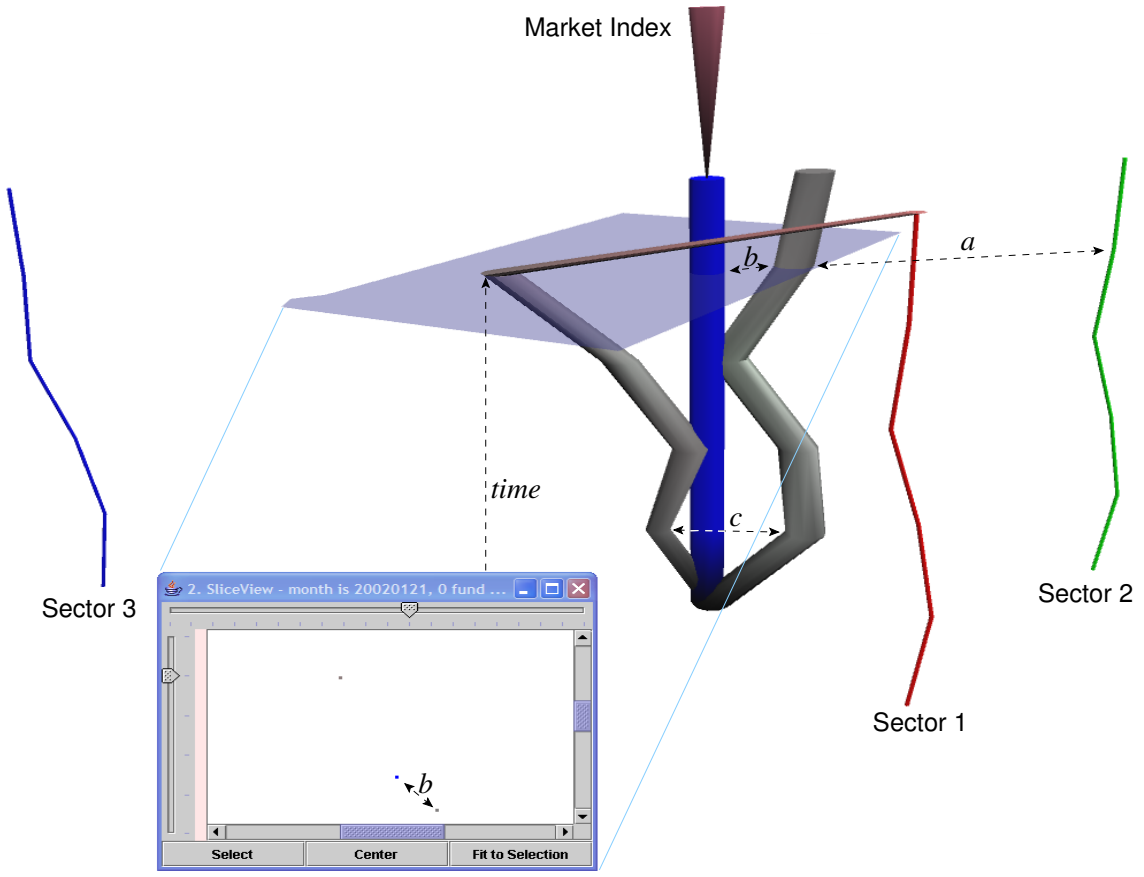


Figure 5.4: The data-set from Figure 5.3, shown with all portfolios placed relative to the market index. The distances marked correspond to those in the previous figure.

5.4.2 User Navigation

An essential part of any 3D (or $2\frac{1}{2}$D) visualisation is providing the user with the ability to freely rotate, zoom-in or otherwise "fly" around the 3D model. When viewing a static projection of a 3D visualisation, the user has no sense of depth and the extra dimension is wasted (see Ware [195]). In the PORTFOLIOSPACE EXPLORER system this capability is provided in a fairly standard way with mouse interaction.

However, PCA provides the ability to navigate around the data-set in fundamentally different ways. By default, the two eigenvectors associated with the two largest eigenvalues are used to define the projection plane. This, by definition, captures the greatest variance in the data. For example, Figure 5.5 gives a histogram, or *scree diagram* [30], of the eigenvalues for the components or eigenvectors of the trivial synthetic data matrix used in Figure 5.4. However, one can just as easily use any pair of eigenvectors. Allowing the user to choose components directly from the scree

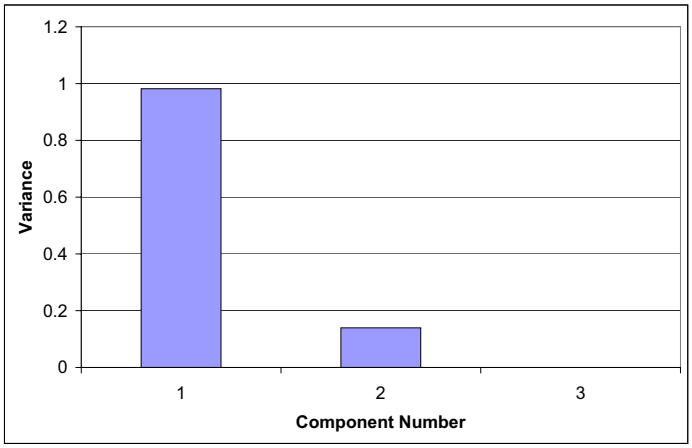


Figure 5.5: A scree diagram showing the variance captured by the components of PCA for Figure 5.4. Users can select the eigenvectors defining the projection plane directly from this display.

diagram, thus changing the eigenvectors used to determine the projection plane, provides a high-dimensional rotation which may capture different aspects of variation in the data.

A zoomed view is also easily obtained by producing another PCA projection of a subset of the visible data. In Figure 5.12 the user is in the process of performing such a zooming operation. In the cross-section on the right hand side the analyst has selected a subset of portfolios by sweeping out a rectangular area with the mouse. The analyst can then select a subset of the available time samples by moving the water-level up or down, thus creating the transparent box seen in the $2\frac{1}{2}$D view on the left-hand side of the figure. The selected subset of the data will then be re-projected as before and the zoomed view shown in a new window. The zoomed window is shown in Figure 5.13. Allowing the user to box a region of the $2\frac{1}{2}$D structure for zooming is a logical extension of the PCA based rotation and zooming strategy given in [90] for interacting with a 2D display.

The synthetic data-set used in these examples is trivially small: three dimensions, two portfolios and an index over six time samples. Section 5.7 demonstrates the utility of the PORTFOLIOSPACE EXPLORER worm view with a much larger example compiled from actual holding data.

5.5 Detailed Graph Visualisation Based View

The PORTFOLIOSPACE EXPLORER "worm" visualisation, described in Section 5.3, provides an overview of the data-set. In order to capture broad movements across as much data as possible the PCA-based dimensional scaling is a necessary, though severe, abstraction of the underlying detail.

To visualise the detailed behaviour of individual fund managers as they re-balance their portfolios, a different approach is required.

Assume that an analyst selects an individual worm from the PORTFOLIOSPACE EXPLORER overview for closer inspection. Effectively, the analyst has isolated a set of data for a single portfolio over a number of time periods. In the UK data-set, this includes share price data and the count of shares of a particular type held in the equity portfolio. That is, we have two matrices in which the columns are associated with market sectors (or any other aggregation of stocks, or individual stocks) and the rows are associated with each of the sample times (the examples shown here have 12 monthly samples). The first matrix P holds share price data. When an aggregation of shares is used this is the average unit price across the aggregate. The second matrix Q contains the counts of shares held in the portfolio.

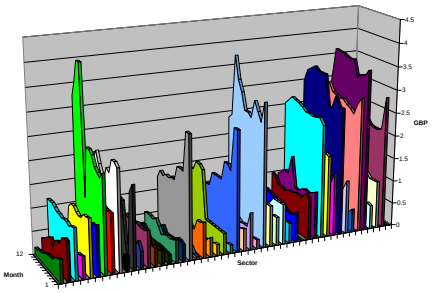
These matrices could be visualised by simple 3D area charts. For example, Figure 5.6(a) shows the share price data P, in this case the average share price for each of $n = 50$ market sectors for a year's worth of data. Figure 5.6(b) shows the numbers of shares held in the portfolio of a particular fund manager across the same time period. Figure 5.6(c) charts the total value of the portfolio across the time period, where the value x at each month j is:

$$x_j = \sum_{i}^{n} P_{ij} Q_{ij}$$

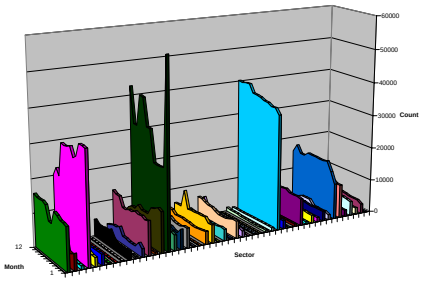
The relatively flat curve in Figure 5.6(c) shows that by re-weighting the portfolio the fund manager has managed to even out the volatility in the share prices. Visualisations like that of Figures 5.6(a) and 5.6(b), show a great deal of activity taking place. However, one must ask whether it is possible to produce a visualisation which focuses an analyst's attention more specifically on the fund manager's movement between market sectors.

In [47] we proposed a graph-visualisation-based approach for the movements of fund managers between different stocks or market sectors (in the sequel only sectors are referred to). The graph for fund manager movement is defined as $G = (V, E)$ consisting of a set V of nodes representing sectors and a set E of edges where each edge $e(u, v) \in E$ represents "movement" of one or more fund managers from sector $u \in V$ to sector $v \in V$.

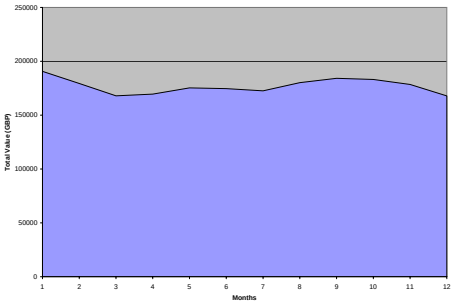
For the matrix Q of share counts in each sector, a graph can be constructed in which a node represents each non-zero column. Beginning with the second row, the values in each column are compared against that column's value in the previous row. For each column (sector) showing a



(a) A 3D area chart showing the fluctuations in share price in a particular portfolio over 12 months. Each of the columns is a different stock, the depth axis is time (most recent at the front) and the vertical (labelled axis) shows share price in GBP (British Pounds Stirling).



(b) A 3D area chart showing the changing count of shares in the portfolio over 12 months. The axes are as in Figure 5.6(a) except the vertical axis which shows count of shares



(c) This chart shows the net effect on the total value of a portfolio of the share price fluctuations, shown in Figure 5.6(a), and the fund manager's re-weighting, shown in Figure 5.6(b).

Figure 5.6: Traditional chart views of fund manager holdings

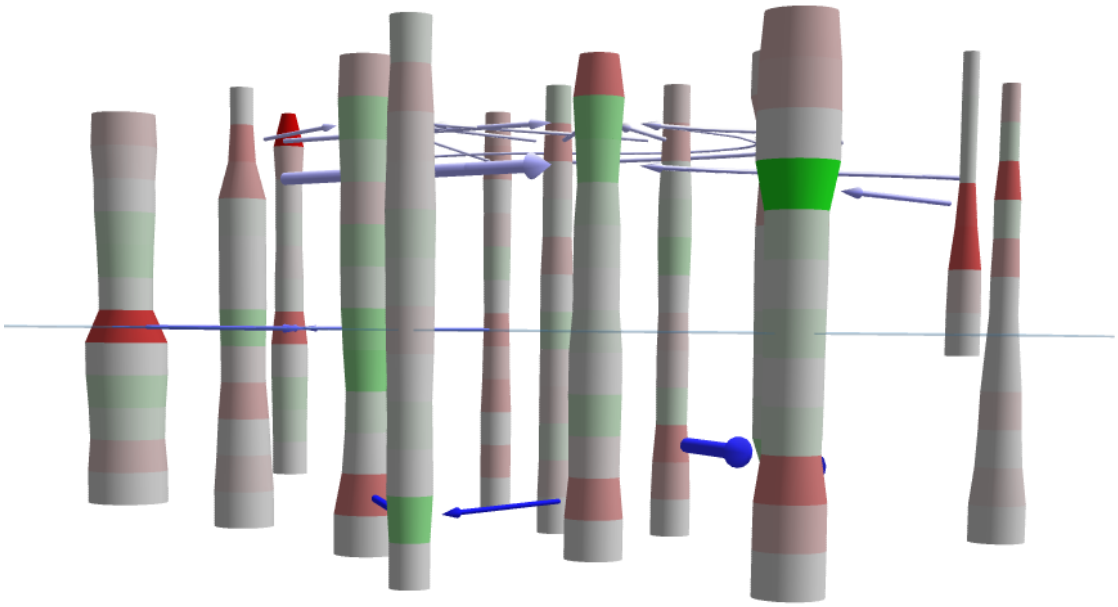


Figure 5.7: Side view of the stratified graph, clearly showing the strata.

decreased holding edges are created connecting the column to all the columns with an increased holding. To allow the user to focus on more significant movements, they can adjust a threshold in increase or decrease of stock price below which no edge is created. Comparison of each pair of rows continues, creating an evolving graph (see Definition 4.2.1), with edges assigned to a subgraph for each time period, until all rows have been examined.

To visualise this graph so that it is easy to see at what time different movements occurred, it is possible to use a $2\frac{1}{2}$D paradigm similar to that followed for the worm view. That is, the graph drawing is extruded into the third dimension, see figures 5.7 and 5.8. The nodes become pillars or columns parallel to the new third axis and the edges are placed perpendicular to the axis at a level dependent on the time (matrix row) at which the movement they represent occurred. At the bottom right of Figure 5.8 there is a cross-section viewer, similar to that used in the worm view. This displays the cross-section highlighted by the transparent blue plane in the $2\frac{1}{2}$D window.

The total value of a market sector at each point in time (i.e. $P_{ij}Q_{ij}$) can be shown by setting the radius of the column node representing that market sector. Clutter in the graph may be reduced by only including nodes for which the maximum value is greater than a threshold. In other words, it is only necessary to include market sectors which make up a significant proportion of the portfolio.

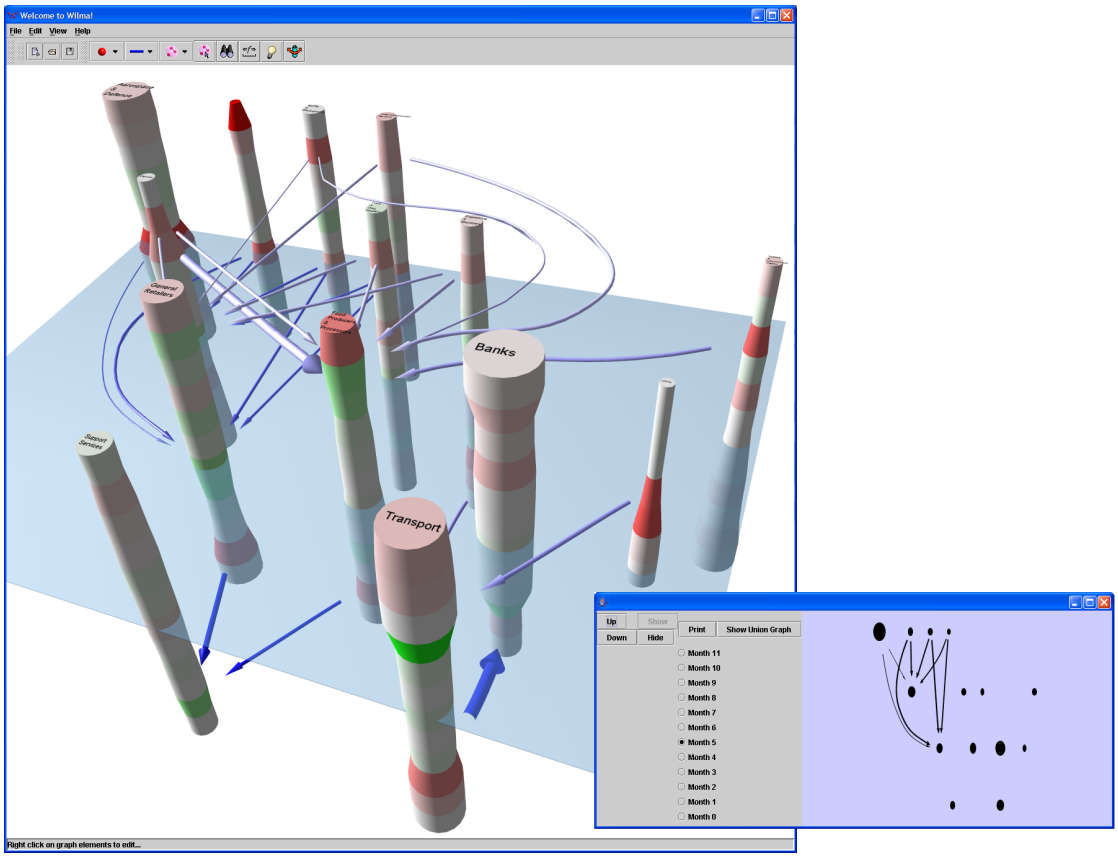


Figure 5.8: Screen shot of the graph based detail view of one portfolio (by industry sector).

By default this is automatically calculated in the system to show the ten largest sectors, but the user can increase or decrease this number. The changing average share price information from P can be shown by colouring each segment of the column. As in the worm view, the hue of the columns is greener if there is an increase in average share price from one period to the next, and redder if there is a decrease. The default colour, light grey, means there is no change.

In the extruded $2\frac{1}{2}$D view the edges can be shown as tubes or pipes between the columns. The total value of a given movement is shown by adjusting the thickness of the edges. That is for an edge showing a change in holding from time samples t and $t+1$ between nodes representing columns i and j in P and Q the radius of the edge is proportional to:

$$(Q_{i(t+1)} - Q_{it})P_{i(t+1)} + (Q_{jt} - Q_{j(t+1)})P_{j(t+1)}$$

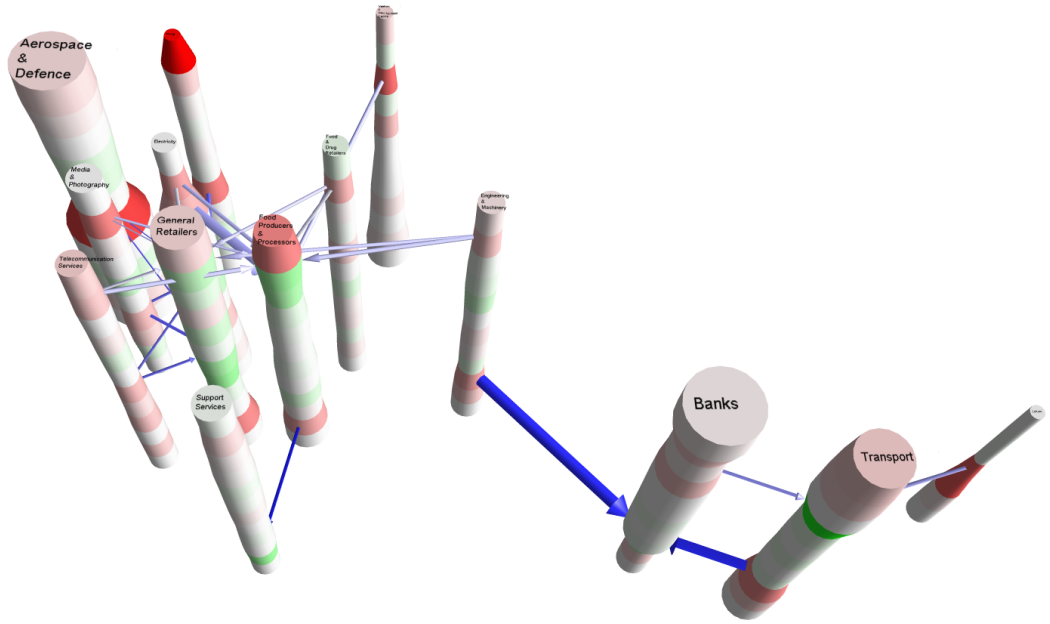


Figure 5.9: PORTFOLIO COLUMNS arranged using force directed layout.

5.6 Stratified Graph Layout

In Section 4.3 extensions of force-directed and hierarchical layout methods for stratified graphs are defined. Figure 5.9 shows an example of a fund-manager-movement graph arranged with a force-directed approach. A directed-field force, as described in Section 3.2.1, is used in addition to the basic inter-node-repulsion and edge-attraction forces to encourage edges to point downwards. As discussed in Section 3.2.1, this helps to show flow from net sources to net sinks.

This graph theoretic concept of flow is useful when considering layout of the portfolio movement graphs in that it roughly corresponds to the flow of money between different stocks. That is, source nodes represent stocks that are mostly being sold and sink nodes represent stocks that are more commonly being bought.

Hierarchical layout methods, as described in Section 3.2.1, are also useful in showing flow, but may not show clustering of highly coupled areas as clearly as force-directed approaches. The relative merits of the various stratified layout methods, for visualising such fund manager movement graphs, are discussed further in the next section.

5.7 Results

In order to evaluate and improve the implementation of the ideas described in this chapter, the system was shown to a number of domain experts, consisting of both professionals from industry and finance researchers expert in fund manager performance analysis. These domain experts were interviewed individually and were encouraged to explore the data using the tool, gathering their feedback in the process. The example walk-through given in this section is typical of the type of explorations of the data-set that were performed with the assistance of these experts.

As described in Section 1.5, the interviews were structured according to an informal cognitive-walkthrough style methodology in order to capture as much qualitative feedback as possible. The system was presented to the interview subjects on a large, rear-projected screen supporting stereo 3D. The interviews were video recorded, and a transcription of a typical interview is given in Appendix A.

The data-set used is limited to publicly available data that is gathered from forms filled in and submitted manually (and sometimes reluctantly) by stock issuers, according to United Kingdom statutory requirements. Early talks with — understandably application oriented — domain experts made it apparent that their attention was easily drawn to data-processing limitations of the tool and the limitations of the UK data-set. For example, the data was incomplete in a number of ways:

- Reference numbers for issuers and holders were often in error, making automatic correlation difficult.
- At best, data was limited to approximately monthly granularity, though sometimes data for certain companies is not updated for months at a time.
- The UK reporting laws require are that only the largest holders be declared by stock issuers. Generally, this means that only the portfolios for the largest fund managers could be reconstructed adequately to show real trends and movement.

Nevertheless, the data-set is adequate for demonstrating the possibilities of the tool to professionals who have access to much better quality data. The quality of the data shown in the visualisation is improved by filtering obviously erroneous data. For example, holders for whom data is not available in every month are removed, as are holdings data for stocks not in the FTSE-350 market index. The remaining data includes 542 portfolios with weightings spread across 45 market sectors.

Below, a step-by-step description of a typical use-case scenario is given, demonstrating all aspects of the system in use on 12 months worth of UK stock market holdings data. This use-case is typical of the scenarios shown to the panel of interview subjects. A number of examples of useful observations the experts were able to make are noted.

5.7.1 Use-Case

Figure 5.10 is a screen-shot of the initial PORTFOLIOSPACE EXPLORER overview of the 542 portfolios that remain after the above filtering process. Twelve months worth of data from September 2001 to August 2002 (at monthly granularity) is shown. This is the starting point for the use-case, as shown to the interview subjects. The PCA projection has been performed relative to the index, as described in Section 5.4.1. Therefore, the index is a partially obscured, straight blue column in the centre, marked by a large inverted cone. Some additional user interface features are shown in this figure that were not described previously. These include a list of sectors down the left hand side of the screen, where the coloured bands indicate the mapping to the sector markers in the 3D window and the widths of these bands indicate the total capitalisation of the sectors within the holdings data shown. Similar displays are also available from the tabs in the top left corner, that give a break down of the total value by portfolio (fund manager) or month. Another feature is a custom zoom control: the slider across the bottom of the screen allows the user to magnify the two projected dimensions independently from the third, time dimension.

Already, we can begin to learn about the data. Some initial observations noted by the interview subjects include:

Clustering — The majority of the portfolios, especially the high valued portfolios represented by relatively fat worms, are clustered in a tight bunch around the index. This makes sense since most fund managers, particularly those managing large, high value funds, should be attempting to either track the index's performance (so called "index funds") or else maintain weights within a specific range from this benchmark ("active funds").

Variance — Most of the variance seems to be captured by the first component which is projected horizontally in both the cross-section view and the main 3D window. This observation is born out by the scree diagram shown in Figure 5.11; clearly the first component captures twice as much variance as

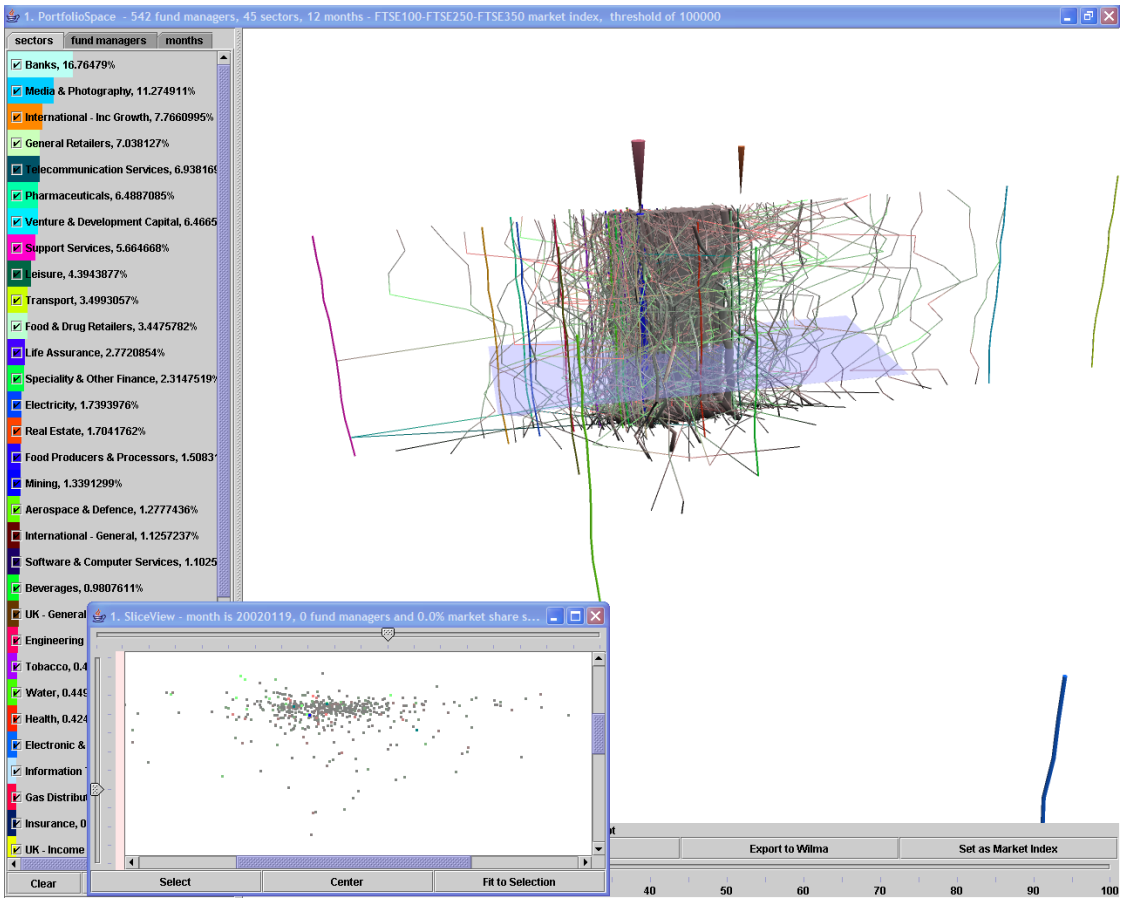


Figure 5.10: A PORTFOLIOSPACE EXPLORER visualisation of the weightings of 542 portfolios across 45 market sectors.

the second.

Sector Variance — The placement of the sector markers also gives us feedback about the spread of the variance. Most of the sector markers lie along the principal component indicating some correlation amongst those sectors. However, there are several distinctly outlying sectors. In the bottom right corner — that is, closest to the camera — the top of the marker corresponding to the "Electronic and Electrical Equipment" sector can be seen. Closer inspection, involving flying around the scene a little, reveals four more sectors lying distinctly away from the main axis of variance: these are "aerospace and defence", "beverages", "steel and other metals" and "health". Feedback from the panel has confirmed that these are volatile sectors for the UK market, where *blue-chip* stocks are mostly in the financial services area.

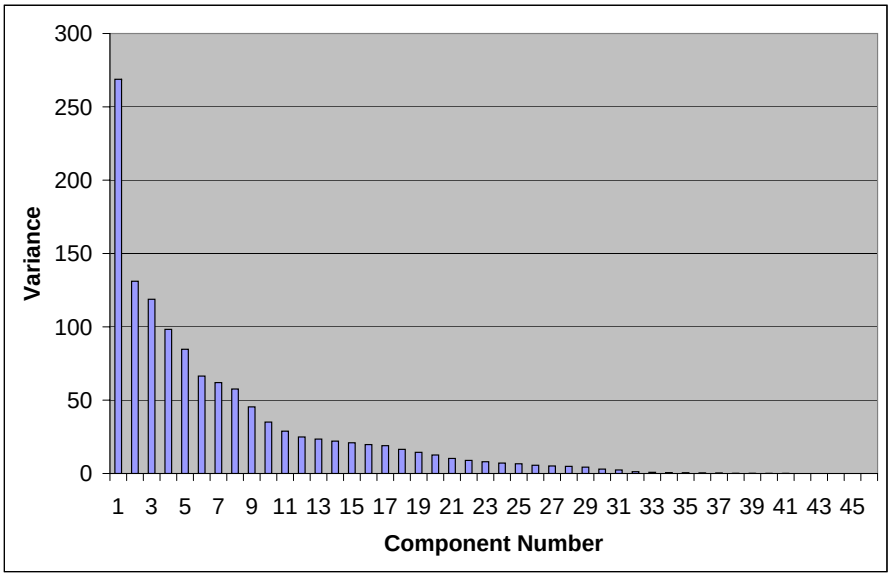


Figure 5.11: A scree diagram showing the variance across components for the PORTFOLIOSPACE EXPLORER visualisation in Figure 5.10.

Outliers — On the fringes of the view, the outliers tend to show extremely chaotic behaviour. The small diameter of these worms indicates that they are relatively low-valued portfolios. Closer inspection of these outlying portfolios reveals that they are generally weighted in only one or two sectors. This may be indicative of a specialised type of portfolio or possibly incompleteness in the UK data.

Volatility — The unique aspect of this $2\frac{1}{2}$D visualisation is that it captures variance of the high-dimensional data across time. One example of the types of pattern that this allows us to see is again apparent in the thinner, lower valued funds. Specifically, compared to the fatter, higher valued funds they are much more chaotic or volatile. Again, explanations could be that the smaller funds are more specialised in particular industries, or that the data for these funds is incomplete.

The attention of the interview subjects is typically drawn to the central cluster of larger funds

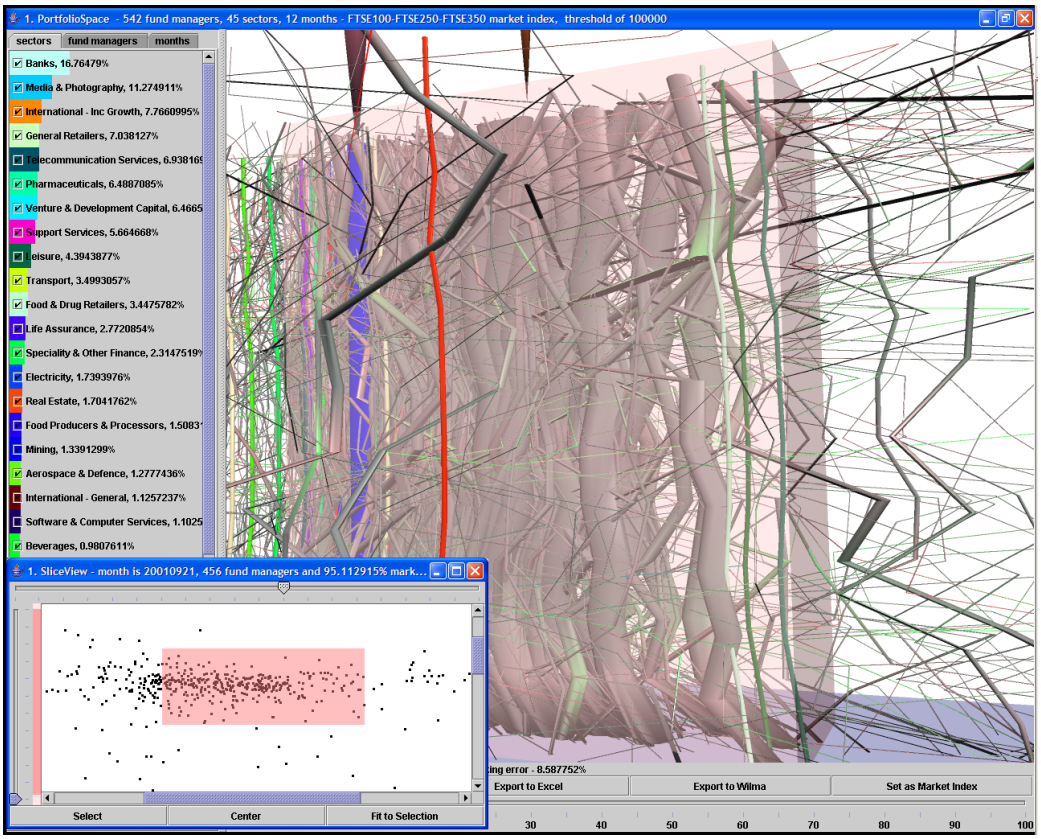


Figure 5.12: A screenshot showing the process of zooming into the cluster identified in Figure 5.10.

closer to the market index. Therefore, this is a good place to probe more deeply. In Figure 5.12, the operator has swept out a region in the slice view and then extruded that region out over the whole twelve month period producing the transparent pink box identifying the zooming range. The zoomed view showing only the 184 portfolios that do not stray outside of this region is shown in Figure 5.13.

Again, there are some interesting observations to note in the PORTFOLIOSPACE EXPLORER visualisation of the 184 portfolios that were isolated in the zooming operation.

Better distribution of variance across sectors — In this scene there seems to be a better spread in all directions of the sector indicators in Figure 5.13 compared to Figure 5.10. This makes sense since one would expect the group of larger funds selected to be more evenly weighted across all major sectors.

Movement relative to index — In the magnified view shown in Figure 5.14, it is possible to see that many portfolios seem to move together at particular points in time. This may indicate significant movement in the market — movement

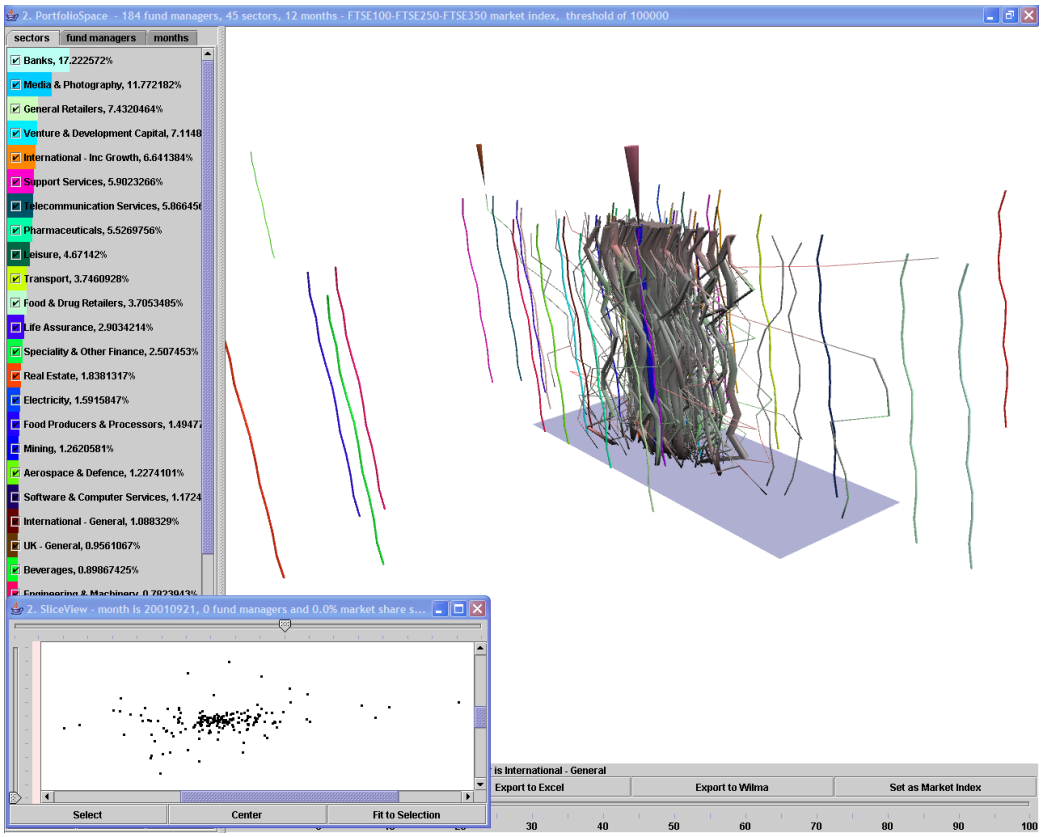


Figure 5.13: The result of zooming the region selected in Figure 5.12.

that is difficult for the fund managers to track — or significant economic events triggering bullish or bearish behaviour.

In Figure 5.14 one portfolio literally stands out from the crowd. The "INVESCO Asset Mgmt" portfolio is a fairly fat portfolio that stands a little in the foreground compared to the main cluster of portfolios that lie closer to the axis of greatest variance. This portfolio has been selected by the operator for closer inspection (the operator simply selects the portfolio with the mouse pointer and an inverted cone appears, marking the top of the portfolio).

In Section 5.5 two methods for visualising the detailed movement within a single portfolio are considered: with more conventional charting techniques; or with the $2\frac{1}{2}$D graph-based PORTFOLIO COLUMNS view. The PORTFOLIOSPACE EXPLORER system allows the analyst to generate both styles of visualisation. Figure 5.15 shows area charts generated by the PORTFOLIOSPACE EXPLORER system for both the INVESCO portfolio and the FTSE350 index. These chart views include a degree of interaction. For example, the user can directly select a particular sector curve and the total value at which this sector intersects the (moveable) transparent blue plane is displayed

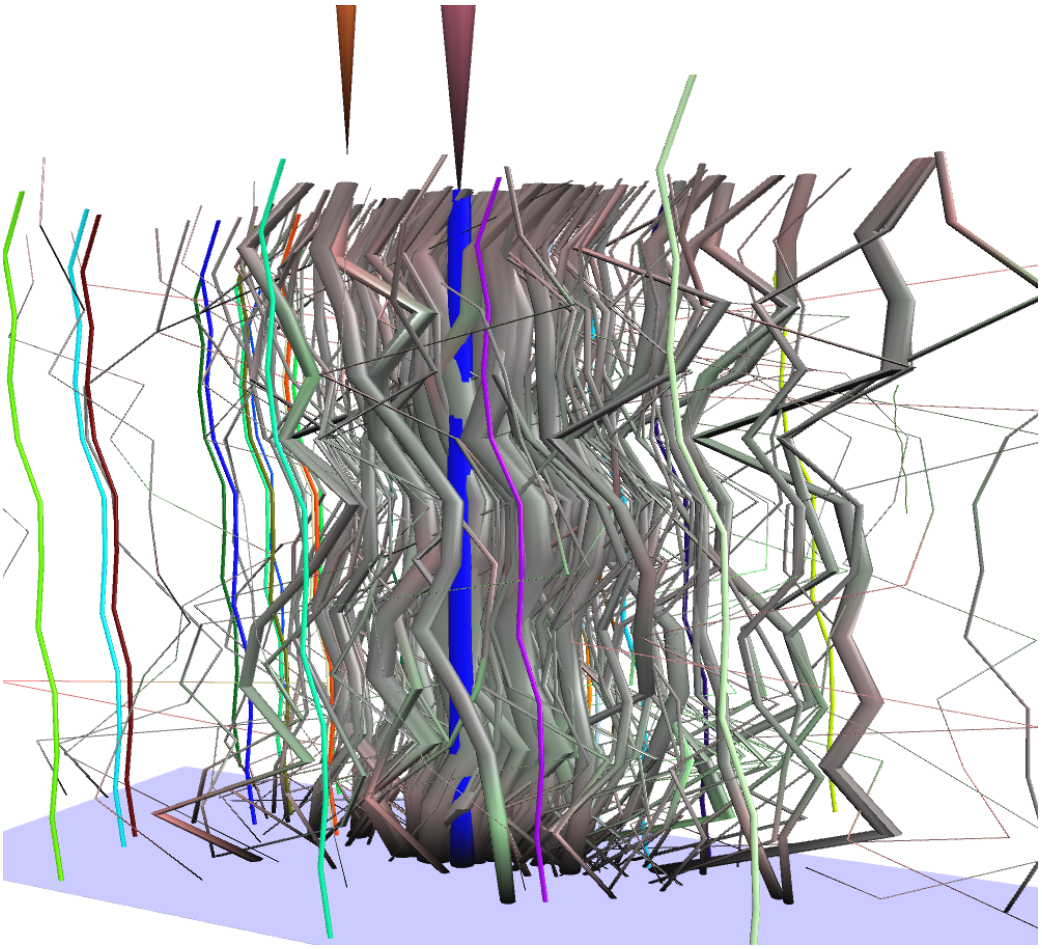
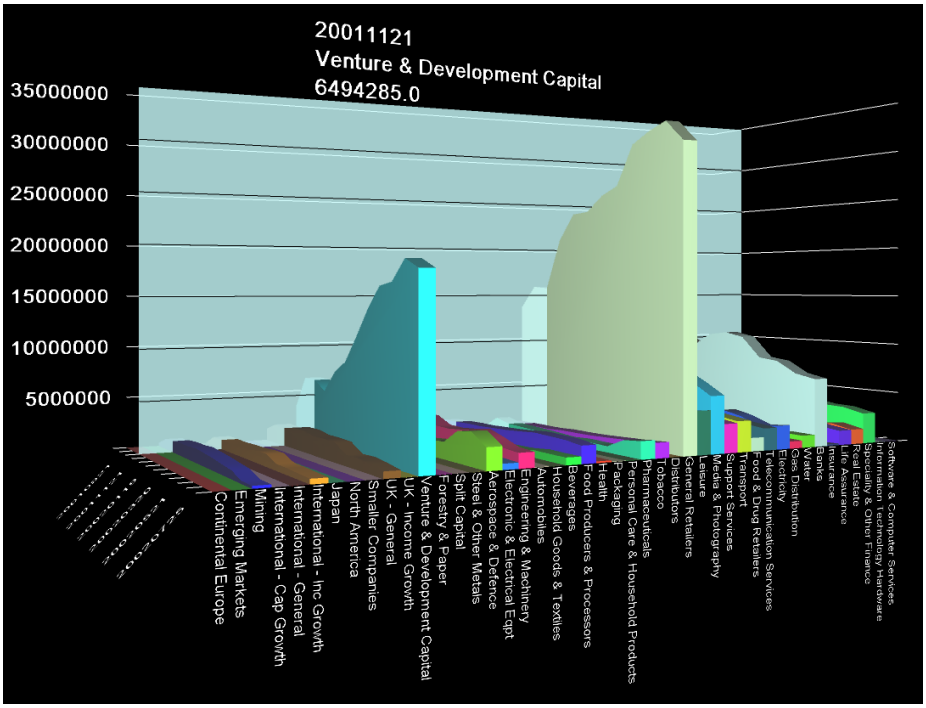


Figure 5.14: A magnified view showing detail around the index from Figure 5.13.

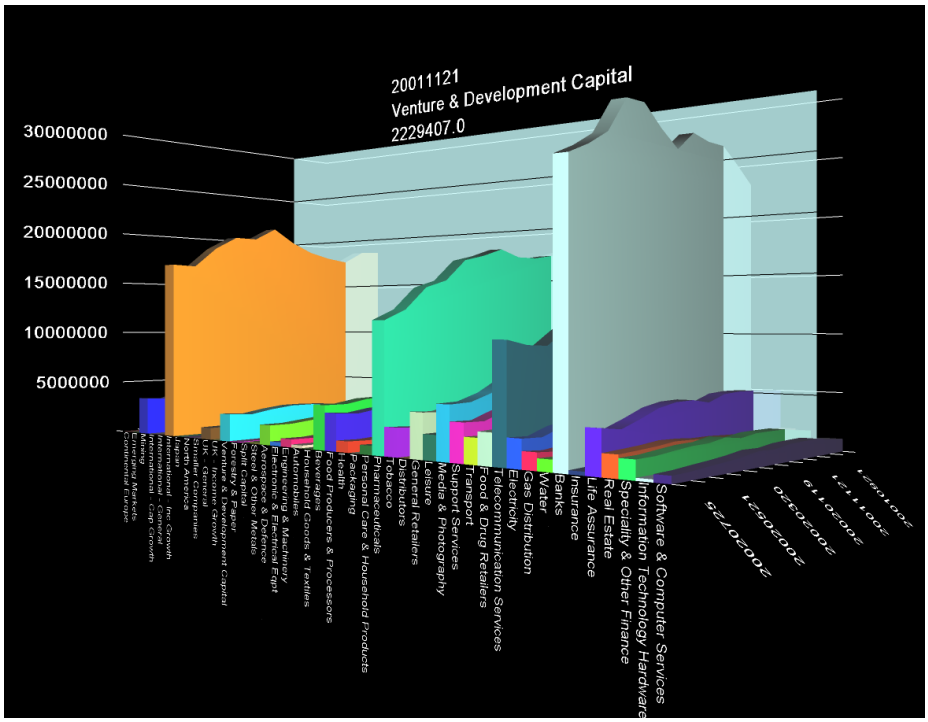
above the figure. In the Figure 5.15 the user is examining the value of the "Venture and Development Capital" sector in November 2001 in both the index and the INVESCO portfolio. From these charts the user can deduce that the reason that the INVESCO portfolio was separated from the main cluster around the index in the PORTFOLIOSPACE EXPLORER overview, is that the INVESCO fund dramatically increases its holding in the "Venture and Development Capital" and "General Retailers" sectors over the twelve month period in question.

The alternative PORTFOLIO COLUMNS view proposed in Section 5.5 for the INVESCO portfolio is shown in Figure 5.16. The total-value by sector information that was the subject of the area chart from Figure 5.15(a) is still visible in the widths of the columns. For example, the two larger sectors, noted above, still stand out as being fatter columns. However, the additional movement information indicated by the arrows gives us some richer feedback. For example:

Periods of greater movement — Turning the columns side on, as in Figure 5.16(b)

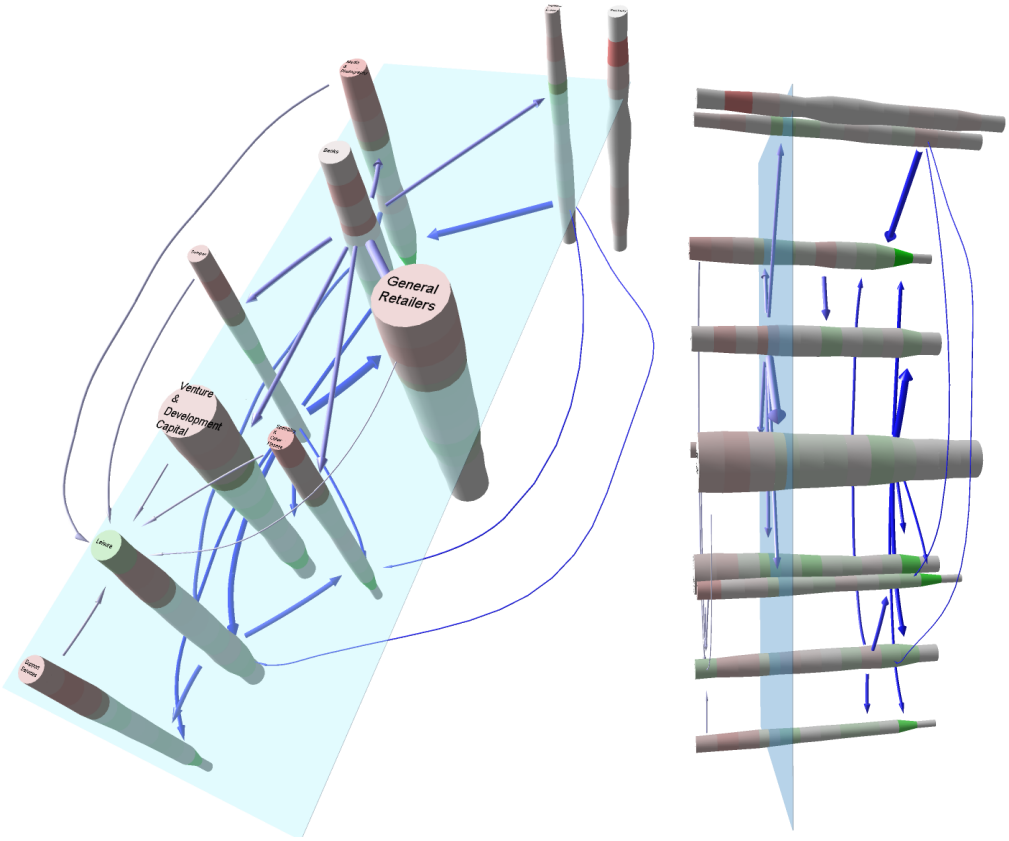


(a) INVESCO Asset Mgmt



(b) FTSE350 Index

Figure 5.15: Area charts generated by the PORTFOLIOSPACE EXPLORER system to show the detailed movements within individual portfolios.



(a) viewed from above (b) from the side

Figure 5.16: A PORTFOLIO COLUMNS visualisation of detailed movement within the INVESCO portfolio

shows that there are three periods of significant movement, one in the early months, and two in the last months.

Net sources and sinks — In general, the Sugiyama layout style places net sources of edges closer to the top of the plane and net sinks nearer the bottom. This is also visible generally in Figure 5.16, and, as a specific example, Figure 5.17 is a close-up showing more clearly that the banking sector is a significant source of movement in the ninth month.

Relationship between stock price volatility and movement — In Figure 5.17 it is evident that most of the movement seems to be concentrated around the most volatile stocks. That is, most of the edges are associated with brightly coloured column segments. This sort of behaviour seems consistent with the way that one would expect fund managers to behave in managing a portfolio,

but the fact it is evident from the visualisation is a reassuring confirmation that the visualisation reflects reality.

5.7.2 Further Feedback

Below, some of the feedback from the interview panel which led to improvements in the tool or which will require further investigation in future, is noted.

Clarifying dimensional scaling — The domain experts, perhaps because of their familiarity with traditonal charts and plots, had particular trouble understanding the concept of multidimensional scaling. They almost always asked questions such as, "What do the x and y axes mean?". It required the domain experts to make a certain "leap of faith" to accept that points that are close together in the projected view correspond to similarly weighted portfolios. Placing the coloured sector markers in the projected view helped crystalise this meaning for the users. Also, projecting relative to the market index helped them make the "leap of faith"; perhaps this was because the straight column representing the market index gave them an understandable reference point in the unfamiliar space.

Performance relative to market index — This factor, as described in Section 5.4.1, was key for the domain experts, and became a central design concern.

Performance based on returns — One comment that was almost universal amongst interview subjects was that an analyst needs to be able to visualise a portfolio based on returns as well as in absolute terms. That is, currently in the PORTFOLIOSPACE EXPLORER the worms are coloured based on increase or decrease in portfolio value. Instead, they could be coloured black if they have matched the performance of the index, green if they have increased in value more than the index, or red otherwise. Although this is a fairly simple change, the lesson is that the interface needs to provide a lot of flexibility in how data is mapped to visual attributes.

Flow versus clustering — In the detailed PORTFOLIO COLUMNS view most interview subjects preferred the hierarchical (Sugiyama) style layout to the force-

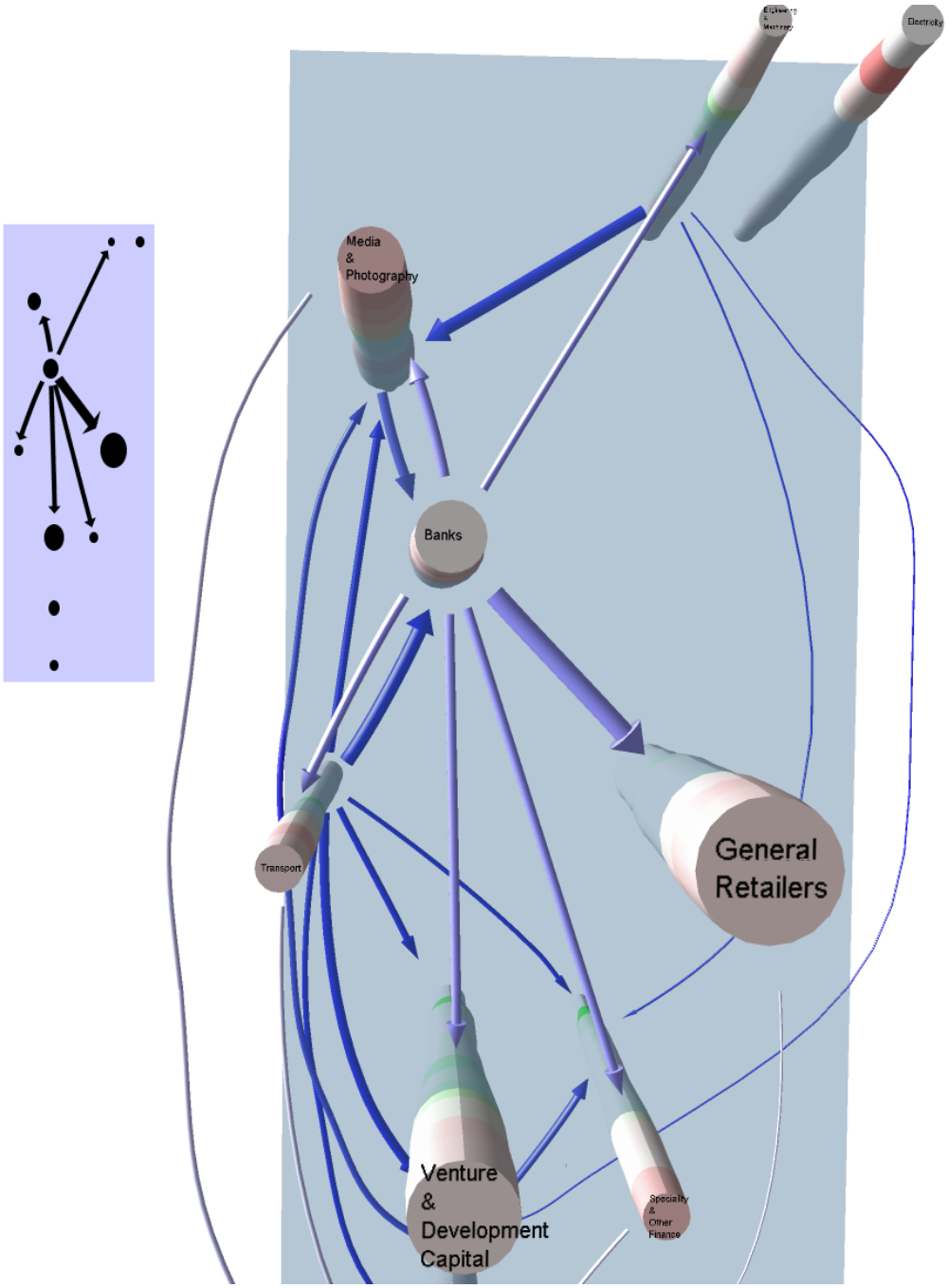


Figure 5.17: A close up showing the movement out of the banking sector in the INVESCO portfolio. The cross-section from the highlighted level is also shown.

directed style, although they could see merit to allowing the user to choose the layout style based on whether they are looking for flow or clustering in the graph.

5.8 Conclusion and Further Work

Two $2\frac{1}{2}$D visualisations for fund manager movement data are demonstrated. These may be coupled to provide an holistic, overview and detail system[1]. The first visualisation, the PCA based PORTFOLIOSPACE-EXPLORER view, compresses a great deal of information about the entire data-set into a single scene and takes advantage of the speed and flexibility of PCA to allow a user to focus quickly on smaller regions of detail. The second visualisation paradigm, the graph based PORTFOLIO COLUMNS view, brings together the most important information from all three charts illustrated in Figures 5.1, 5.6(a) and 5.6(b) into a single visualisation and draws an analysts attention to the features in which they are most interested. Particularly, it allows an analyst to directly see the correlation (if any) between stock price and a fund manager's behaviour in re-weighting the portfolio.

A broad overview and definition of these two paradigms is provided. A concrete prototype system has been developed and evaluated with domain experts. This chapter describes some enhancements made to this system based on this feedback.

Finally, since the paradigms proposed should be applicable to any high-dimensional, multivariate data-set, it is hoped that in future their utility can be tested in other application domains.

[1] More information on the implementation of this system is given in Appendix B

CHAPTER 6

Metabolic Pathways

"Vision is the art of seeing things invisible." — Jonathan Swift, 1667–1745

This chapter presents the second case study using the two-and-a-half dimensional, stratified graph visualisation paradigm. Some applications are proposed involving the visualisation of metabolic pathways using the $2\frac{1}{2}$D graph visualisation techniques introduced in Chapter 4. All the applications discussed are based on a graph model for metabolic pathways described in Section 6.1.

The first application[1] (Section 6.2) involves visualising experimental data in the context of the underlying metabolic processes. A 2D layout is found for the metabolic pathway being studied in the experiment and this representation is then extruded into the third dimension to show changes in experimental data over time. This is a fairly straightforward adaptation of the stratified graph visualisation techniques previously discussed in Chapter 5 for visualising changes in fund manager holdings over time.

The second application[2] (Section 6.3) is a method for visualising a set of related metabolic pathways across organisms using $2\frac{1}{2}$D graph visualisation. Interdependent, 2D layouts of each pathway are stacked on top of each other so that biologists get a full picture of subtle and significant differences among the pathways. The (dis)similarities between pathways are expressed by the Hamming distances of the underlying graphs which are used to compute a stacking order for the pathways.

Finally, Section 6.4 discusses interaction methods allowing users to explore the set of metabolic pathways being studied[3]. Specifically, a prototype is demonstrated for a *coordinated visual triangulation* system in which the $2\frac{1}{2}$D view of the set of pathways, and the cross-section view of an individual pathway, are complemented by a view of the phylogenetic tree showing the relationships between the organisms to which the pathways belong.

[1] Work completed in conjunction with Schreiber and Rolletschek and published in [50].
[2] Work completed in conjunction with Brandes and Schreiber and published in [19, 18].
[3] Work completed in conjunction with Brandes and Schreiber and published in [17].

Section 1.4.2 gives a broad introduction to metabolic pathways and current methods for visualising them. This chapter begins by giving a more precise model for a metabolic pathway and a more in-depth survey of the visualisations that are most relevant to the work contributed here.

6.1 Representation of Metabolic Networks

In Section 6.1.1 a graph-based model for metabolic pathways is given. The problem of automatically visualising these structures can then be formulated as a graph drawing problem, which is discussed in more detail in Section 6.1.2.

6.1.1 A Graph Model for Metabolic Pathways

From a formal point of view, a metabolic pathway is a hyper-graph (see Section 3.1). The nodes represent the metabolites and the hyper-edges represent the reactions. Each hyper-edge connects all metabolites of a reaction, directed from reactants to products and is labelled with the enzymes that catalyse the reaction. As described in Section 3.1, such hyper-graphs can be represented by bipartite graphs, and this is often applied to modelling metabolic pathways. For example, Hofestadt et al. [95] and Reddy et al. [155] offer bipartite models for simulations of metabolic pathways. In bipartite models of metabolic pathways the reactions themselves are nodes, and edges are binary relations connecting metabolites of reactions with reaction nodes. An illustration of these two models is given in Figure 6.2. For the applications discussed in this chapter the following bipartite definition is used as the underlying model:

Definition 6.1.1 *A metabolic pathway is a directed bipartite graph $G = (V_1, V_2, E)$ with nodes $u, w \in V_1$ representing metabolites, nodes $v \in V_2$ representing reactions and pairs of directed edges $(u, v), (v, w) \in E$ representing reactions converting metabolite u to metabolite w. A bidirectional reaction $v \in V_2$ between two metabolites $u, w \in V_1$ is represented as a pair of edges both outgoing from v, i.e. $(v, u), (v, w) \in E$.*

However, when visualising the graphs it is not always necessary to show reaction nodes. In some of the images a reaction between two metabolites is simply drawn as an edge.

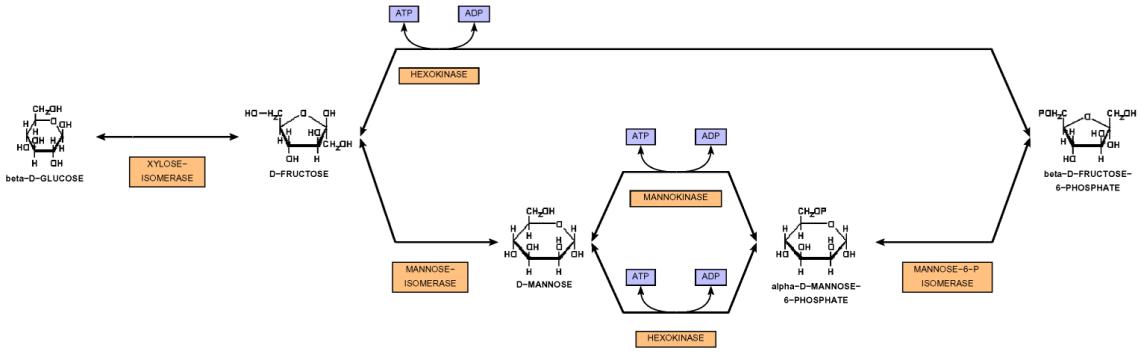


Figure 6.1: A metabolic pathway, drawn using the BIOPATH system. Courtesy Falk Schreiber.

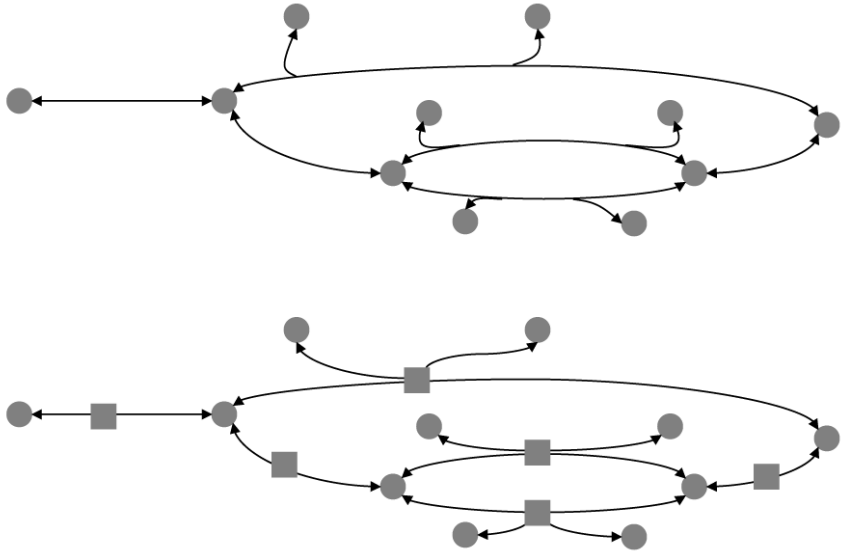


Figure 6.2: This picture shows: top - a directed hyper-graph representation of the metabolic pathway of Figure 6.1; bottom - a bipartite graph representation of the same pathway. Courtesy Falk Schreiber.

6.1.2 Graph Drawing for Metabolic Pathways

Three structural characteristics of metabolic pathways are particularly relevant for visualisation. They are typically:

Small in size — textbook pathways usually consist of less than two-dozen reactions.

Sparse — because most substances are involved in only a few reactions.

Directed — reaction edges are directed from reactants to products, though they may be bi-directional.

Have few cycles — due to a dominant direction of most reactions. However, a small number of cyclic pathways do exist such as the well-known citrate cycle.

Force-directed methods (see Section 3.2.1) can provide reasonable arrangements of metabolic pathway graphs, for example [111, 12, 174]. However, the structural characteristics of metabolic networks make them particularly amenable to hierarchical graph drawing methods (see Section 3.2.1). For example systems such as BIOMINER [174] and BIOPATH [169] include special extensions exploiting these structural characteristics to produce high quality drawings of metabolic pathways. That is, they handle (for example) highly variable node sizes, sparse pathways, and the occasional cycles that appear in metabolic pathways.

This chapter applies the hierarchical layout method modified for $2\frac{1}{2}$D visualisation as described in Section 4.3.2 and examines various methods for mapping metabolic pathway elements into the third dimension. Although the stratified hierarchical layout method has not been specifically modified for metabolic pathways, any of the specialised techniques described above, would also be applicable in finding a 2D layout for the union-graph.

6.1.3 Metabolic pathway data

The sources of the metabolic pathway data used in the applications in this chapter are the KEGG LIGAND database [84] and the BIOPATH system [69]. The KEGG LIGAND database contains a number of small pathways related to specific processes. To build larger networks of interest to the biologists involved in this work, two steps were applied:

1. The data from these databases was transformed into graphs and stored as GML files. GML (Graph Modeling Language) is a widely used and easily extensible exchange format for graphs [94, 93]. Several smaller networks were merged into one large network of metabolic reactions.

2. For the application involving experimental data (Section 6.2), the relevant partial network was identified. This network is defined by the metabolites for which experimental data exists and the main connections between these metabolites.

A difficulty in using these online metabolic databases is that, due to the rapid development of the field and the difficulty in maintaining the databases, the data is often of poor quality. For example, in preparing the metabolic pathway graphs the following issues were encountered:

- The merging of data from different sources introduced inconsistencies into the network such as different names for the same metabolite.

- In corroborating the data with experts, species specific physiological aspects were found that are not correctly represented in the above-mentioned general metabolic pathway databases.

These problems had to be resolved by labouriously editing the files. Wittig et al. [200] highlight the need for better quality online metabolic pathway databases. In future, metabolic pathway visualisation tools, such as those presented in this chapter, may prove useful in exploring these databases simply to find and correct such erroneous data.

6.2 Representing Experimental Biological Data in Metabolic Networks

Sections 1.4.2 and 6.1 gave an introduction to metabolic pathways and current methods for visualising these pathways. Consideration is now given to the problem of overlaying these visualisations with data such that they can convey quantitative information about the functionality of the pathways. This problem was tackled with assistance from scientists at the IPK crop plant research centre in Gatersleben, Germany. For these scientists, measuring the amount of metabolites consumed in various metabolic processes is crucial to learning about how a plant's genes affect metabolic function. This field is known as *plant functional genomics*.

To analyse major metabolites of a primary metabolism (e.g. sugars, sugar alcohols, amino acids, intermediates of glycolysis and the citrate cycle, nucleotides and their sugars) enzymatic and chromatographic methods are widely used, and have been available for some time. However, more recently, new tools for metabolic profiling have become available. For example, in [27, 65, 162] mass spectroscopy is coupled to liquid and gas chromatography. Such techniques mean that more quantitative data, showing the function of metabolites in various processes, is available than ever before. Hence, the problem of trying assimilate this new data into the expanding body of knowledge about the metabolic networks is becoming more complex.

To visualise experimental data several standard methods are regularly used, for example: displaying data in tables, histograms and line-graphs. A typical example of traditional visualisation techniques applied to metabolic data is shown in Figure 6.7. Also, new approaches have been developed such as visualisation of gene expression micro-array data [204]. Comparison of single

metabolites may give a detailed view of individual aspects. However, to recognise causal relationships within the metabolic network the researcher must mentally map experimental data back to positions of metabolites in the pathway network. To represent the data within this network directly more sophisticated methods are necessary.

In [144] the metabolic pathway diagrams of the KEGG system (see Section 1.4.2) were linked to the EXPRESSION database for integration with DNA micro-array data. Wolf et al. [202] use manually prepared visualisations of metabolic pathways from KEGG and in-house sources, and display protein and mRNA expression data in these diagrams such that colours indicate the relative change in expression level and reproducibility. However, as discussed in Section 1.4.2, manually arranged pathway visualisation has several drawbacks which negatively impact on this particular application.

In this work automatic network visualisation is combined with mapping of experimental data — especially time-series data about metabolites — using a stratified graph visualisation approach. The goal is to show how changes over time in metabolite data fit into the network.

This novel approach allows an easy visual analysis of processes in organisms and may assist users in the analysis of large data sets from biological experiments. This section is structured as follows: Section 6.2.1 describes how the experimental data is mapped onto metabolic pathway graphs; Section 6.2.2 deals with the novel visualisation method and some implementation aspects; finally, in Section 6.2.3 the approach is used for the visual analysis of experimental data from the seed development of barley.

6.2.1 Data from Biological Experiments

Starting with the metabolic pathway graph of interest as a GML file, experimental data can be added to the metabolites and reactions. A GML file consists of a hierarchical key-value list: a key is a sequence of alphanumeric characters (e.g. `graph`, `id`), and a value is a string, an integer, a floating-point number, or a list of key-value pairs. Using GML it is possible to attach additional information to every object. For example, if such information exists for a specific metabolite (the metabolite was measured and quantified over several days), its corresponding node is assigned the data about the measured amount on the different days. Quantitative time-series data associated with specific reactions may be similarly associated with edges. An example of a GML file representing a network and additional time-series data is shown in Figure 6.3.

```
graph [
   node [
      id 1
      label "Sucrose"
      experimental_data [
         day "day 0"
         value 1534
         day "day 2"
         value 2801
         day "day 4"
         value 2914
      ]
   ]
   node [
      id 2
      label "Fructose"
      experimental_data [
         day "day 0"
         value 2341
         day "day 2"
         value 2894
         day "day 4"
         value 2786
      ]
   ]
   edge [
      id 1
      label ""
      experimental_data [
         day "day 0"
         value 54
   ...
      ]
   ]
]
```

Figure 6.3: An example of a GML file incorporating experimental data.

6.2.2 Visualisation method

The focus for this section is to visualise the structure of a metabolic network as well as showing the amounts of metabolites and the flow within the network as they change over time. That is, the third dimension is now mapped to the ordinal variable of time.

The visualisation shown in Figure 6.5 was produced using the WILMASCOPE 3D graph visualisation tool as described in Section 4.3 and Appendix B. In this visualisation each level or slice in the $2\frac{1}{2}$D structure corresponds to sample data taken on a particular day, with the oldest data (day 0) on the bottom slice and the most recent (day 20) at the top. The small window floating before the WILMASCOPE window in Figure 6.5 shows a cross-section through the lowest slice of the stack and corresponds with the semi-transparent blue plane that can be seen in the $2\frac{1}{2}$D view. This "water-level" can be moved up or down to highlight an individual cross section of the graph. Cross-sections allow the user to clearly see the amount of each metabolite present at a particular point in time, while the $2\frac{1}{2}$D view shows an overview of the whole time series.

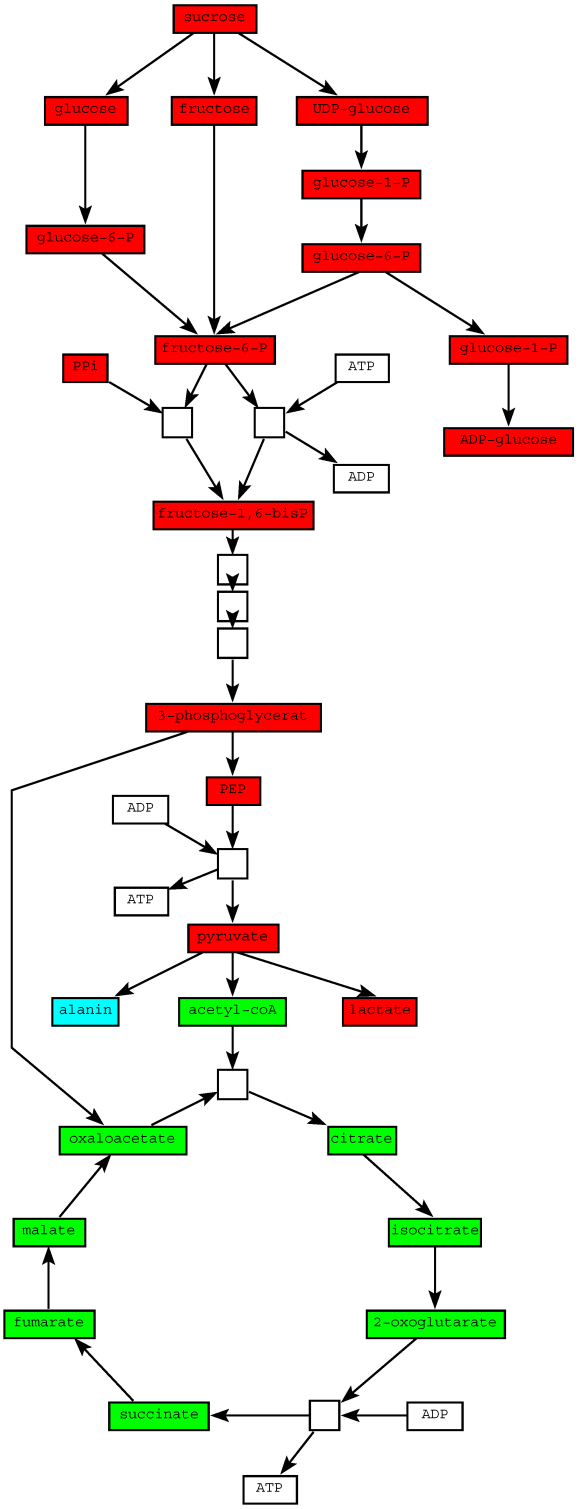


Figure 6.4: The metabolic network for the experimental data described. Different colours are used to distinguish different pathways in the network such as glycolysis and TCA cycle.

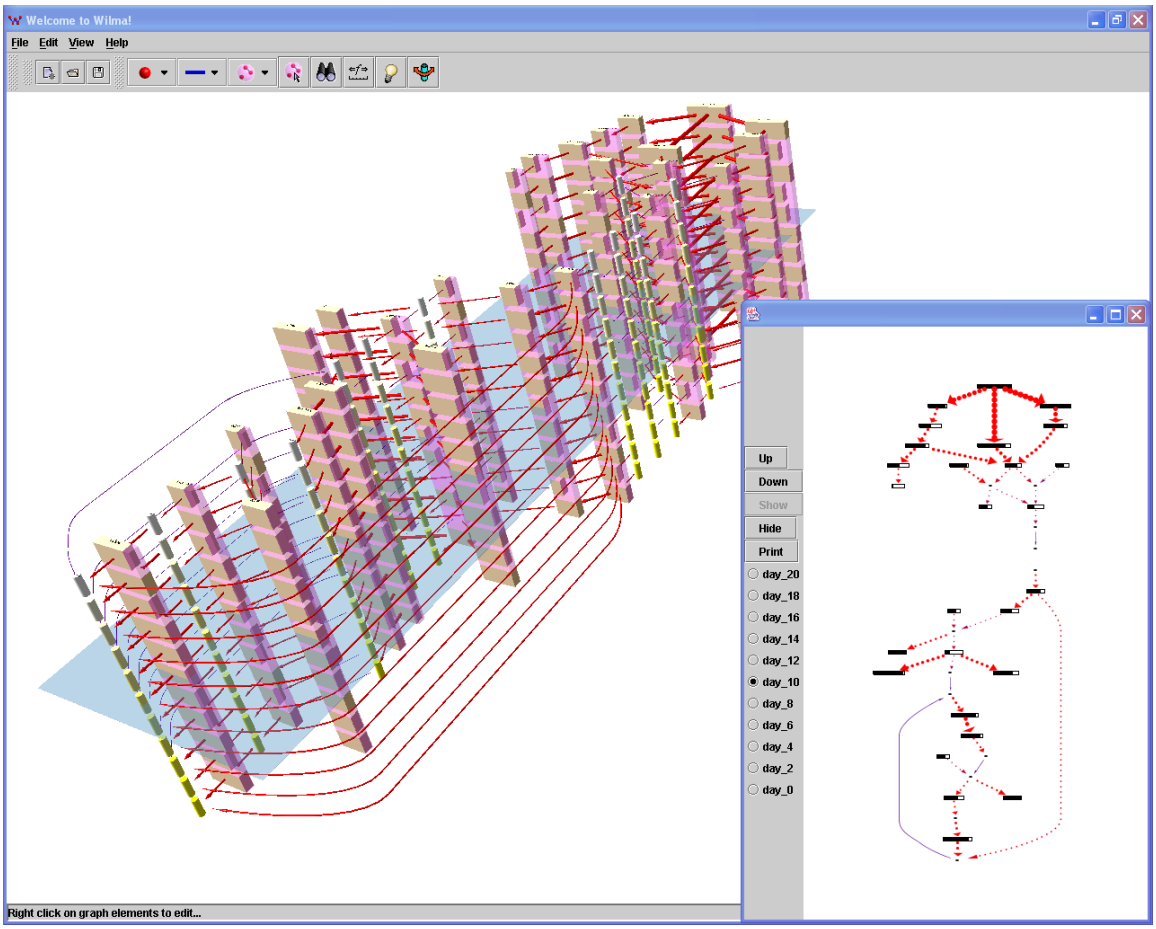
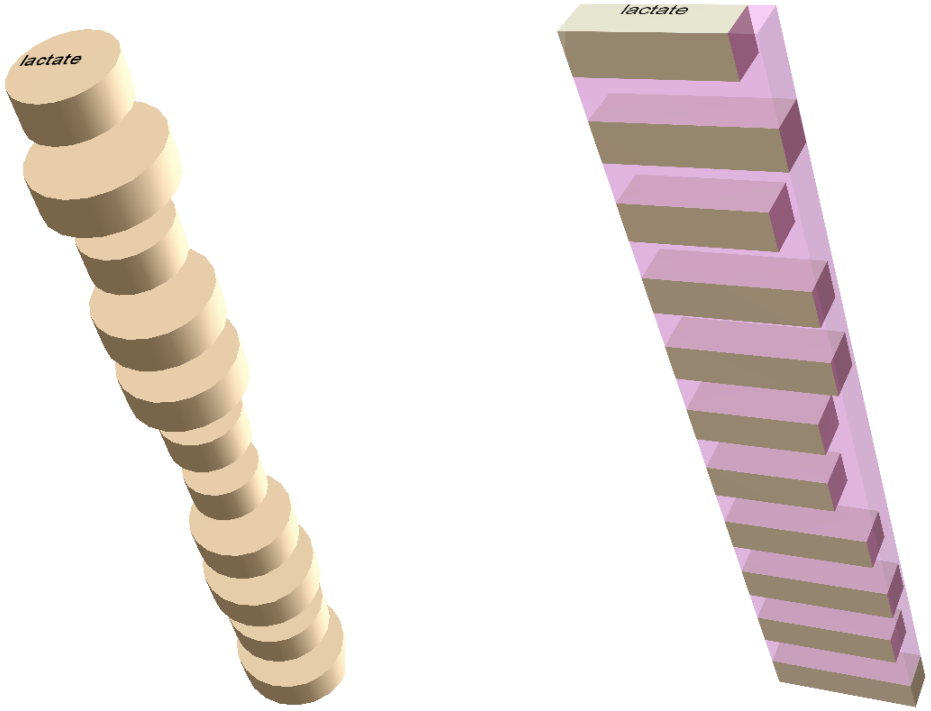


Figure 6.5: A screenshot of the WILMASCOPE system examining a $2\frac{1}{2}$D visualisation of the metabolic network from Figure 6.4

Each metabolite can be shown as a histogram giving the measured values at each point in time. Where time-series data for a particular metabolite is not available, or is not interesting to the user, the metabolite is shown as a narrow column. Figure 6.6 shows two possible $2\frac{1}{2}$D representations for a single metabolite node with associated time-series data. Each section represents the measured value of the metabolite on a particular day. In the representation shown in Figure 6.6(a), a square-root scale is used to determine the radius such that the cross-sectional area of the disc is directly proportional to the amount of the metabolite. In the histogram-like representation of Figure 6.6(b), a log scale is used to better fit the bars into the space available. Note that the transparent pink box makes it easy to compare each bar against the maximum value, or another benchmark such as the median or mean value.

Figure 5 shows the complete network using the disc representation. Static screen shots of



(a) The cross-sectional area of each disc in the stack represents the amount of metabolite present in one day's sample

(b) A more conventional histogram style representation using a log scale for the width of each bar to show the amount of metabolite present. The transparent bounding box allows for easy comparison of each bar to the maximum value.

Figure 6.6: Two possible 3D representations for the lactate metabolite node from Figure 6.4 showing quantities in the time-series data.

perspective projections of 3D models are less effective at conveying 3D structure than interactive visualisations. Figure 6 shows the network projected in parallel from several different angles to better display the strata in the $2\frac{1}{2}$D structure.

The diameter and colour of the reaction edges can also give an indication of flow given by the activity and quantity of enzymes (this could, for example, be derived from expression data). Note that in the application example (Section 6.2.3) at the time of writing such information is not yet available from the experiments of the biochemists involved in this work. Therefore, in the visualisation the value represented by edge widths is an estimate of flow based on the quantity of source and sink metabolite. It is shown here purely to illustrate the potential of the visualisation paradigm.

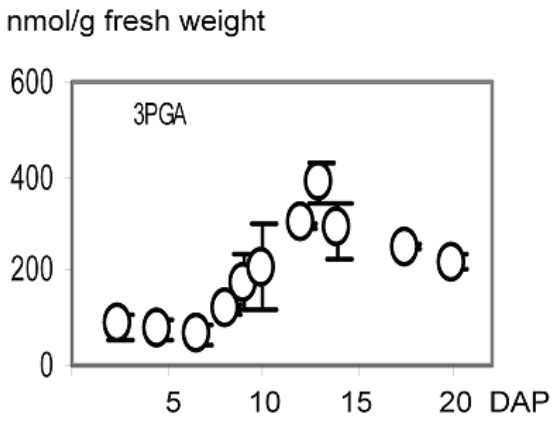


Figure 6.7: A typical chart of time-series data showing the metabolite 3PGA at several days post anthesis (DPA). Courtesy Hardy Rolletschek.

6.2.3 Application Example

Metabolic Data

The quantitative experimental data used in these visualisations was provided by biologists at the IPK Crop Plant Research centre. To investigate the metabolic processes of barley (*Hordeum vulgare*) seeds (or *caryopses*) in different stages of development, the biologists use mass spectroscopy, coupled to liquid and gas chromatography techniques, to measure the amount of metabolites present at different days in the maturation process. To do this, they harvest a sample of seeds and measure the metabolites present every second day over a growth period of about 20 days. About 70 metabolites were measured and quantified. A typical example of this time-series data, and the conventional visualisation method used by biologists to study it, is given in Figure 6.7.

Visual Analysis of the Experimental Data

Two different node-scaling methods were used for analysing experimental data in the context of metabolic networks. The different approaches suit slightly different types of analysis.

In the first approach, node size was normalised such that the maximum width of the node for each metabolite was the same across all metabolite nodes. Thus, the size of the node on a particular stratum indicates the amount, as a percentage of the maximal value, of a specific metabolite on a specific day. This approach emphasises the relative change of metabolites over time and the relative flow through the network. An example of this visualisation style is given in Figure 6.10.

In the second approach, the sizes of nodes corresponded directly to the absolute amount of the

corresponding metabolites. This allows the analyst to study the absolute ratio of different metabolites involved in different parts of the pathway. An absolute node size example is shown in Figure 6.11.

6.3 Comparison of Related Metabolic Pathways

The evolutionary relationships among species are usually computed by phylogenetic analysis of protein or DNA sequences. This analysis results in a phylogenetic tree where nodes represent species and edges represent ancestry relationships. More recent methods have been based on comparing higher-level functional components such as metabolic pathways [68, 125, 185]. This section demonstrates an approach for visualising several related metabolic pathways in such a way that the inherent differences can be explored by trained biologists in order to understand the evolutionary relationships among species.

Metabolic pathways differ across organisms because different species may, for example, have developed different ways to synthesise a specific substance. Studies suggest significant variations even in the most central pathways such as *glycolysis* [37]. Comparative analysis of pathways across species has several applications:

- Understanding the evolutionary relationships between species.
- Development of species-specific drug targets (e.g. antibiotics).
- Identification of previously unknown parts of pathways in a species.

This Section proposes visualising sets of related pathways in $2\frac{1}{2}$D. That is, interdependent, 2D, layered layouts for all pathways are produced and stacked into the third dimension so that the most similar pathways are adjacent.

To realise such a design, a new stratified graph issue needs to be addressed. That is, it is necessary to determine a suitable ordering to reduce the variation between consecutive pathways. Existing distance measures — based on a-priori knowledge about the pathways — could be used to compute such an ordering. However, a new measure for graph pathway similarity is introduced here, which only depends on the structure of the pathway graph.

This section is organised as follows. Section 6.3.1 defines the method used to measure similarity between pathways. The visualisation design is specified in Section 6.3.2. Section 6.3.3 addresses

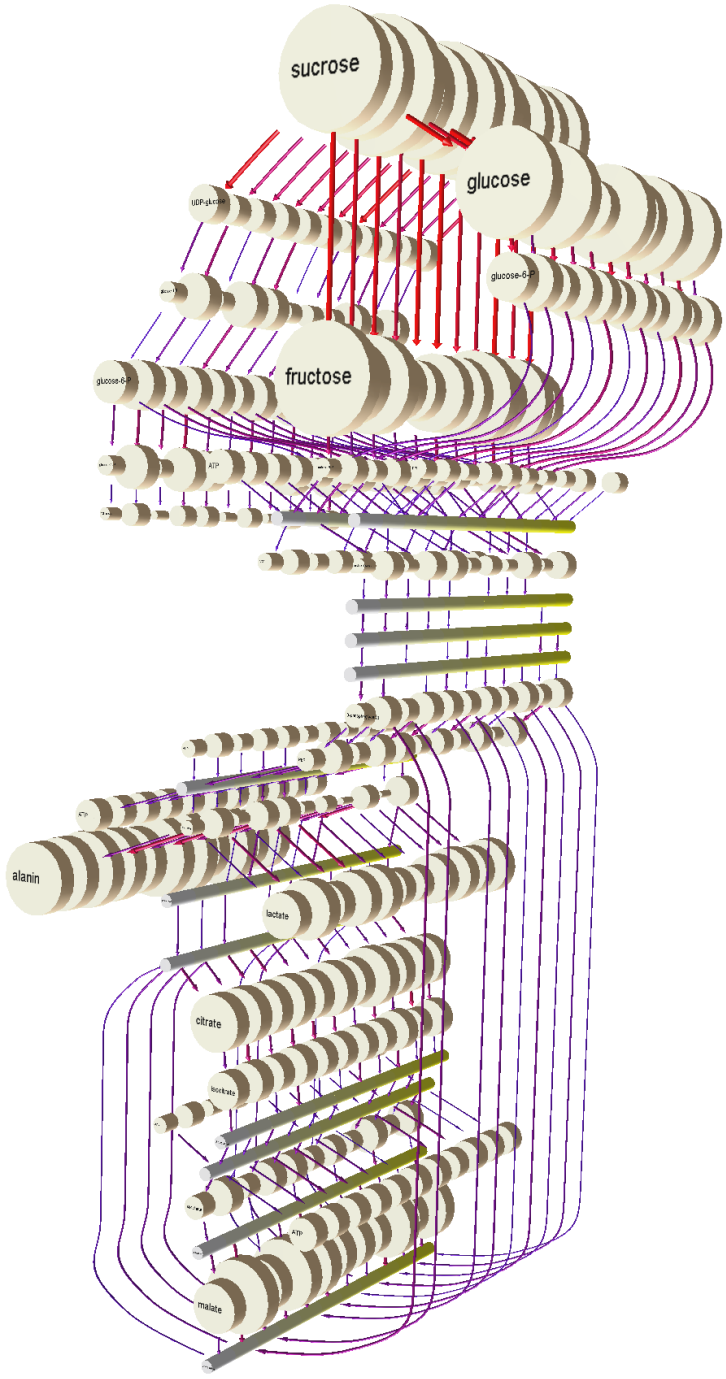


Figure 6.8: A view of the $2\frac{1}{2}$D visualisation of the network using the "disc" representation for the metabolites.

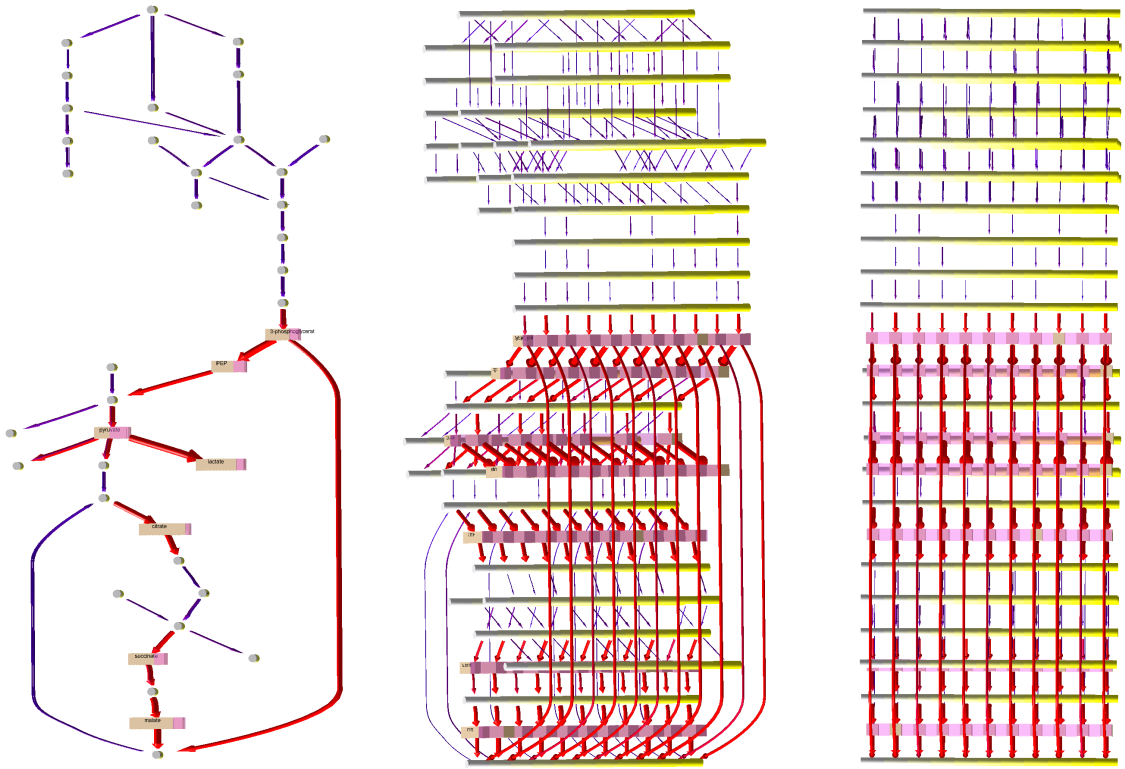


Figure 6.9: Three parallel projected views of the network, viewed from three different orientations to give the reader a better understanding of the $2\frac{1}{2}$D stacking method. WILMASCOPE allows interactive navigation to explore the stacked network from all directions

the problem of finding a stacking order for the set of pathways based on the similarity measure used. To demonstrate the utility of the method it is applied to typical real-world data in Section 6.3.4.

6.3.1 Similarity measures

Several similarity measures have been introduced to compare pathways. They are characterised by the combination of structural information about metabolic networks with additional data, such as sequence information [68], the enzyme classification hierarchy [185], or information about the hierarchical clustering of reactions into pathways [125]. All these similarity measures require additional information to the metabolic network, for example, the genome sequence of the organisms.

A more general measure is suggested for reaction or pathway similarity, which only depends on the structure of the graph, that is, on the presence or absence of nodes and edges. This facilitates comparison of metabolic pathways from different sources (databases or experiments) even if no additional data is available or the pathway boundaries are user-defined. Note that, in general, the

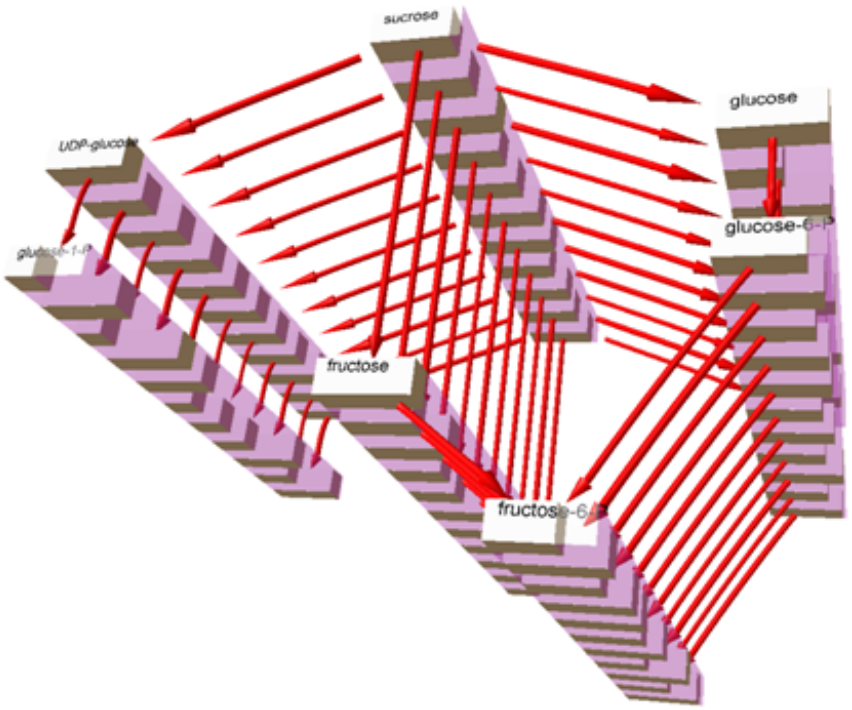


Figure 6.10: Detail of the neighbourhood around sucrose using the fixed node width style.

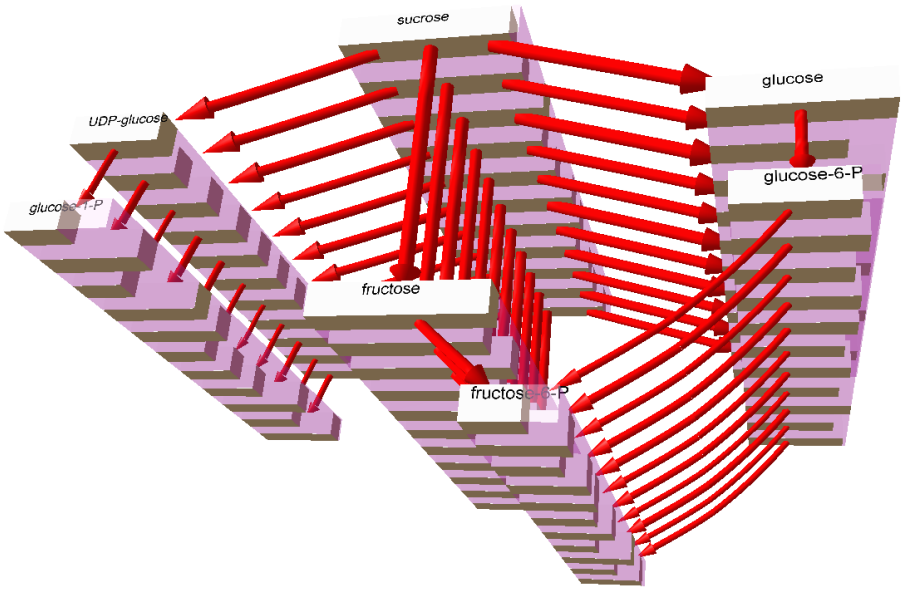


Figure 6.11: The sub-network from Figure 6.10 shown with absolute node size.

visualisation method is independent of the similarity measure which is only used to compute the order of the stacking.

Formalisation

Recall from Definition 6.1.1, the bipartite definition of a metabolic pathway graph, $G_b = (U_1, U_2, E)$. The similarity measure that follows does not distinguish between a metabolite node $u_1 \in U_1$ and a reaction node $u_2 \in U_2$. In the sequel, therefore, a non-bipartite definition is considered for a pathway graph $G = (V, E)$ where $V = U_1 \cup U_2$. G is a labelled graph in the sense that each node $v \in V$ represents either a distinct substance or reaction.

The goal is to visualise a set of graphs that represent related pathways:

$$\{G_1 = (V_1, E_1), G_2 = (V_2, E_2), \ldots, G_r(V_r, E_r)\}$$

To express the dissimilarity of two pathways $G_i = (V_i, E_i)$, $G_j = (V_j, E_j)$ the following measure is used. Consider the *union graph*:

$$G = (V, E), \qquad \text{where} \quad V = \bigcup_{i=1}^{r} V_i \quad \text{and} \quad E = \bigcup_{i=1}^{r} E_i$$

and a set of relevant elements $P \subseteq (V \cup E)$. Many metrics exist for measuring the difference between two labelled graphs. *Hamming distance* is traditionally used in information theory to measure the number of bits that differ between two binary strings. Different definitions of the Hamming distance between two graphs exist. A Hamming distance between two graphs G_i, G_j, $1 \leq i, j \leq r$, is defined as the cardinality of the symmetric difference of relevant elements present in either graph. That is:

$$\delta_P(G_i, G_j) = |((V_i \cup E_i) \triangle (V_j \cup E_j)) \cap P)| \,.$$

Thus, dissimilarity can be defined in terms of missing edges, nodes, or both, by choosing P accordingly. Further, P could be selected to define regions of interest or specific views of the pathway such as the enzyme graph model used in [149] where enzymes are the nodes of the hyper-graph and substances are the hyper-edges.

6.3.2 Visualising Similar Networks

There are two common approaches to comparing pathways in different species visually. One can either combine all pathways into one diagram or a separate drawing can be produced for each species.

The first method is used in many textbooks; on the *Biochemical Pathways* poster [136]; and in systems such as BIOMINER [174] and BIOPATH [69]. In general, the drawings contain either multiple (parallel) reaction-edges or single ones which are colour-coded depending on the occurrence of a reaction in a set of species. An example for the second approach is the visual interface of the KEGG database [110] where all enzymes found in the gene catalog of a specific species are marked in the reference pathway map in order to identify the species-specific pathways. To compare pathways in r different species, r diagrams are needed. A visual comparison method producing a diagram for each species is presented by Schreiber in [170].

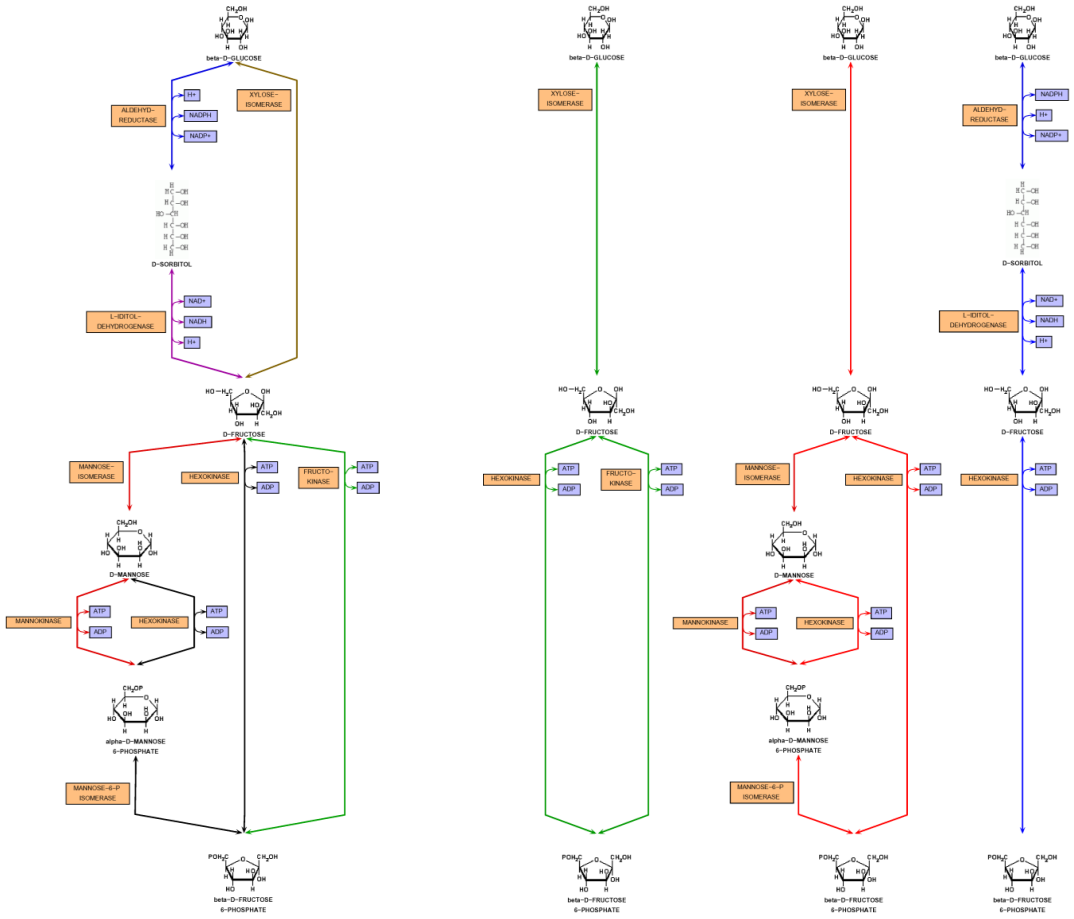
These solutions are restricted to the comparison of only a few pathways, because either (in the first approach) the readability of the diagram decreases or (in the second approach) the size of the picture increases dramatically with each new pathway; see Figure 6.12. Furthermore, none of the above mentioned methods deal with the problem of computing an appropriate ordering of the pathways.

6.3.3 Stacking Order

Let $\{G_1 = (V_1, E_1), G_2 = (V_2, E_2), \ldots, G_r(V_r, E_r)\}$ be a set of pathway graphs, and $G = (V, E)$ their union graph. Let $P \subseteq (V \cup E)$ be a set of relevant elements. When producing a stratified $2\frac{1}{2}$D visualisation of this set of graphs by stacking them into the third dimension, where there is no obvious predefined ordering of the pathways, we are free to choose a stacking order. Since the aim here is to allow biologists to compare pathway graphs, an ordering is chosen that supports visual understanding of the similarities between the graphs. A logical ordering, therefore, would place similar pathway graphs adjacent to one another with the following objectives:

Objective 1 — the distance between two pathway graphs in the stack provides a visual cue as to the difference, in terms of missing elements, between the two graphs;

Objective 2 — by finding a stacking order that maximises the number of times



(a) Combination of all pathways into one diagram

(b) One diagram for each species

Figure 6.12: Visual comparison of metabolic pathways, courtesy Falk Schreiber [170].

each element appears in consecutive graphs, the analyst's attention is focused on elements that are unique to each graph.

Therefore generally, it is desirable to order these graphs so that those which are similar with respect to Hamming distance, are close to each other. Variations of this problem arise in many applications and two of them are especially relevant in the context of this work. The first is slightly more related to Objective 1 and the second is more related to Objective 2.

Problem 6.3.1 (MIN SUM GRAPH STACKING) *Find the stacking order where the total hamming distance between all adjacent graphs is minimal:*

INSTANCE: *A set of graphs* $\{G_1, G_2, \ldots, G_r\}$ *and a permutation of this set* $\sigma =$

$(\sigma_1, \sigma_2, \ldots, \sigma_r)$.

QUESTION: *Find σ such that:*

$$\sum_{i=1}^{r-1} \delta_P(G_{\sigma_i}, G_{\sigma_{i+1}})$$

is minimised.

Murgai et al. [143] consider the closely related problem of ordering binary strings such that there is minimal Hamming distance between adjacent strings. In the context of transmitting data words over a bus where order of the words is irrelevant, such a minimum transition ordering reduces power usage. A less restricted version (with an arbitrary distance matrix) is considered by Keim in [114], where the goal is to order the axes in parallel coordinates visualisations such that the positions of data tuples on adjacent axes are as similar as possible.

Theorem 6.3.2 *MIN SUM GRAPH STACKING is $\mathcal{NP}$-hard.*

Proof Straightforward reduction from HAMMING DISTANCE TSP (Travelling Salesman Problem).

Alizadeh et al. [3] discuss HAMMING DISTANCE TSP further with respect to ordering the rows of binary matrices such that adjacent rows have minimal Hamming distance. Equivalence of MIN SUM GRAPH STACKING to the TSP is easy to see. Essentially, each city in the TSP is a graph. The distance between cities is the Hamming distance metric. A tour of all cities, visiting each precisely once, corresponds to stacking order for the graphs.

Objective 2 is to maximise the number of consecutive graphs in which particular elements are present. For a permutation $\sigma = (\sigma_1, \ldots, \sigma_r)$, the *lifetime* of an element $p \in P$ is defined to be $\Lambda_\sigma(p) = \{1 \leq i \leq r \,:\, p \in G_{\sigma_i}\}$. An element $p \in P$ is called *persistent*, if its lifetime spans the entire interval $\{1, \ldots, r\}$, and *transient* otherwise. The number of *appearances* and *disappearances* of $p \in P$ are defined by

$$a_\sigma(p) = |\{1 < i \leq r \,:\, p \in G_{\sigma_i} \setminus G_{\sigma_{i-1}}\}| \quad \text{and}$$

$$d_\sigma(p) = |\{1 \leq i < r \,:\, p \in G_{\sigma_i} \setminus G_{\sigma_{i+1}}\}|.$$

Note that persistent elements make no appearances or disappearances.

Corollary 6.3.3 *MIN SUM GRAPH STACKING is equivalent to minimising*

$$\sum_{p \in P} \big(a_\sigma(p) + d_\sigma(p)\big) \ .$$

A related alternative objective, which is more highly motivated by Objective (2) than Objective (1), is therefore:

Problem 6.3.4 (MIN INTERVAL GRAPH STACKING) *minimise the maximum number of times an element appears or disappears in an ordering:*

INSTANCE: *A set of graphs* $\{G_1, G_2, \ldots, G_r\}$ *and a permutation of this set* $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_r)$.

QUESTION: *Find* σ *such that:*

$$\max_{p \in P} \{a_\sigma(p), d_\sigma(p)\}$$

is minimum.

This problem is a generalisation of the *consecutive ones property*. This is apparent if a matrix M is defined with columns for each element $p_i \in P$ ($P = \{p_1, p_2, \ldots, p_n\}$) and a row for each graph G_j. Set each entry M_{ij} to 1 if the element p_i exists in G_j and to 0 otherwise. The k-consecutive ones property holds for a $(0, 1)$ matrix if there exists a row order such that in each column the occurrences of all ones appear in at most k consecutive blocks. Since, if $p \in P$ is transient, $\max\{a_\sigma(p), d_\sigma(p)\}$ is the number of lifetime intervals (and zero otherwise). The problem then is to find the smallest k for which the k-consecutive ones property holds. This problem has the following complexity status.

Theorem 6.3.5 *MIN INTERVAL GRAPH STACKING is $\mathcal{NP}$-hard, but it can be determined in linear time*[4] *whether there is an ordering such that each relevant element appears or disappears at most once: that is, the k-consecutive ones property where $k = 1$.*

[4]Time complexity of testing for the consecutive ones property with PQ-trees is $O(|P|+r+m)$ where r is the number of pathway graphs (or rows in matrix M) and m is the total number of times each element in P is present in all graphs (or the number of non-zero entries in M).

Proof Goldberg et al. [82] give a proof of $\mathcal{NP}$-completeness with discussion related to the k-consecutive ones problem for assembling fragments of DNA. Since the restricted problem corresponds exactly to the consecutive ones property, it is linear-time solvable using PQ-trees [13].

Since optimisation problems similar to MIN SUM GRAPH STACKING and MIN INTERVAL GRAPH STACKING arise in many contexts, a variety of algorithms is available to determine an ordering. For MIN SUM GRAPH STACKING, for instance, heuristics for the TSP are easily adapted to yield good orderings. Other alternatives include a simple greedy heuristic that successively inserts a new element where it causes the smallest increase of the objective (this method is claimed to perform well for instances in data transmission [143]). Also, a one-dimensional projection of the distance matrix obtained by principal component analysis can provide a reasonable solution. Note that this process is similar to the 2D PCA projection of a distance matrix discussed in Chapter 5, but projecting onto only one eigenvector rather than two.

6.3.4 Application Examples

The utility of this approach is demonstrated on two sets of pathways extracted from the KEGG database. Both examples include detail of the Hamming distance calculation and the MIN SUM GRAPH STACKING, the second example shows some additional graphical features.

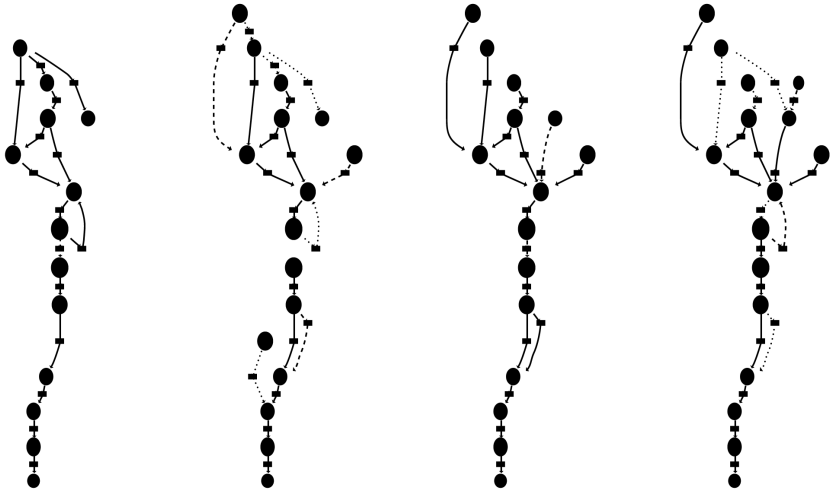
Example 1

The first data-set consists of parts of the *glycolysis* and *fructose/mannose* metabolism pathways in seven organisms that show significant differences.

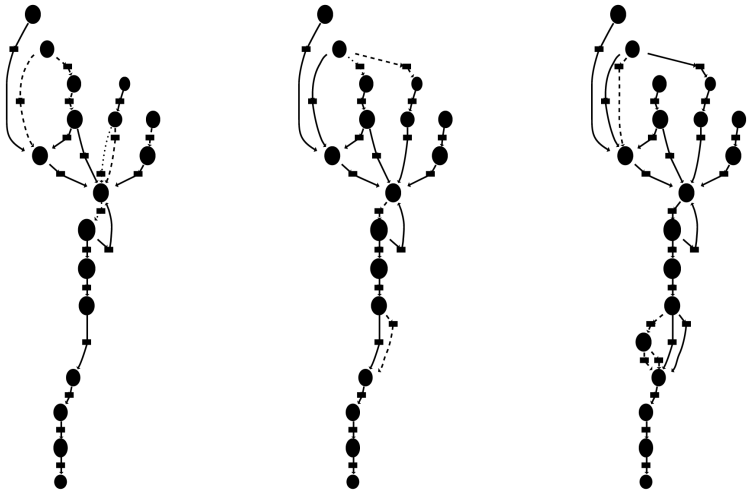
Table 6.1 gives Hamming distances between these pathways with all elements $P = (V \cup E)$ considered relevant. The order in which the organisms are listed is optimal with respect to MIN SUM GRAPH STACKING and was computed by enumeration.

Using the adapted version of the dot program described in Section 4.3.2, a layout of the union graph of these seven pathways was computed. The resulting individual layouts are shown in Figure 6.13.

The $2\frac{1}{2}$D representations shown in Figures 6.1 and 6.12 have been created with the WILMASCOPE 3D graph visualisation system. Edge appearances and disappearances are colour-highlighted using green and red, respectively. By moving a semi-transparent plane through the image, users can navigate forward and backward in the similarity-ordered sequence of pathways.



(a) H. Influenzae (b) E. Coli (c) S. Pyogenes (d) B. Subtilis



(e) A. Thaliana (f) D. Melanogaster (g) H. Sapiens

Figure 6.13: Layouts of individual pathways obtained from a union graph layout in the computed order (from (a) to (g)). Appearing edges are shown dashed, disappearing edges dotted.

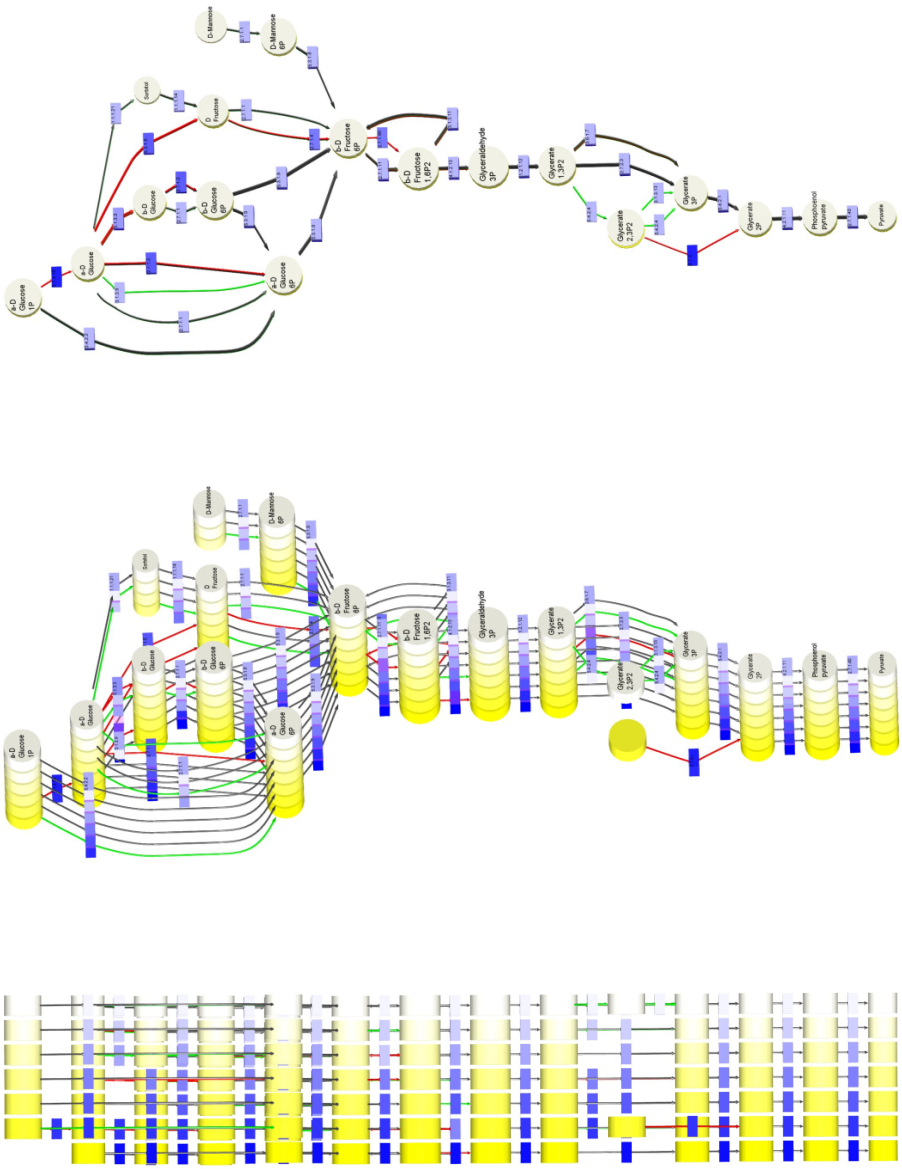


Figure 6.14: $2\frac{1}{2}$D drawing of seven related pathways (parallel projection).

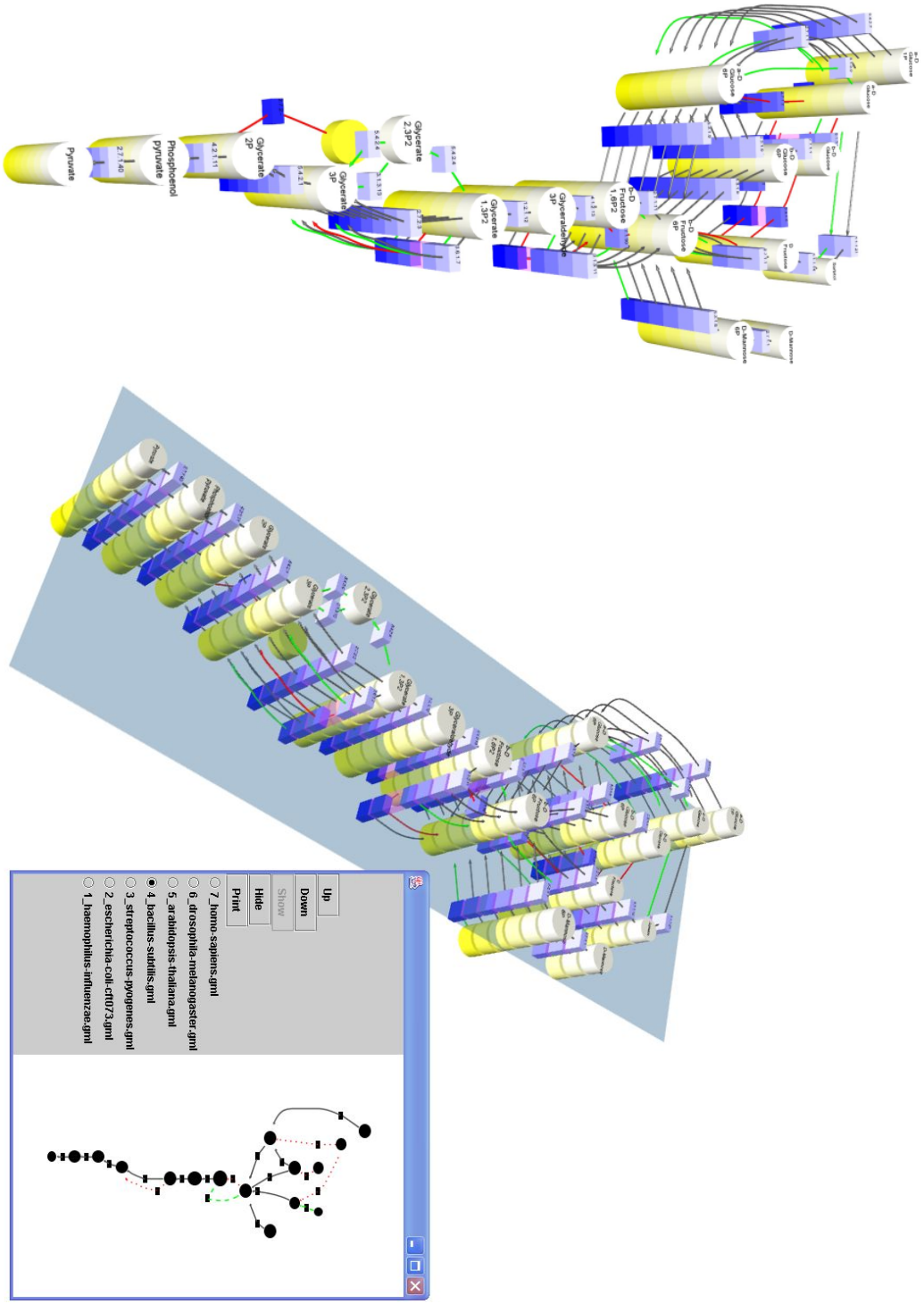


Figure 6.15: WILMASCOPE screenshots showing perspective projection and the cross-section viewer for interactive exploration

Table 6.1: Hamming-distance matrix for parts of the glycolysis and fructose/mannose metabolism pathways from seven different species

	(a)	(b)	(c)	(d)	(e)	(f)	(g)	
(a)	0	21	23	21	43	40	56	*Haemophilus influenzae*
(b)	21	0	22	20	48	39	53	*Escherichia coli CFT073*
(c)	23	22	0	10	38	35	45	*Streptococcus pyogenes*
(d)	21	20	10	0	34	31	41	*Bacillus subtilis*
(e)	43	48	38	34	0	15	31	*Arabidopsis thaliana*
(f)	40	39	35	31	15	0	16	*Drosophila melanogaster*
(g)	56	53	45	41	31	16	0	*Homo sapiens*

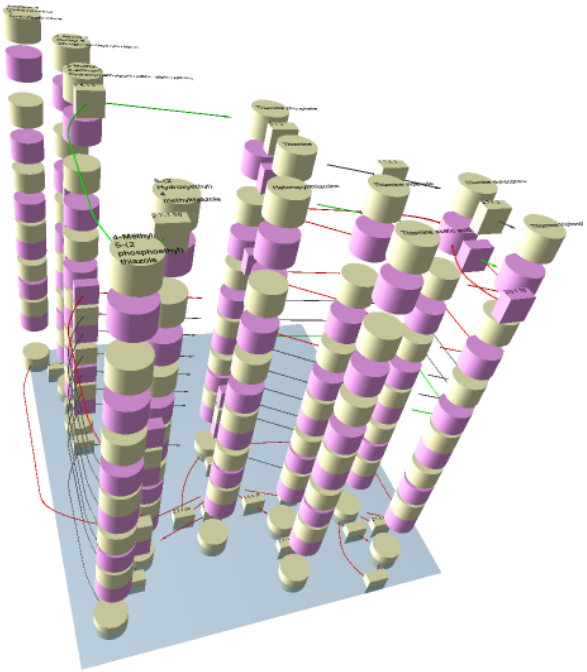
	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	(k)	(l)	
(a)	0	13	9	7	9	3	12	15	3	19	3	10	*Yersina pestis*
(b)	13	0	10	12	10	16	13	22	10	18	16	3	*Methanobacterium thermoautotrophicum*
(c)	9	10	0	16	0	6	3	18	6	28	6	7	*Archaeoglobus fulgidus*
(d)	7	12	16	0	16	10	13	16	10	12	10	9	*Treponema pallidum*
(e)	9	10	0	16	0	6	3	18	6	28	6	7	*Pyrococcus horikoshii*
(f)	3	16	6	10	6	0	9	12	6	22	0	13	*Haemophilus influenzae*
(g)	12	13	3	13	3	9	0	15	9	25	9	10	*Arabidopsis thaliana*
(h)	15	22	18	16	18	12	15	0	18	10	12	25	*Saccharomyces cerevisiae*
(i)	3	10	6	10	6	6	9	18	0	22	6	7	*Mycobacterium leprae*
(j)	19	18	28	12	28	22	25	10	22	0	22	21	*Mus muculus*
(k)	3	16	6	10	6	0	9	12	6	22	0	13	*Bacillus subtillis*
(l)	10	3	7	9	7	13	10	25	7	21	13	0	*Aeropyrum pernix*

Table 6.2: Hamming-distance matrix of thiamine pathways

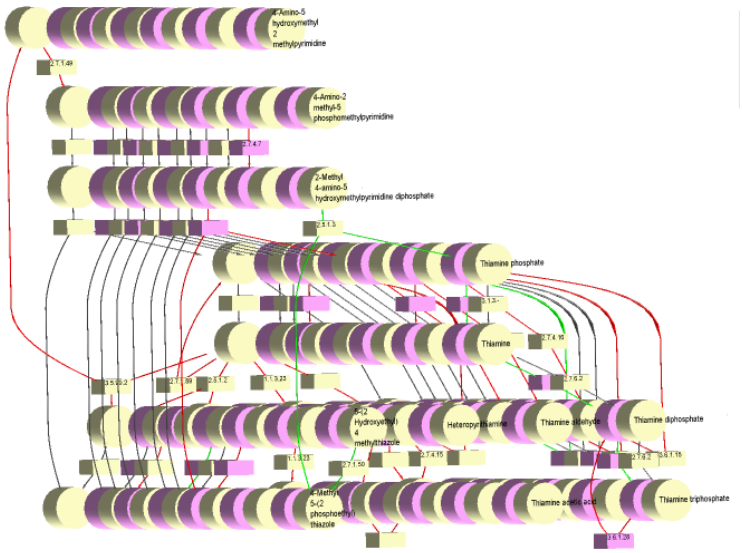
Example 2

The second example compares the known involvements of enzymes in the *thiamine* metabolism from 12 species against the complete pathway and each other. The set of relevant elements P for the computation of the Hamming distances is therefore restricted to nodes that represent enzymes and the edges that connect them to metabolites. The complete matrix of Hamming distances is shown in Table 6.2.

To make it easier for the user the general pathway for the Thiamine metabolism from KEGG is shown as the lowest stratum and neighbouring strata have contrasting colours, see Figure 6.16. Furthermore, the Hamming distances between pathways are used to compute a corresponding spacing between adjacent strata. This spacing was an outcome of discussions with biologists who suggested it as an additional visual cue aiding comparison of species, see Section 6.5.



(a) Perspective projection with the stratum representing the full metabolic pathway highlighted by the transparent blue plane



(b) Parallel projection which better shows the relative spacing between adjacent strata

Figure 6.16: WILMASCOPE screenshots for the Thiamine metabolism example.

6.4 Navigation by Visual Triangulation

As discussed in Section 1.4.3, phylogenetic trees are another way of exploring the relationships between species. Phylogenetic trees are built by examining differences in the biological traits of a set of species. An example of such a trait is a metabolic pathway, such as those examined in Section 6.3, common to a number of species but with subtle differences in each.

Phylogenetic analysis is an attempt to uncover the evolutionary relationships between organisms. It is an important tool in understanding evolutionary processes and in measuring genetic variations between species. Applications include the design of new drugs and reconstruction of the history of infectious diseases [96]. The result of phylogenetic analysis is a phylogenetic tree representing hypothetical ancestral relationships among a set of entities (see Figure 6.17). Established methods for phylogenetic analysis are based on morphological attributes or nucleotide and protein sequences [61, 67, 1].

Recently, new methods using metabolic pathway data have been introduced, and phylogenetic trees based on such data are becoming increasingly important [68, 92, 125]. Usually, the trees induced from metabolic data are visualised using conventional 2D drawing methods (see Figure 6.17) and are viewed in isolation from the underlying metabolic pathway graph structures corresponding to each node. However, in reality, the structure of the trees is dependent on similarities between these pathways and a method that allows both phylogenetic tree and pathway graph structures to be studied side-by-side may provide deeper insights.

A naïve approach is to draw the metabolic pathways inside the leaf nodes of their phylogenetic tree. Such diagrams, however, fail to convey the essential information in the data because viewers are not able to easily compare the similarities and differences between pathways.

In the social sciences the term *triangulation*[5] has been adopted to refer to the use of multiple complementary tools to study an object that is not easily understood when applying a single method. For example, one might seek correlations between qualitative and quantitative data, or data from different sources [106]. A visualisation approach based on the idea of triangulation is proposed here. In this approach different visualisations are shown for different aspects and a diagram of the phylogenetic tree is used as the main selection panel to determine the data shown in the other views.

This section is organised as follows. In Section 6.4.1 some background is given on the type of phylogenetic trees considered, together with a brief review of related approaches to visualising

[5] In surveying or navigation, *triangulation* is the process of using basic geometry to locate a point in a plane knowing only its direction from two other points of known position.

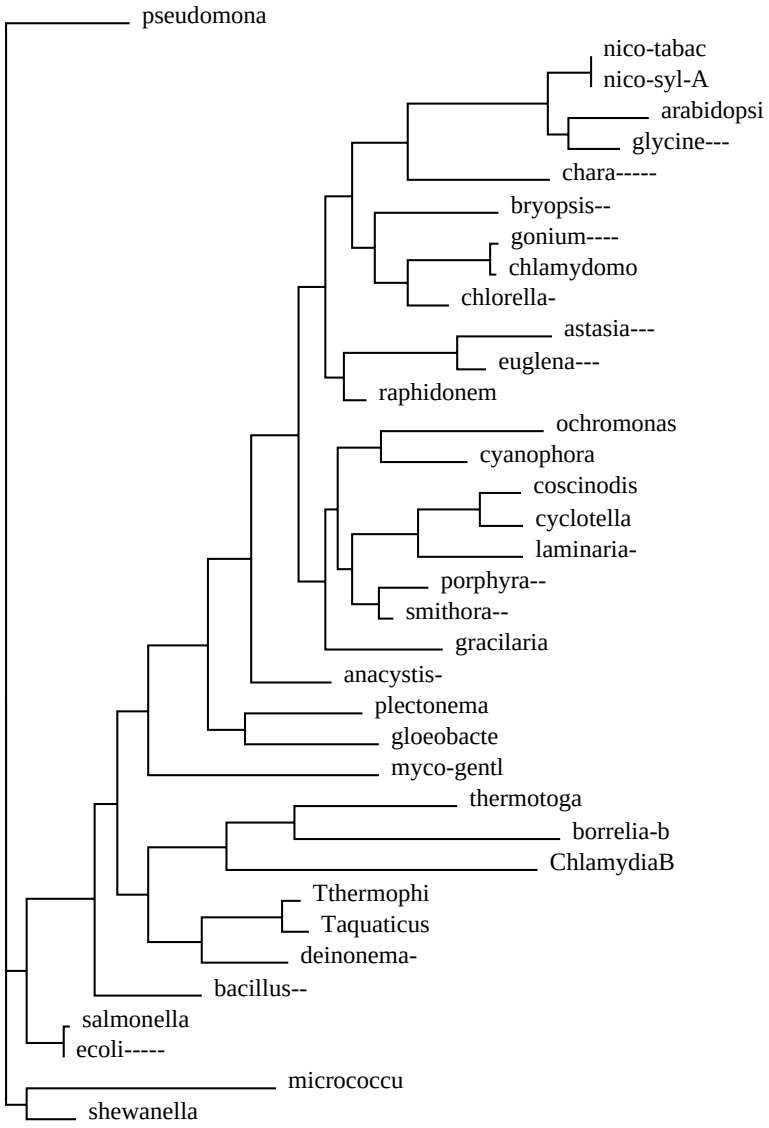


Figure 6.17: Dendrogram representation of a (simple) phylogenetic tree in which leaves represent species and internal nodes (or branching points) represent hypothetical ancestors. Branch lengths give an indication of evolutionary time. (Data from [39])

phylogenetic trees. The proposed method of visualisation is introduced in Section 6.4.2, and its application to typical real-world data demonstrated in Section 6.4.3.

6.4.1 Preliminaries

This section provides some background on the kind of data that is to be visualised.

Phylogenetic Trees

A *phylogenetic tree* $T = (S, L)$ is a tree consisting of a set S of nodes (species) and a set L of edges (links). Leaf nodes of the tree (that is, nodes having exactly one link) represent given species, sequences or similar entities; they are called *operational taxonomic units*. Internal nodes represent hypothetical ancestors generated from phylogenetic analysis; they are called *hypothetical taxonomic units*. See Figure 6.17 for an example of a conventional drawing of a phylogenetic tree.

Complex Phylogenetic Trees

Not only morphological attributes and nucleotide sequences differ across organisms, but also metabolic pathways. Studies show significant variations even in central pathways such as glycolysis [37]. Such variations can be used for phylogenetic analysis.

In Section 6.3 some approaches to measuring similarity across pathways are surveyed. Section 6.3.1 discusses a similarity measure depending only on the structure of the pathways. Once a similarity matrix is derived for a set of species, the process of deriving the phylogenetic tree is the same regardless of the source of similarities in the matrix, be it the more traditional gene sequence analysis or the metabolic pathway comparison.

The term *complex phylogenetic tree* is introduced to describe a phylogenetic tree of metabolic pathways. That is, a tree $T = (\{G_1, \ldots, G_k\} \cup \{H_1, \ldots, H_l\}, L)$ defined on directed metabolic pathway graphs as defined in Section 6.1.1. Leaf nodes $s \in \{G_1, \ldots, G_k\}$ represent given metabolic pathways (i.e. the operational taxonomic units), whereas internal tree nodes $s \in \{H_1, \ldots, H_l\}$ represent fictitious metabolic pathways (i.e. a hypothetical taxonomic unit) obtained from phylogenetic analysis. A tree edge $e \in L$ indicates a putative evolutionary relationship between the two incident pathways and can be labelled with a numerical value that serves as an indicator for distance in evolutionary time.

6.4.2 Coordinated Visual Triangulation

In Section 1.4.3 several systems for visualising and comparing phylogenetic trees are discussed. However, to the best of this author's knowledge, there is no software or method available that deals with the problem of visualising complex phylogenetic trees: that is, trees in which leaf nodes represent complex structures such as pathways.

The prototype system presented is a strong example of both *coordinated and multiple-view displays* and *Triangulation*. The process of designing interactive systems based on coordinated and multiple-view displays is fast becoming a research area unto itself [157]. Basically the concept applies to interactive systems featuring several linked views of a particular data-set. *Triangulation* denotes the use of a combination of methodologies in the study of a particular phenomenon [106]. Several views are proposed, each capturing different aspects of the complex phylogenetic tree structure. They are brought together in such a way that interaction with one view, such as a change of focus, effects a change of detail in the other views. This coordination allows the user to explore the intimate relationship between the tree structure and its pathway nodes visually.

The four views: *Phyogenetic tree*, *Related pathways*, *Operational pathway* and *Hypothetical Pathway* are described in more detail below. The coordination of these four views is demonstrated by the use-case diagram shown in Figure 6.18.

Phylogenetic Tree View

A conventional phenogram, or 2D drawing of the phylogenetic tree, serves here as the highest level overview of the complex phylogenetic tree structure (see the central box in Figure 6.18). As is discussed in Section 1.4.3, usually phylogenetic trees are drawn with no specific leaf ordering. However, in this application, where the focus is on comparing metabolic pathways, it makes sense to find a leaf ordering that minimises differences between the underlying pathways of adjacent leaves. Note the similarity to MIN SUM GRAPH STACKING (Problem 6.3.1), however, in this new ordering problem the neighbourhood constraints of the tree must be respected such that the tree drawing remains planar. Such orderings are known as *optimal leaf orderings* and, thanks to the neighbourhood constraints, can be computed efficiently for any similarity measure [8].

The tree diagram is interactive in that an internal node can be selected to focus on the subtree consisting of all its descendants. This selection causes the subtree to be highlighted and affects other visualisations as described below. Furthermore, a leaf node in the currently selected subtree

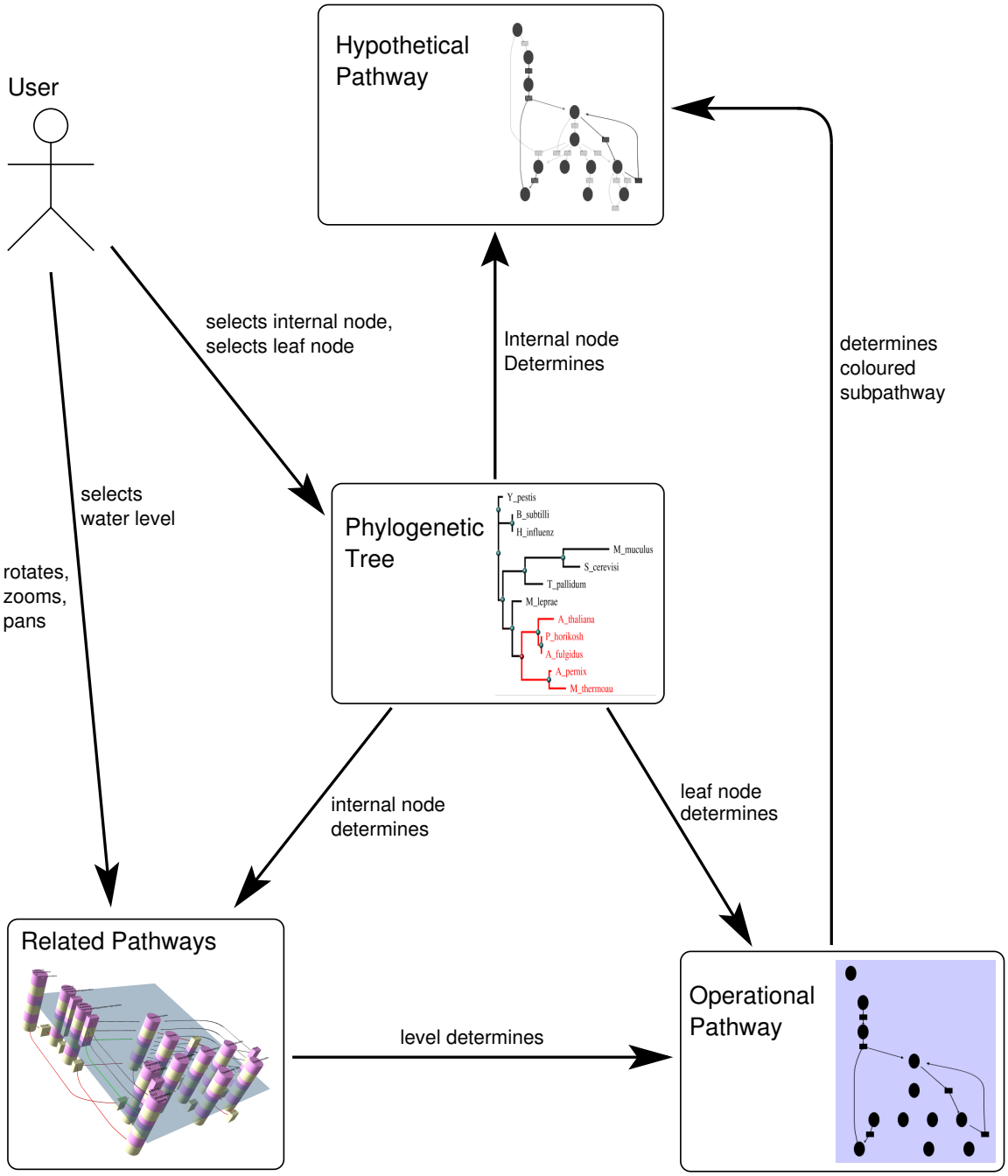


Figure 6.18: Use-case diagram for triangulating a complex phylogenetic tree by four coordinated visualisations.

may be checked to select the operational pathway.

Hypothetical Pathway View

As described in Section 6.4.1, an internal node in a phylogenetic tree corresponds to a hypothetical ancestor of the leaves descendant from that node. The hypothetical pathway view shows a possible metabolic pathway for the currently selected internal node in the phylogenetic tree view (see the top box in Figure 6.18). This pathway graph is inferred by taking the union graph of the pathway graphs for each descendent leaf node. In the pathway graphs corresponding to these leaf nodes, some elements may appear more frequently than others. Therefore, in the visualisation for this hypothetical pathway, each element is displayed with a shading intensity that is proportional to its frequency of occurrence. In the examples shown, an element that appears in all the pathways associated with the selected leaves appears solid black, while an element that appears in just a single leaf is shown as light grey. In the prototype system demonstrated here, the 2D pathway views (hypothetical pathway and operational pathway) are very simple. In a practical system more elaborate visualisations such as those produced by the BIOPATH system would be used for these 2D views, see Figure 6.1.

Related Pathways View

The $2\frac{1}{2}$D metabolic pathway comparison method described in Section 6.3, is used for visualising the set of pathways corresponding to the descendent leaves of the internal node selected in the phylogenetic tree view (see the lower left box in Figure 6.18).

Operational pathway View

An individual leaf can be singled out for detailed inspection (see the lower right box in Figure 6.18) by selecting it, either:

- Directly from the phylogenetic tree.
- From a list of the leaves of the selected subtree.
- From the $2\frac{1}{2}$D visualisation of these leaves.

For the last option, the user can move a semi-transparent “water-level” plane up and down through the $2\frac{1}{2}$D structure and the cross-section is shown in a 2D window. The water-level plane

used in the selection process for the example is shown in Figure 6.22.

While additional details on the selected operational pathway can be shown, it is important to maintain consistency across visualisations. Therefore, the positions of elements in the operational pathway view and also the hypothetical pathway view must correspond to positions of elements in the $2\frac{1}{2}$D view.

6.4.3 Application Example

To illustrate the use of visual triangulation in the analysis of complex phylogenetic trees the thiamine pathways of twelve different species are studied.

Using the similarity matrix in Table 6.2, the phylogenetic tree was constructed using the *neighbour-joining* algorithm of Saitou and Nei [166]: that is, a hierarchical clustering method implemented in the phylogenetic analysis package *phylip* [61]. This tree is shown in Figure 6.19.

As noted in Section 6.1.3, metabolic pathway databases are known to contain inconsistencies and errors. Furthermore, the species-specific pathway data in KEGG is derived from genome data but not experimentally proven. This lack of complete and reliable data has a direct negative influence on the quality of the phylogenetic tree.

Visualisations

A prototype system supporting coordinated visual triangulation, as described in Section 6.4.2, was implemented based on the WILMASCOPE 3D graph visualization system (see Appendix B).

Figure 6.19 shows the phylogenetic tree with edge lengths set according to the computed values. The hypothetical pathway view, consisting of all elements present in any of the leaf pathways, is shown on the right-hand side of the same figure. Any node of the phylogenetic tree may be selected by the user, thus changing the information shown in the hypothetical pathway view. In the figure no selection is made so the union graph of all elements present in any of the pathways is shown. Note that the lightest boxes in the hypothetical pathway view correspond to reactions which are found in the thiamine reference pathway of the database but not in any of the species-specific pathways. These have been included as a reference for the biologist.

Figure 6.20 shows our related pathway view, displaying the set of pathways corresponding to the leaves of the selected subtree from the phylogenetic tree view (Figure 6.19). Note that the stacking order matches the order of leaves shown in the tree view. Note also that the bottom layer shows the full set of reactions from the thiamine reference pathway, as described above.

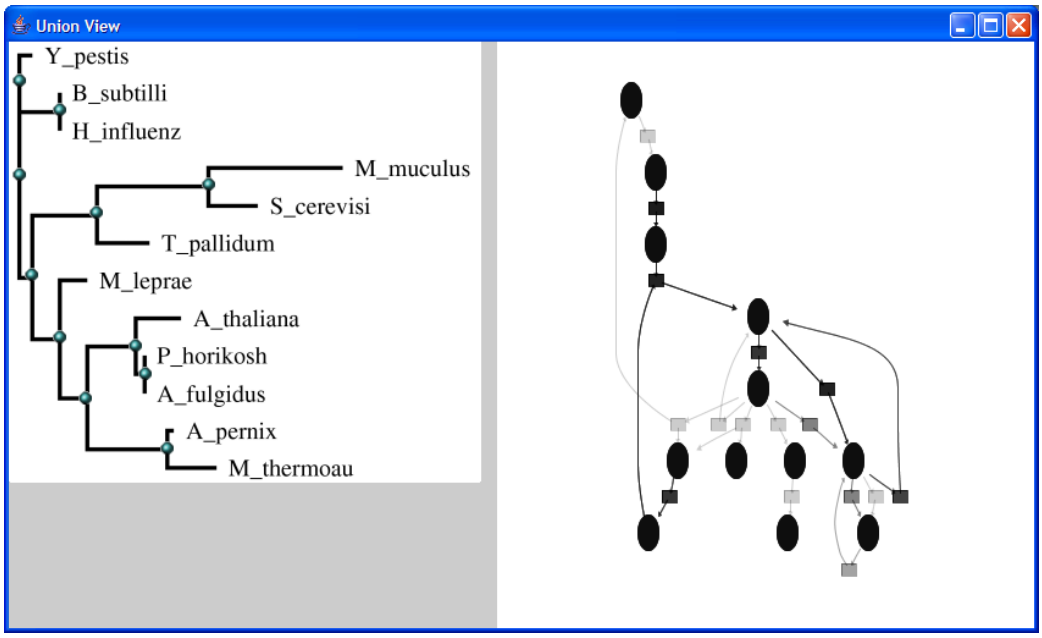


Figure 6.19: Phylogenetic tree view and Hypothetical pathway view showing the phylogenetic tree built from the thiamine pathway of twelve species. Since no internal node of the tree has been selected by the user the hypothetical pathway corresponds to the root of the tree.

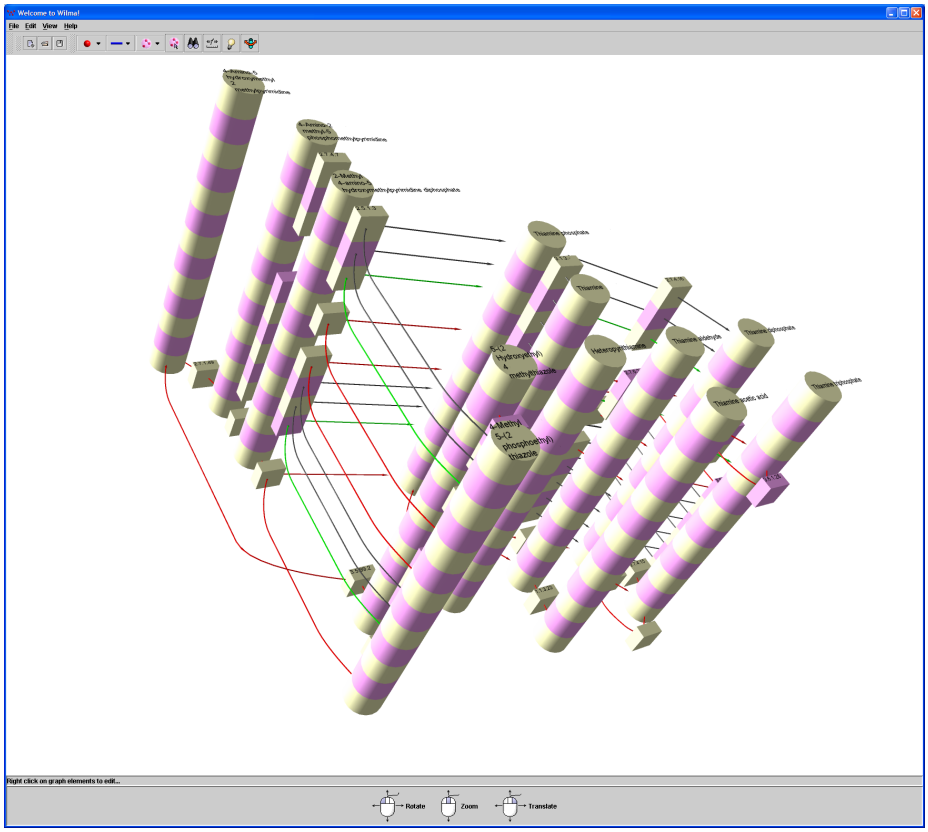


Figure 6.20: Related pathway view showing pathways from Figure 6.19 stacked in leaf order with the thiamine reference pathway located at the bottom of the stacking

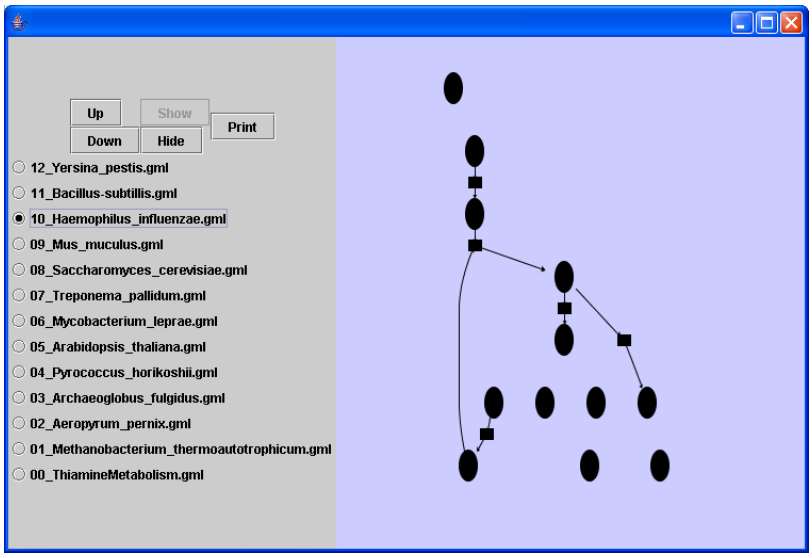


Figure 6.21: An operational pathway can be selected either directly, using the radio buttons shown, or from the $2\frac{1}{2}$D view for more careful analysis.

The process of browsing operational pathways (individual leaf pathways) is shown in Figure 6.21 for the pathway of *Haemophilus influenzae*. Figure 6.22 shows how a pathway (*Archaeoglobus fulgidus*) was selected as a cross-section of the correspondence view by moving the transparent blue "water-level" plane.

In Figure 6.22 all windows in the system showing all the coordinated views are presented in a single screenshot. In the tree view window, on the bottom-left corner of the figure, an internal node of the phylogenetic tree has been selected, thereby highlighting the left and right subtrees beneath it. The hypothetical pathway corresponding to this internal node is shown in the right panel of the tree view window. Note that many of the elements of the pathway are a lighter grey indicating that they occur in fewer of the pathways represented by the leaves. Selecting a subtree in this view also affects the other views. The larger window, in the background of the figure, shows the related pathways view for the selected subtree and the list of navigable leaves is also reduced in the operational pathway browser on the bottom right. Possibilities for navigation through the complex phylogenetic tree and operations on the various views (e.g. zooming) are also shown in Figure 6.18.

6.5 Discussion

Two of the methods described in this Chapter were designed based on requirements gathered from consultations with biologists and bioinformaticists from the IPK Crop Plant Research Centre in

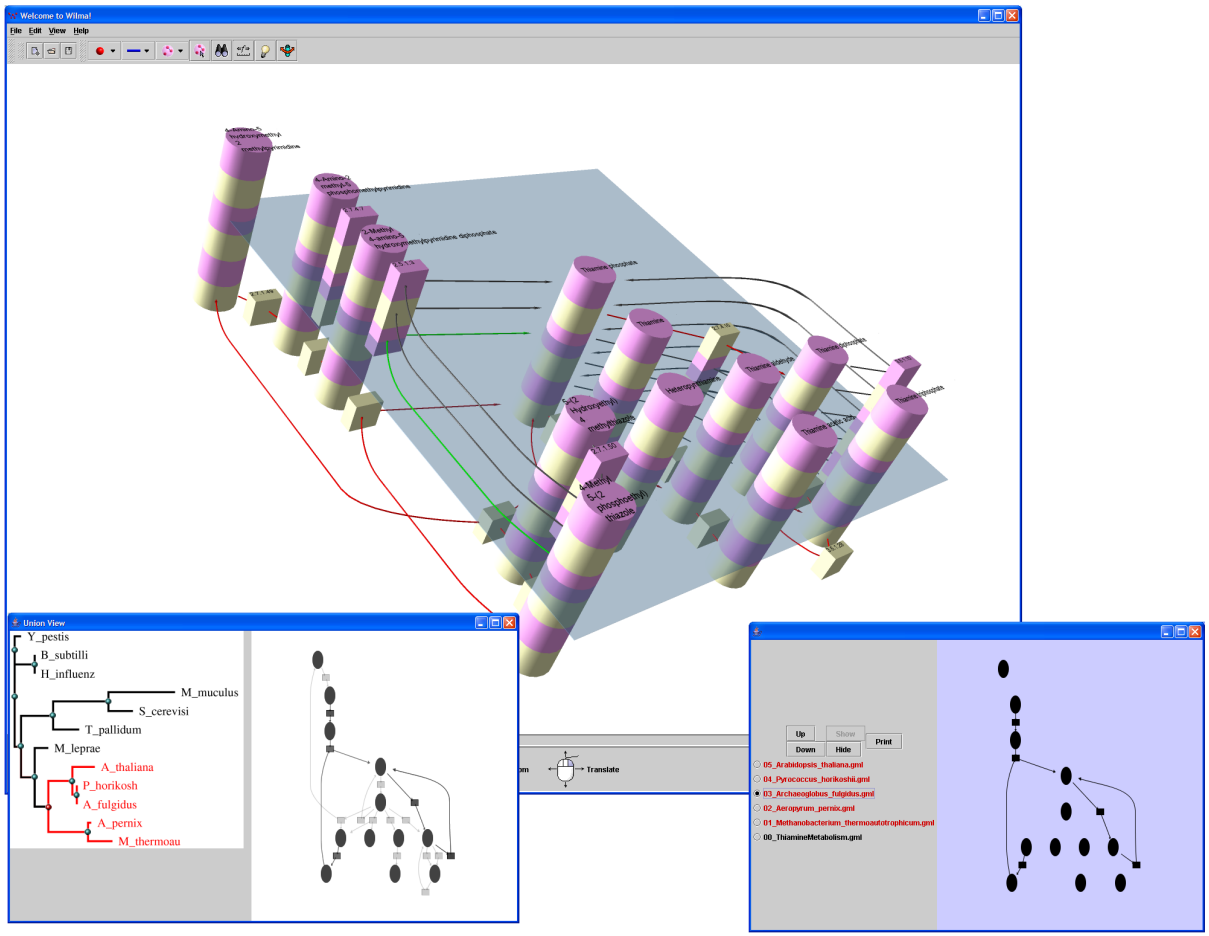


Figure 6.22: A screenshot showing all the coordinated views. Again the phylogenetic tree window from Figure 6.19 is shown. However, here an internal node has been selected by the user highlighting a subtree. Only elements which exist in leaves of this subtree and the reference pathway are shown in the hypothetical pathway on the right side of this window. The selection also applies to the leaf pathways shown in the $2\frac{1}{2}$D stack (background window) and the individual pathways available for browsing in the slice view (lower right window)

Gatersleben, Germany. These are the visualisation method described in Section 6.2 — for mapping experimental data onto nodes and edges of a metabolic pathway graph — and the method described in Section 6.3 — for comparison of the structure of a set of metabolic pathway graphs. Several additional and impartial biologists, also from the IPK Crop Plant Research Centre, were interviewed to evaluate the utility of our application of $2\frac{1}{2}$D stratified network visualisation techniques to these types of metabolic pathway analysis. Using a strategy similar to the evaluation of the portfolio data visualisation system in Chapter 5, a cognitive-walkthrough style evaluation was followed, as introduced in Section 1.5. Three pairs of biologists were interviewed separately and the visualisation systems described here were demonstrated in "guided walkthroughs". The prototypes for these applications, however, did not feature as much interaction as the portfolio visualisation system. As such, after a relatively brief "walkthrough" the interviews became brain-storming sessions. Audio-tape recordings of these interviews are available and an example transcription is included in Appendix A.

The IPK biologists, inspired by the visualisations they had seen, suggested ideas for a number of refinements. Some of these refinements have already been implemented, most notably:

- The inter-stratum spacing to indicate distance in the underlying distance matrix for pathway comparison (see Section 6.3.4).
- A method for browsing larger sets of related pathways with a coordinated view of the phylogenetic tree.

The latter suggestion became the system for coordinated visual triangulation discussed in Section 6.4.2. The following discussion includes additional points raised in these discussions as well as other ideas for further work, which it is hoped, may eventually lead to a fully functional system supporting biologists in metabolic pathway analysis.

In general, the feedback regarding the use of the $2\frac{1}{2}$D stacked view for comparing related pathways was very positive. Most interview subjects, however, felt that the number of pathways which could be successfully visualised simultaneously in this way was limited. As a general "rule-of-thumb" the limit seems to be in the vicinity of six to ten pathways in a single visualisation before the complexity of the scene becomes unviable. A method for interactively selecting which of the available pathways are included in the stack is therefore essential. Thus, the phylogenetic tree view not only provides this facility but does so in a way that is meaningful and intuitive for the biologists.

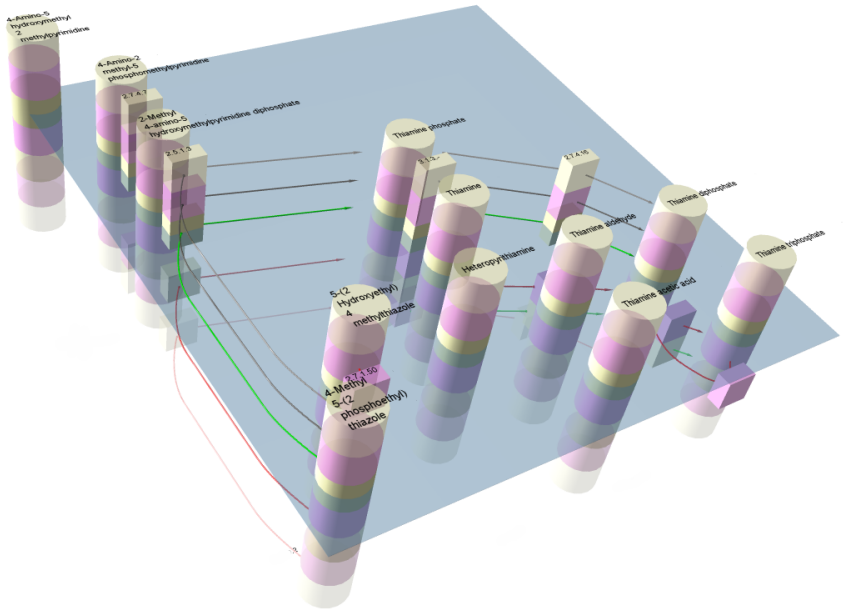


Figure 6.23: A demonstration of a possible way of limiting the number of pathways shown when a stack of related pathways is too large and visually complex. Seven pathways are shown, a focal pathway and three either side becoming progressively more transparent the further the pathway is from the focus.

A further logical extension supporting this mode of interaction is to allow for the selection of arbitrary multiple internal nodes and leaves from the phylogenetic tree for inclusion in the stack. It is also planned to further develop the components visualising operational and hypothetical pathways to produce graphics comparable to those in Figures 6.1 and 6.12, to eliminate the need for cross-implementation data exchange.

Another way to limit the number of pathways shown in a single scene is to show only a *window* of a small number (for example, seven) pathway strata from a larger stack. The user is able to move the focal stratum (marked by the water-level) up or down through a stack of potentially unlimited height. On moving up an additional pathway is added to the top of the visible stack while another is removed from the bottom; the reverse occurs when the user moves the focal stratum down. Figure 6.23 demonstrates this idea, where the transparency of graph elements is controlled such that the strata appear to fade to invisibility on either side of the focal stratum. This transparency effect is added to convey a sense of continuity: that is, a sense of the scene being only a sample of a larger stack. Visual complexity is thus limited within a single view. In the example the global layout is still used for the entire stack. That is to say, the layout does not change as the visible set of graphs changes. The layout could, however, be recalculated each time the user changes the focus such that layout is always optimal for the visible set of graphs, rather than the full set.

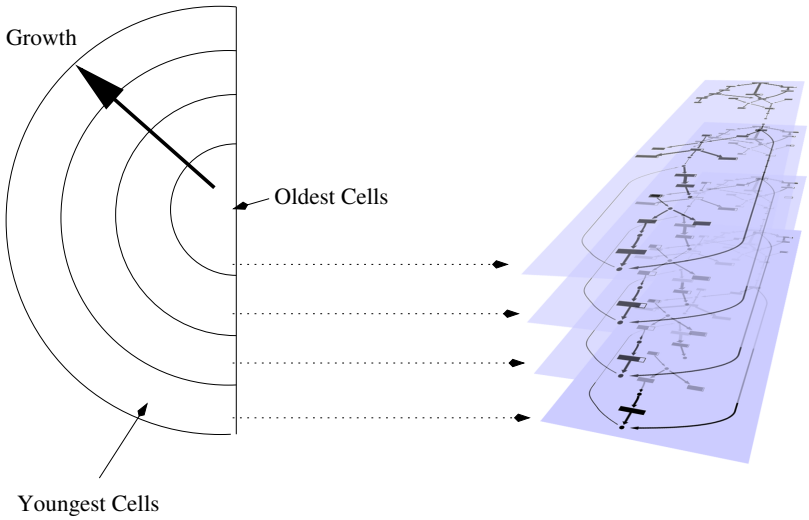


Figure 6.24: A physical mapping for the metabolic pathways from different stages in the development of a seed to a stratified graph visualisation.

The IPK biologists were also interested in the possibility of a combination of both of our applications: that is, in creating a method for comparing experimental data from different time samples across a number of different species and/or a wild-type (an arbitrarily chosen benchmark strain). This leads to an extra degree of complexity and an explosion in the total number of pathway graphs that need to be visualised. One way to handle the situation in an interactive stratified graph visualisation system is to coordinate two stratified graph visualisation windows. In addition to the $2\frac{1}{2}$D related pathway view, showing the stack of pathways from different species, a second $2\frac{1}{2}$D view of experimental data associated with the operational pathway could be shown. As in the coordinated visual triangulation system, the operational pathway would be selected as a cross section from the related pathway view.

Another idea raised by the IPK biologists is related to a method for visualising a plant seed's metabolic development. Biologists study the various phases of a seed's germination by harvesting sample seeds at key stages in their growth and examining tissue samples from the outer (youngest) tissue layer. Doing this they are able to construct a model of development with data relating to metabolic pathway structures at each stage. The IPK biologists were intrigued at the prospect of visually presenting these pathways as a stack ordered from earliest to latest stages in the seed's development, where the strata would correspond directly to layers in its physical structure.

Finally, an issue that became apparent in conducting the cognitive-walkthroughs with the IPK biologists was the clumsiness of collaborating in a 3D-environment with poor display technol-

ogy. The interviews with fund-manager performance analysts (as discussed in Chapter 5) were conducted in a visualisation laboratory equipped with a large, rear-projected, stereo 3D display environment. By contrast, the interviews with the IPK biologists were conducted on a laptop, on-site at the IPK Crop Plant Research Centre facilities. Although no controlled side-by-side comparison was performed, from this anecdotal evidence it seems that interview subjects are more easily confused when forced to navigate around a 3D environment on a small screen. This conclusion is supported by Ware's findings [195] on the effect of emersiveness of the display environment on users' understanding of a 3D graph visualisation. The lesson here is that even carefully designed 3D visualisation applications may not gain widespread popularity until the hardware infrastructure is readily available.

CHAPTER 7

Phylogenetic Trees

> "With the failure of these many efforts [to explain the origin of life] science was left in the somewhat embarrassing position of having to postulate theories of living origins which it could not demonstrate. After having chided the theologian for his reliance on myth and miracle, science found itself in the unenviable position of having to create a mythology of its own: namely, the assumption that what, after long effort, could not be proved to take place today had, in truth, taken place in the primeval past." — L. C. Eiseley, *The Immense Journey* [58, Page 199]

This chapter presents the final case study for the two-and-a-half-dimensional graph visualisation metaphor. A new aesthetic criterion is introduced for the $2\frac{1}{2}$D stratified visualisation of sets of related phylogenetic trees. This aesthetic requires that the number of crossings between edges linking similar nodes in adjacent trees is minimised, thus allowing a user to easily track the position of a particular species through its lifetime in the stack. Also introduced, is an algorithm for efficiently minimising such crossings together with a discussion of its complexity. Finally, a new "barrel" layout style for stratified tree visualisation is introduced, which benefits from such crossing minimisation. Much of the work presented in this chapter has been published in [51].

This chapter is organised as follows. First, Section 7.1 introduces the area of phylogenetic tree comparison. In Section 7.2 *stratified tree visualisation* is defined as a type of $2\frac{1}{2}$D graph visualisation for examining sets of trees. An optimisation problem involved in arranging these trees in a manner supporting visual comparison is introduced. Section 7.3 presents an algorithm for solving this problem and Section 7.4 discusses the complexity of the algorithm. Section 7.5 demonstrates an alternative layout style for stratified tree visualisations. Finally, in Section 7.6 an implementation of the algorithm is discussed together with feedback from biologists and suggestions for further refinements.

7.1 Introduction

A general introduction to phylogenetic analysis and phylogenetic tree visualisation is provided in Section 1.4.3. A specific problem that biologists frequently face in phylogenetic tree analysis, is the comparison of a set of related phylogenetic trees. Such a situation arises when:

- Common methods and computer programs for reconstructing phylogenetic trees often produce a set of related trees, not a single optimal one. For example, such programs may apply maximum parsimony and maximum likelihood methods to a data-set (such as the protein sequences of a specific enzyme in different species) to produce a set of possible trees ranked by a probability score. Furthermore, the tree with the best score (the "optimal" tree) does not necessarily show the correct ancestral relationships among a set of species.
- To improve the quality of the predictions, a set of different trees based on different morphological and molecular data sets is often built. One example is the set of trees built from sequences of different enzymes where each tree is computed using the protein sequences of a specific enzyme in different species.

Therefore, to analyse the evolutionary relationships between species, biologists often wish to compare a set of related phylogenetic trees in order to find the most likely one; the most likely tree being the one that shows most accurately the ancestry relationships considering all given trees. To help users perform this analysis, a $2\frac{1}{2}$D visualisation method for such a set of trees, making similarities and differences between the trees easily recognisable, is considered.

7.2 Background

7.2.1 Visualising Sets of Phylogenetic Trees

As mentioned above, comparing sets of phylogenetic trees is an important part of many biologists' work. As such, a number of tools for such tree comparisons have been suggested. Few, however, allow for the detailed inspection of more than two trees at a time. Recent tools allow for the browsing of trees as nodes in a "similarity graph" [117]. Detailed analysis of a smaller subset of the trees is restricted to a view of a *consensus tree* (see Section 7.6.1). Another recent tool [142] allows for comparison of two graphs side by side. The visualisation method described in this paper

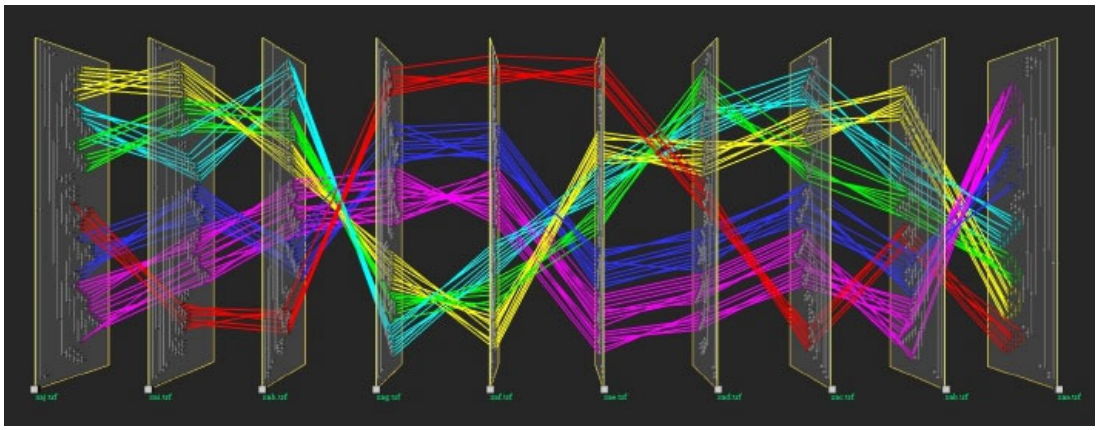


Figure 7.1: The $2\frac{1}{2}$D visualisation of a set of phylogenetic trees proposed by Stewart et al. [179]. Note the large number of crossings between the coloured edges joining similar species in adjacent trees. Visualisation produced by TREEVIEWER. Used by permission.

is differentiated by the ability to simultaneously display the entire structure of a number of similar trees.

7.2.2 Stratified Tree Visualisations

Phylogenetic trees are hierarchies representing the evolutionary relationships between species. The structure of such trees defines only ancestor–descendant relationships. No ordering of the leaves of the trees is specified. Stewart et al. [179] introduce a $2\frac{1}{2}$D method for visualising the output of their phylogenetic inference program. As their program approaches an optimal solution it regularly outputs approximate trees which gradually converge to a best estimate. They found it useful to visualise these trees stacked in the order that they were produced by the algorithm. Their visualisation also allows the user to track the position of groups of leaves across multiple trees by joining the similar leaves with edges. However, doing so with the default leaf arrangements often leads to a large number of crossings between these edges when viewed from above (that is, perpendicular to the leaf edges); see Figure 7.1[1]. Their visualisation tool allow the user to manually swap nodes in the tree in order to better arrange these leaf nodes, but doing so is arduous. In coming sections an algorithm which performs this task automatically is explored.

[1] Visualisation produced by the TREEVIEWER system: http://www.avl.iu.edu/projects/Tree3D/

7.2.3 Optimal Leaf Ordering

The readability of these diagrams is improved by arranging the leaves such that the number of crossings between edges connecting matching leaves in adjacent trees is minimised. Figures 7.2 and 7.3 show $2\frac{1}{2}$D visualisations of a set of phylogenetic trees with the default ordering and then with crossings removed.

First, some useful terms are defined. A phylogenetic tree $T = (S, L)$ has a set S of nodes (species) and a set L of edges or links, where each edge $(u, v) \in L$ indicates that u is an ancestor of v. The set of leaves $V \subseteq S$ are nodes in T with only one link. A linear ordering for the leaves is given by $\pi = (v_1, v_2, \ldots, v_n)$ where $v_i \in V$. A given π is said to be a *linear leaf ordering consistent with the tree constraints of* T, if T can be drawn such that:

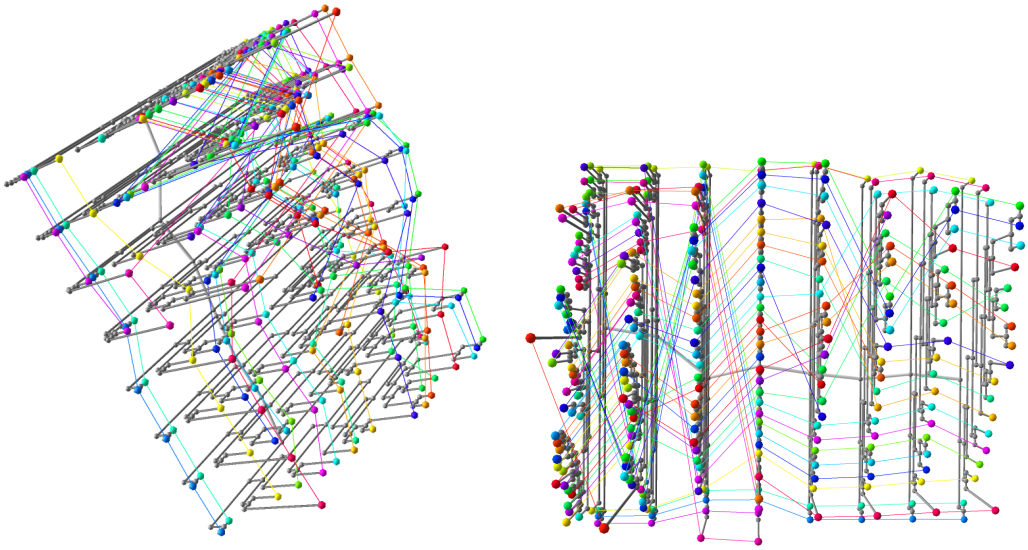
- All leaves $v \in V$ are arranged along a straight line with order π.
- The internal nodes $S \setminus V$ are arranged to one side of the line.
- All edges are drawn monotonically on the same side as the internal nodes with no crossings.

Intuitively, such an ordering for the leaves of T requires that the children of any internal node of T must be ordered consecutively.

Two or more phylogenetic trees may be "related" in the sense that they have leaves in common. In an *ordered set of related trees* $S_T = (T_1, T_2, \ldots, T_r)$, the sets of leaves of each pair of adjacent trees, V_i and V_{i+1}, have a non-empty intersection, i.e. $V_i \cap V_{i+1} \neq \emptyset$.

A *stratified tree visualisation* is a visualisation of an ordered set of related trees S_T. Each tree T_i is drawn in the $x \times y$ plane with leaves arranged in a linear ordering consistent with the tree constraints of T. The set of r planes is stacked into the z-dimension according to the order defined by S_T. Additional edges A are drawn between matching leaves in each pair of adjacent trees (T_i, T_{i+1}).

Inter-leaf edge crossings occur when edges in A appear to intersect as the *stratified tree visualisation* is viewed from the leaf side, as in Figure 7.1. More precisely, the number of crossings $cross(A_{i,i+1})$ between a pair of trees in adjacent strata (T_i and T_{i+1}) is the number of inversions between the ordering π_1 of leaves on one stratum and the ordering π_2 of their matching pairs on the adjacent stratum. Let $\pi_1(v)$ be an integer giving the position of leaf v in the leaf ordering π_1 of T_1. Let V be a list with entries $V[i]$ of the leaves in π_2 order. Let λ be a list with entries $\lambda[i] = \pi_1(V[i])$.



(a) Oblique view (b) Top view

Figure 7.2: Two views of a $2\frac{1}{2}$D visualisation of a set of eight phylogenetic trees with unmodified leaf orderings. Note the large number of crossings (799) between edges linking leaves on adjacent planes.

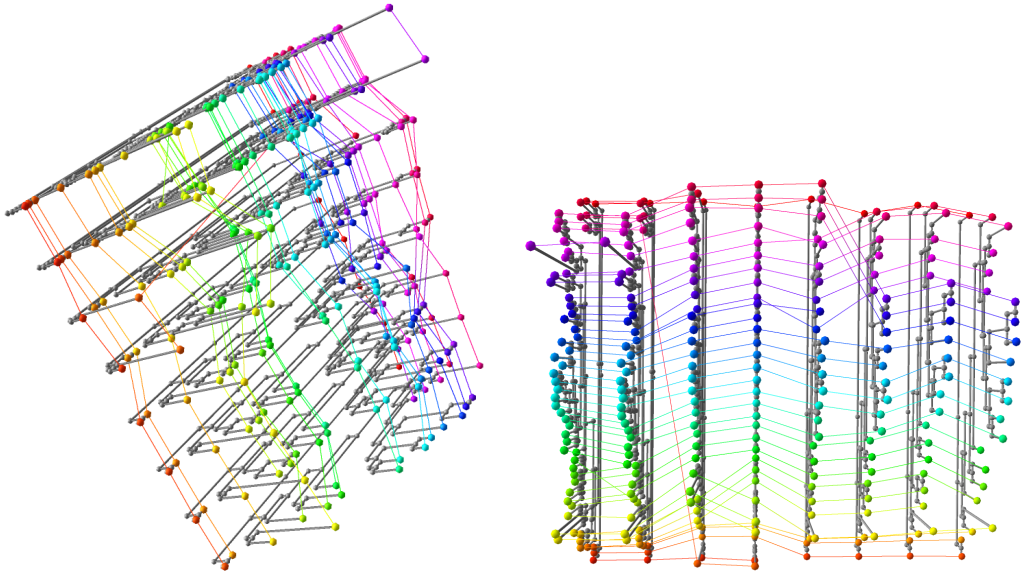
Then $cross(A_{i,i+1})$ is the number of inversions in λ: that is, the number of pairs $(\lambda[k], \lambda[l])$ with $k < l$ and $\lambda_k > \lambda_l$. The total number of inter-leaf edge crossings in a stratified tree visualisation with a set S_T of r trees is $cross(S_T) = \sum_{i=1}^{r-1} cross(A_{i,i+1})$.

The occurrence of inter-leaf edge crossings in a stratified tree visualisation and the notion that a visualisation with fewer such crossings is more readable than one with many crossings, leads to the following optimisation problem:

Problem 7.2.1 (Stratified Leaf Ordering Problem - SLOP) *Find a stratified tree visualisation with minimum inter-leaf edge crossing.*

INSTANCE: *An ordered set of related trees $S_T = (T_1, T_2, \ldots, T_r)$ and a set A of edges between similar leaves in adjacent trees.*

QUESTION: *Find a set $\Pi = (\pi_1, \pi_2, \ldots, \pi_r)$ of linear leaf orderings for each T_i in S_T such that $cross(S_T)$ is minimum.*



(a) Oblique view (b) Top view

Figure 7.3: Another pair of views of the $2\frac{1}{2}$D visualisation from Figure 7.2 but with crossings minimised using Algorithm 2. Note that crossings have been greatly reduced, only 37 crossings remain.

7.3 SLOP Solver Algorithm

In Section 4.3.2 the problem of reducing edge crossings in a hierarchical graph drawing is examined. There are many parallels between the hierarchical graph drawing crossing minimisation problem and SLOP. Recall that the first problem uses a layer-by-layer sweep approach to reduce crossings one layer at a time. A similar strategy is employed here for solving a SLOP by sweeping through the stack of trees, reducing the number of inter-leaf edge crossings between strata by rearranging one tree at a time.

Analogous to the one-sided crossing minimisation problem in hierarchical graph drawing, a *one-sided SLOP* is defined to be a SLOP with only two strata in which the permutation of one tree is fixed. A major difference, however, is that while the one-sided crossing minimisation problem is $\mathcal{NP}$-complete [57], an optimal solution for a one-sided SLOP can be found in polynomial time using a divide and conquer strategy. One caveat, explained below, is that the unfixed tree must be binary.

Crossings can then be reduced globally in a SLOP spanning r strata with a sweeping approach. Each pair of trees in the stack is examined in turn, and that pair's local one-sided SLOP solved. In only a few sweeps the set Π of leaf orderings should converge to a globally optimal, or near optimal,

solution.

The one-sided SLOP is somewhat similar to another leaf ordering problem, *dendrogram seriation* [140], that is often considered in relation to phylogenetic trees. Recall that phylogenetic trees are usually generated from a hierarchical clustering of an underlying multidimensional data-set. Dendrogram seriation is the process of arranging the tree such that the order of the leaves minimises the distance between neighbouring leaves in the underlying data space. Solving this problem is akin to finding a solution to the *travelling salesman problem* subject to the tree constraints. Morris et al. [140] proposed a simulated annealing approach to find an approximate solution. Bar-Joseph et al. [8] gave a dynamic programming method for finding the optimal solution which runs in $O(n^4)$ time using $O(n^2)$ space.

Before explaining the algorithm, we show that the problem can be divided into a set of independent subproblems. The remainder of this section considers a one-sided SLOP between two trees $T_1 = (S_1, L_1)$ and $T_2 = (S_2, L_2)$, as a directed acyclic graph $G = (V, E)$ encompassing the unfixed tree T_1 and the leaves of the fixed tree T_2. That is, $V = S_1 \cup V_2$ and $E = L_1 \cup A_{12}$ where A_{12} is the set of edges connecting the matching leaves of T_1 and T_2. It is the crossings in A_{12} that need to be minimised. A set of virtual edges is defined between an internal node $v \in V$ and the neighbours of its descendent leaves as $F_v = \{(v, u_2) | (u_1, u_2) \in A_{12} \wedge \exists \, \text{directed path}(v, u_2)\}$.

A composite virtual node u is defined for all the internal nodes to the left of v in an in-order traversal of the tree and a similar composite virtual node w for all nodes to the right of v. For each node replaced by u, let there be a virtual edge from u to each descendent leaf of u, and call the set of these virtual edges F_u. The set F_w of edges between w and descendent leaves of w is defined in the same way. If all the leaves are then visited in π_2 order and their neighbours $\in u, v, w$ from the edges in $F_u \cup F_v \cup F_w$ listed, then the number c of crossings between edges in F_v and the remaining edges $A_{12} \setminus F_v$ is the number of inversions (violating the ordering $u < v < w$) in this list.

Lemma 7.3.1 *In a particular permutation of the tree above v, if there are c crossings between edges in F_v and the remaining edges $A_{12} \setminus F_v$ then $cross(A_{12}) \geq c$ for all permutations of the subtree of v.*

Proof If all edges in the directed paths from v to the leaves of T_2 are removed and replaced with the virtual edge set F_v then all the crossings between edges of leaves that are descendants of v are also removed. The only crossings remaining are those between F_v and the remaining edges in E. In a planar drawing of T the leaves below v must remain together: that is, a leaf below a sibling of

v cannot be placed between the leaves below v. Therefore, any further permutation of the leaves below v cannot further reduce crossings, and the statement above holds.

Thus, we have a notion of *total crossings* and *local crossings* for a particular internal node v in T_1. The children of v are labelled: $u_1, u_2, \ldots, u_\delta$ where $\delta = degree(v)$ and are ordered by π_1 as they appear in the stratified tree visualisation. Let there be a set F_{u_i} of virtual edges for each child of v connecting u_i with its descendent leaves. Local crossings of the children of v ($cross_{local}(u_1, u_2, \ldots, u_\delta)$) are then the crossings between virtual edges of all u_i. That is, if all the leaves are visited in π_2 order and their neighbours $\in u_1, u_2, \ldots, u_\delta$ from the edges in $\bigcup_{i=1}^{\delta} F_{u_i}$ are listed, then $cross_{local}(u_1, u_2, \ldots, u_\delta)$ is the number of inversions in this list.

Total crossings of v ($cross_{total}(v)$) is the sum of the crossings beneath each F_{u_i} and the local crossings of v. Thus, a recursive definition for crossings at each internal node v is:

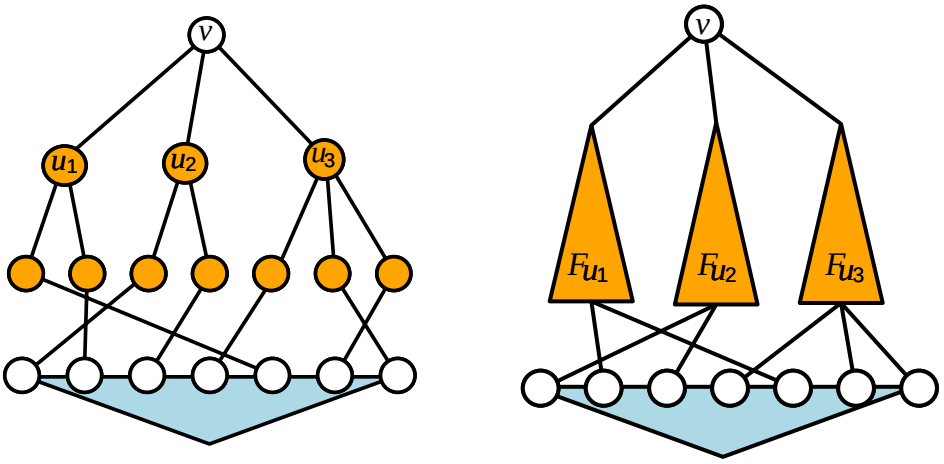
$$\begin{aligned} cross_{total}(v) = & cross_{total}(u_1) + cross_{total}(u_2) \\ & + \ldots + cross_{total}(u_\delta) \\ & + cross_{local}(u_1, u_2, \ldots, u_\delta) \end{aligned} \tag{7.1}$$

Theorem 7.3.2 *A one-sided SLOP can be recursively decomposed into a number of independent subproblems.*

It follows from Lemma 7.3.1 that an ordering of internal nodes $u_1, u_2, \ldots, u_\delta$ that causes a minimal number of crossings $cross^*_{local}(u_1, u_2, \ldots, u_\delta)$ remains optimal regardless of how the subtrees of $u_1, u_2, \ldots, u_\delta$ are arranged. Thus, an effective divide and conquer strategy is to process the tree bottom up, eliding subtrees as described in Lemma 7.3.1.

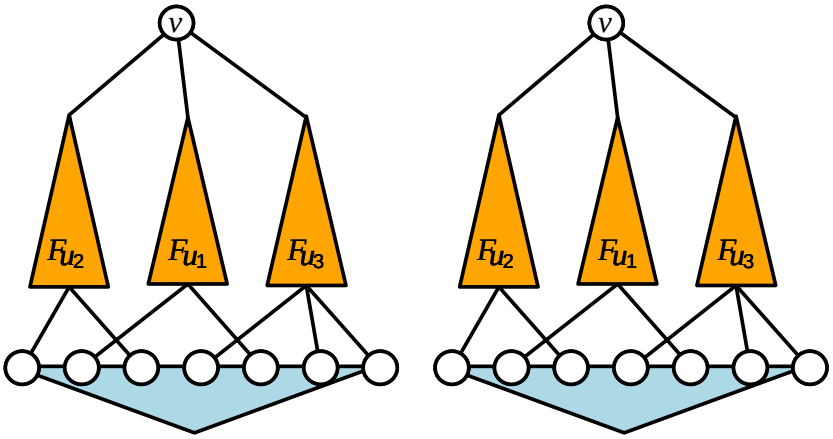
A visual demonstration of the decomposable nature of the SLOP is given in Figure 7.4. Figure 7.4(a) shows the subtree below an internal node v of a larger tree T_1. The leaves descendent from v are shown in order π_1, connected by edges to matching leaves of an adjacent tree T_2, with the permutation π_2 of the lower tree fixed. Note that, for the three children of v: $cross_{total}(u_1) = 1$, $cross_{total}(u_2) = 0$ and $cross_{total}(u_3) = 1$.

Collapsing the subtrees below each of these children into the virtual nodes F_{u_1}, F_{u_2} and F_{u_3}, as in Figure 7.4(b), elides the contribution to the total number of crossings from each of these subtrees and leaves only $cross_{local}(u_1, u_2, u_3) = 4$ crossings. In Figure 7.4(c) the children of v have been reordered leaving $cross^*_{local}(u_1, u_2, u_3) = 2$ crossings. In Figure 7.4(c) the reordered tree is shown



(a) A tree with leaves linked to another tree

(b) After collapsing the subtrees below the nodes on level 2



(c) Rearranged to minimise crossings

(d) Revealing the subtrees again

Figure 7.4: Demonstration that one-sided crossing minimisation problem subject to tree constraints is decomposable

with the subtrees below u_1, u_2 and u_3 expanded. $cross^*_{local}(u_1, u_2, u_3)$ is still 2 and $cross_{total}$ for each of the subtrees have not changed. Thus, it can be seen that the contribution to the total number of crossings due to an ordering of children below a particular internal node is independent from the crossings contributed by the subtrees of those children. Note also, that an arrangement of children of a particular node which minimises the number of crossings between leaf edges descendent from those children, remains optimal regardless of arrangements of their descendants.

This recursive divide and conquer strategy is realised in the procedure used in Algorithm 2.

Lines 1–3 cause a depth-first traversal of the tree and provide a list of virtual edges connecting v to its descendent leaves (or an actual edge if v is a leaf). The algorithm minimises crossings for each internal node by exchanging children of the node if doing so reduces crossings.

Algorithm 2 Reduce crossings below internal node v of the unfixed tree in a one-sided SLOP

procedure $decross(v)$

Require: $v.children$ is an array indexed from 0 in which each element is also a tree node. If v is a leaf $|v.children| = 0$ and $|v.edges| = 1$ else $|v.children| > 0$ and $|v.edges| = 0$.

```
1:  for u ∈ v.children do
2:      v.edges ← v.edges ∪ decross(u)
3:  end for
4:  for i ∈ {x ∈ ℤ : 0 < x < |T.children|} do
5:      u ← v.children[i − 1]
6:      w ← v.children[i]
        {(u, w) are adjacent children}
7:      c_uw ← countCrossings(u, w)
8:      if deg(u)deg(w) > 2c_uw then
9:          v.children[i − 1] ← w
10:         v.children[i] ← u
11:         return v.edges
12:     end if
13: end for
```

The test in line 8 of Algorithm 2 is testing $c_{wu} > c_{uw}$ and comes from the observation [11, Lemma 9.3(a),Page 290] that:

$$c_{uw} + c_{wu} + \chi_{uw} = deg(u)deg(w) \tag{7.2}$$

where $deg(u)$ is the degree of node u and χ_{uw} denotes the number of common neighbours of u and w. In a SLOP the neighbours of u and w all have degree 1 and therefore $\chi_{uw} = 0$. Therefore, if $c_{wu} > c_{uw}$ we can substitute from Equation 7.2 to obtain:

$$\begin{aligned} deg(u)deg(w) - c_{uw} &> c_{uw} \\ deg(u)deg(w) &> 2c_{uw} \end{aligned} \tag{7.3}$$

Note that for an unfixed tree of depth one, the algorithm functions identically to the well known *adjacent-exchange* or *greedy-switch* algorithms [11] (see Algorithm 1 in Chapter 4). For trees with internal nodes of high degree it is possible that an algorithm based on the *median heuristic* might converge more quickly. However, since most of the input trees considered are binary, or of degree

no greater than three, this is not an issue in this application. In fact, a result of Theorem 7.3.2 is that for binary trees the algorithm reaches an optimal solution for the one-sided problem in a single pass.

An obvious potential bottleneck in the algorithm is the crossing count. A naïve algorithm would check every edge attached to u against every edge of v leading to time $O(|E_u||E_v|)$. Algorithm 3 is a simplification of a two layer crossing counting algorithm by Barth et al. [10]. Their algorithm has running time $O(|E| \log |V_{small}|)$ where E is the set of edges and V_{small} is the smaller of the two sets of nodes. In the binary tree case, $|V_{small}| = 2$ so the existence an $O(|E_u| + |E_v|)$ time algorithm is obvious. Although the linear time radix sort on line 2 does not increase this complexity bound, it can be eliminated to further simplify the algorithm. In the algorithm of Barth et al. it was required when nodes on the fixed level had degree greater than one. In the one-sided SLOP this is not the case, so if nodes on the fixed side are stored in an array in π_2 order, then a simple iteration through the array will provide the list of edges in the required order.

Algorithm 3 Count edge crossings between two nodes

procedure $countCrossings(u, v)$
Require: u and v each have lists of edges $u.edges$ and $v.edges$
1: $E \leftarrow u.edges \cup v.edges$
2: sort E using radix sort {sort order is $(\pi_2(tail), u < v)$ – sort can be eliminated, see text}
3: $crossCount \leftarrow vCount \leftarrow 0$
4: **for** $e \in E$ **do**
5: **if** $e.head = v$ **then**
6: $vCount \leftarrow vCount + 1$
7: **else**
8: $crossCount \leftarrow crossCount + vCount$
9: **end if**
10: **end for**
11: return $crossCount$

7.4 Complexity

Considered first is the complexity for a one-sided, binary tree SLOP only. A single descent of a binary tree with n leaves takes exactly $n - 1$ steps. The most expensive part of Algorithm 2 is the call to the *countCrossings* subroutine. Counting crossings with the algorithm given in Algorithm 3 takes $O(|F_v|)$ time for each internal node v of the tree where F_v is the set of edges connected to leaves descendent from that node and $|F_v| \leq n$.

If the root has two children, with l and $n - l$ descendent leaves respectively, then time $\tau(n)$ satisfies:

$$\tau(n) = \tau(l) + \tau(n - l) + O(n) \tag{7.4}$$

In the worst case $\tau(n)$ is $O(n^2)$, however, if T_1 is balanced then $\tau(n)$ is $O(n \log n)$. Since the clustering algorithms used to infer phylogenetic trees are designed to produce trees that are balanced, typical running times are closer to this lower bound.

As mentioned above, to minimise crossings for a set of r-trees the one-sided SLOP algorithm is applied to each pair of adjacent strata in a stratum-by-stratum sweep strategy. In limited tests, this approach has been found to converge to a reasonable solution in a small number (less than ten) passes. Further work might involve testing under what conditions this approach fails to reach a globally optimal solution. However, in practice, the approach has been found to be at least adequate for test cases of the size shown in the figures. That is, data-sets with less than 10 trees and 50 leaves per tree. Larger cases might not be practical anyway due to the obvious bottleneck in wet-ware: that is, the ability of a human to understand an overly complex visualisation.

7.5 Barrel Tree Layout

On seeing the $2\frac{1}{2}$D visualisations shown in Figure 7.3 it was thought that arranging the trees in a configuration with the leaves spaced evenly around the perimeter of the circle might be a better utilisation of the available space. This is achieved using a weighted barycentre algorithm, as follows:

- The leaves, ordered according to a proper leaf ordering π for the tree, are equally distributed around the perimeter of a circle.
- Internal nodes are traversed bottom-up (from leaves to root) and for each parent node v a position $pos(v)$ is found relative to its set C_v of children (leaves or internal nodes). The position $pos(v)$ is determined from the weighted barycentre of the positions of nodes in C_v, where each child $c \in C_v$ is weighted by branch length c_w. That is, the position in the plane for each internal node v is given by:

$$pos(v) = (v_x, v_y) = \frac{1}{\sum_{c \in C_v} w_c} \sum_{c \in C_v} w_c(c_x, c_y)$$

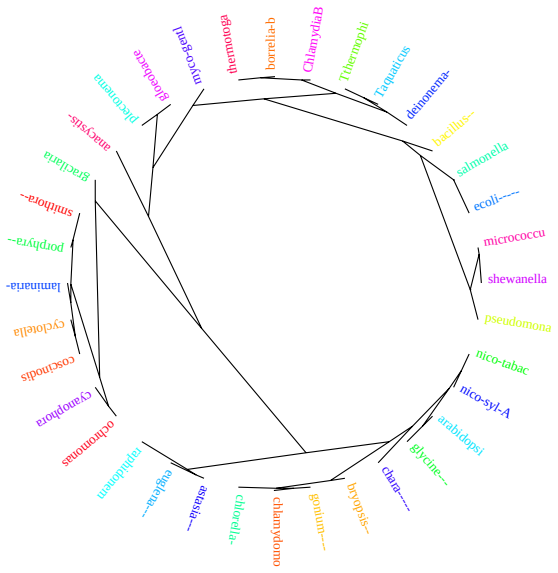
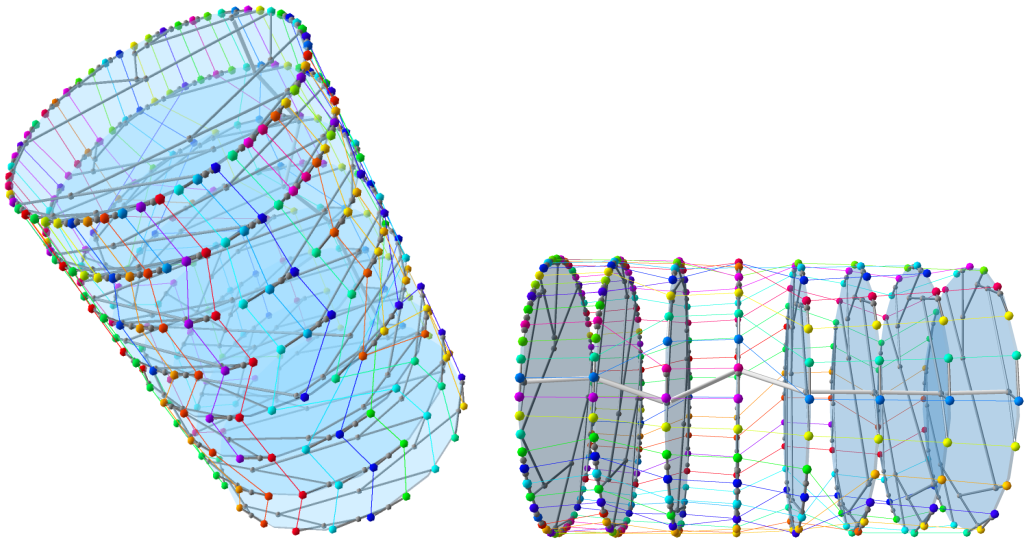


Figure 7.5: An example of a circular drawing of the phylogenetic tree from Figure 6.17.

Note that this procedure guarantees a planar embedding. This is because, with the leaves arranged around the circumference of a circle, they form boundary nodes on a convex polygon. Inscribing the tree inside this polygon produces a tri-connected planar (since the leaves are arranged in π-order) graph. Therefore, the embedding must be planar according to the famous theorem by Tutte [189].

An example of this style of layout for a single tree is shown in Figure 7.5. Figure 7.6 gives the $2\frac{1}{2}$D "barrel" view of the set of trees. Another feature added to the barrel layout is a "spine" connecting the roots in adjacent trees. As well as identifying the position of the root in each layer, this spine may also provide an informal measure of similarity between adjacent trees. That is, trees with leaf ordering determined by the SLOP-solver method and arranged according to the barrel layout method above, should also have similarly placed roots in adjacent trees. Therefore, a stratified stack of trees with similar internal structure should have a fairly straight spine, while a set of trees with disparate structure should have a spine with bends and kinks. Clearly, using such a device is not a precise similarity measure, but it has been met with favourable feedback from biologists who feel it may be useful for providing a "summary" of differences between trees.



(a) Oblique view (b) Top view

Figure 7.6: A visualisation of the same data-set used in earlier figures but arranged in the circular style shown in cross-section in Figure 7.5.

7.6 Discussion and Further Work

An algorithm is introduced for minimising the number of crossings among the edges linking similar leaves in adjacent planes of a $2\frac{1}{2}$D phylogenetic tree visualisation. This chapter also suggests a new layout for such $2\frac{1}{2}$D tree visualisations: the "barrel" view.

A program implementing the SLOP solver algorithm — and also the standard phenogram and barrel stratified tree layout styles — was created in the PYTHON scripting language. It includes a parser for simple FASTDNAML tree description files[2] and generates XML graph files with the stratified tree visualisation positions assigned to nodes by the layout algorithms. These XML graph files can then be viewed with standard 3D navigation facilities in WILMASCOPE (see Appendix B).

Unlike the other case studies in Chapters 5 and 6, where more extensive, interactive, prototype systems with a number of novel user interface features are described, the implementation for this case study simply produces static stratified tree visualisation models for viewing in WILMASCOPE. For this reason, the evaluation for the work in this chapter did not involve the cognitive walk-through style evaluation as described in these other chapters. Rather, the stratified tree visualisations were simply shown to biologists with an interest in phylogenetic analysis and their feedback requested.

[2]Kindly supplied by the team at the University of Indiana who generated the original stratified tree visualisation shown in Figure 7.1.

The discussions were recorded on audio tape and a transcription of a sample discussion is given in Appendix A. Mostly, these discussions revolved around the potential of the stratified tree visualisation paradigm and features that would be useful in a complete interactive system for exploring sets of phylogenetic trees. The following is a report on findings from this feedback as they become relevant to the discussion of further work, below.

The feedback from biologists has for the most part been very positive. They found the paradigm very intuitive and felt $2\frac{1}{2}$D stacking was a significant improvement over viewing a set of diagrams side by side. However, their enthusiasm for this particular visualisation style is difficult to assess, since they are generally enthusiastic about any sort of automated tree manipulation tools. The tool that they most commonly use is a fairly straightforward tree drawing tool with few interactive analysis features. Until recently, it seems, advances in bioinformatics research were focused on computation of data. It is still "early days" for research into interactive systems supporting biologists' day-to-day analysis tasks.

As discussed in Section 7.4, limited test data was available, but this data was enough to demonstrate the efficacy of the system to biologists. Further experiments, possibly on artificially generated data, could test the layer-by-layer (or stratum-by-stratum) sweep application of the SLOP solver algorithm to larger stratified tree visualisations. Such a study might identify cases where the method does not converge to a reasonable solution. However, the general sweep approach has been proven effective by similar studies in layout of hierarchical graphs [107] and our initial tests seem to indicate similar performance for stratified tree visualisation.

In any case, it is not expected that a stratified tree visualisation would be applied in practical situations to sets of extremely large trees (greater than 100 or so leaves) or to sets of a large number of related trees (perhaps more than ten or so). This is simply because of the visual complexity that such a stratified tree visualisation would present.

Biologists were able to confirm for us that there is a need to compare large numbers of large related trees; however, in such cases the stratified tree visualisation system is probably more useful if applied in an overview-and-detail style system. That is, a biologist might use a system for exploring large numbers of trees, such as the multi-dimensional scaling approach by Klingner et al. [117], as an overview of the entire data-set and browse interesting subsets of trees using the stratified tree visualisation technique. The SLOP solver algorithm could then be run "on the fly" to produce the visualisation for the subset of interest.

In the case where the trees themselves are large, selecting subtrees for closer examination is

already a practice that biologists use regularly. Biologists are usually specialists who have a particular interest in smaller sets of related species. Therefore, they routinely examine only the subtree encompassing those species of interest, rather than the larger "tree of life" for all cellular organisms.

As discussed in Section 7.3, the SLOP-solver algorithm works well for binary trees. For trees of higher degree, however, the algorithm is (locally to each internal node) equivalent to the adjacent-exchange algorithm commonly used in crossing minimisation for hierarchical layout. As discussed in Section 3.2.1, the adjacent-exchange method for hierarchical layout is most effective when used as a post-processing refinement of methods based on heuristics such as barycentre or median placement. Variations of the SLOP-solver algorithm implementing equivalent heuristics could be implemented and studied. However, doing so may be of limited research interest since these methods have already been well studied for general hierarchical layout and, as has been demonstrated with the adjacent-exchange example, they are readily transferrable to SLOP.

Although, as described in Section 7.4, the results of most phylogeny inference programs are binary trees, feedback from biologists has suggested that there is sometimes a need to study trees with higher degree nodes. Particularly, this occurs when trees from different sources are merged to produce consensus trees. Consensus trees are discussed in more detail below in Section 7.6.1. Thus, although currently of limited research interest, the addition of heuristics to the SLOP-solver algorithm for efficiently handling higher degree nodes may be useful in a practical system.

Further experiments could also test the effectiveness of the "spine" connecting roots in the barrel tree layout in conveying the similarity of adjacent trees. When the barrel-layout style was demonstrated to biologists they expressed enthusiasm for the idea of the "spine", though they were concerned overall that the barrel layout may obscure internal structure and generally felt more comfortable with the standard phenogram layout style for each stratum in the stratifed tree visualisation.

7.6.1 Consensus Tree

A further refinement of the stratified tree visualisation approach is to pre-compute the *consensus tree* for the set of trees. A consensus tree is a single tree that is constructed by merging a larger set of related trees. Generally, it provides a summary, or precis of the set of trees, but the precise definition of how conflicting branching patterns from different trees (that is, differences in internal node structure) are merged can vary. For example, (see Nei et al. [145]):

Strict consensus tree — conflicting branching patterns are merged into a single

node with many descendent branches.

Majority consensus tree — a majority rule is applied to choose the branching pattern which occurs in most trees.

Adams consensus tree — the structure of the largest common sub-tree is preserved. Leaves not included in this common sub-tree are made direct descendants of the root.

Consensus trees play an important role in the tree comparison applications described by [117] and [142]. Consensus trees can generally be computed in linear time in a fairly straightforward way. Knowing which nodes are in the consensus tree is useful in two ways:

- Drawing the nodes in the consensus tree only on one plane, perhaps the topmost plane, might significantly simplify the visualisation. Nodes not in the consensus tree can still be drawn on parallel planes, thus focusing the user's attention to the areas of change.
- The SLOP-solver algorithm need only be applied to nodes that are not in the consensus tree. Thus, the number of nodes that need to be examined by the algorithm to reduce crossings for a set of trees that are largely similar is significantly reduced.

Another situation where consensus trees might play a role in stratified tree visualisation, is in eliding detail to provide a focus-and-context style browsing system. The limited scalability of the system, to sets of larger and more numerous trees due to visual complexity, has already been discussed. As in [142], large numbers of related trees can be reduced to representative consensus trees. A large set of trees could be dynamically reduced to a smaller set of representative trees producing a stratified tree visualisation of reasonable visual complexity which could be browsed as an overview. The analyst could then interactively obtain a detailed view of the constituent trees in any representative consensus tree in separate stratified tree visualisations.

7.6.2 Ordering of Strata

In the examples discussed in this chapter the trees are stacked according to an ordinal variable associated with the inference algorithm, where the ordinal variable is either processing time to produce the estimate or the probability that the tree is correct. Another possibility for improved comparison,

especially where no logical ordering is defined across the set of related trees, might be to find a stacking order for the trees that minimises differences between adjacent trees in the stack. Such an approach is analogous to the method used for finding a stacking order for metabolic pathways in Chapter 6. Note that the method used for calculating similarity measures between these pathways (general graphs with labelled nodes) does not extend to the trees with unlabelled internal nodes that are considered here. However, there are many well-known methods for measuring distances between arbitrary trees which would be appropriate. For example, a particularly famous inter-tree distance metric defined by Robinson and Foulds [161] is based on the minimum number of merging and splitting operations on internal nodes required to transform one tree into another.

CHAPTER 8

General Remarks, Conclusion and Further Work

This chapter provides a summary of the work presented in this thesis. Principles of $2\frac{1}{2}$D visualisation and the stratified graph visualisation paradigm are discussed further, with regard to the lessons learned from conducting the three case studies. Detailed ideas for further work are presented in the conclusions to each chapter in this thesis. However, some more general ideas for further work — particularly further work associated with different application domains — is also discussed here.

8.1 Principles of Two-and-a-Half-Dimensional Visualisation

In Chapter 2 the term "$2\frac{1}{2}$D" is found to be used inconsistently, not only in different industries and academic fields, but also within the body of visualisation literature. Information visualisation is defined in this chapter as a mapping from an information (data) space to two- or three-dimensional geometry and a set of graphical attributes such as glyphs, colour, texture and so on. A concrete definition for a $2\frac{1}{2}$D paradigm for information visualisation is established, in which position in the third spatial dimension is treated as a separate visual channel. This definition, however, does not dictate which data attribute in a particular data-set should be mapped to the third dimension. Based on an understanding of the limitations of human spatial perception some guidelines for choosing a $2\frac{1}{2}$D mapping to create an effective 3D visualisation are presented. These guidelines stipulate that the choice of data variable to map to the third spatial dimension should be *discrete*, *independent* and *direct*.

These $2\frac{1}{2}$D information visualisation principles are demonstrated in an extended application example which is then evaluated in an experimental study. The results of this study seem to indicate that 3D visualisations based on a $2\frac{1}{2}$D mapping can help analysts to understand complicated relationships involving more than two variables but that relationships between a pair of variables are better represented in 2D.

The primary advantage of the $2\frac{1}{2}$D visualisation paradigm over 2D visualisation is that it provides a spatial mapping where in 2D a third variable would have to be represented by some other graphical attribute such as colour, texture, symbol or motion. That is, it provides an extra (though, as described in Chapter 2, limited) dimension for the visualisation designer to use to create richer visualisations of high-dimensional data-sets. However, $2\frac{1}{2}$D is not advocated as dogma for all 3D information visualisation designers. Rather, it is just another visualisation tool that seems appropriate for certain domains.

A side result from this study is a recognition of the importance of interaction in choosing visual mappings. Different analytical tasks require different visualisations. Interactive visualisation systems should provide the user with the ability to choose an appropriate mapping for the task.

8.2 Two-and-a-Half-Dimensional Stratified Graph Visualisation

In Chapters 3 and 4, existing 3D graph visualisation methods are assessed in terms of their "$2\frac{1}{2}$D-ness". Generally, a 3D graph visualisation is considered to be $2\frac{1}{2}$D if the position of nodes is constrained to a set of parallel planes.

In Chapter 4, a $2\frac{1}{2}$D paradigm for visualising evolving graphs or comparing sets of related graphs is introduced. This paradigm is called *stratified graph visualisation* since edges are constrained to lie between nodes on the same plane and thus create a well defined set of levels or strata. Popular layout methods such as force-directed layout and hierarchical layout can be used to arrange stratified graphs. However, some modifications to these methods are provided which specifically aim to improve the results for the stratified graph visualisation paradigm.

Stratified graph visualisation is just one possible application of $2\frac{1}{2}$D design principles to graph visualisation. It is hoped that the contributions of this thesis, namely:

- the precise definitions given for $2\frac{1}{2}$D visualisation, $2\frac{1}{2}$D graph visualisation and stratified graph visualisation;
- the survey of existing graph visualisation methods that are effectively $2\frac{1}{2}$D; and
- the new examples of the application of $2\frac{1}{2}$D guidelines to graph visualisation;

will lead to a greater awareness of $2\frac{1}{2}$D design principles amongst graph visualisation practitioners. In turn, it is hoped that this leads to a unification of efforts in the field of 3D graph visualisation research to designing new visualisations that conform to the limitations of human spatial perception.

8.3 Application Case Studies

The $2\frac{1}{2}$D graph visualisation principles described above are explored further in three case studies. Evaluations of these case studies with domain experts produced considerable feedback, both positive and negative. One result, however, seems universal across all applications. Regardless of the specific domain the feedback emphasised the importance of providing the user with the ability to choose the mapping from data attributes to visualisation. This addresses two issues as follows.

- Flexibility of the visualisation system to supporting different lines of enquiry — for example, in the visualisation of fund-manager portfolios different styles of layout were found to be appropriate for different types of analysis, particularly:
  - absolute positioning or positioning relative to the index in the overview;
  - force-directed or hierarchical column layout in the detail view.
- Choosing a mapping that limits the visual complexity of a particular view potentially makes the visual paradigm scalable to larger data-sets. The visual triangulation system discussed in Chapter 6 is an example of a possible solution to this problem in the metabolic pathways domain.

8.3.1 Fund Manager Movement

Both the overview and detail visualisations proposed for analysis of movement in fund-manager portfolios are examples of application of stratified graph visualisation to evolving graphs.

The PORTFOLIOSPACE EXPLORER system for visualising a number of portfolios in a single overview, demonstrates that high dimensional data can be treated as a complete graph and visualised using a “worm” style stratified graph visualisation to show changes over time.

The PORTFOLIO COLUMNS view for detailed inspection of individual portfolios demonstrates two methods of stratified graph layout, that is, force-directed and hierarchical layout. Feedback from domain experts confirmed that force-directed layout is generally better at showing clustering (related groups of market sectors) while hierarchical layout is better for showing flow of money between market sectors.

8.3.2 Metabolic Pathways

The application incorporating experimental time-series data *in-situ* in a stratified graph visualisation is another example of $2\frac{1}{2}$D techniques used to support visual analysis of evolving graphs.

The application involving metabolic pathway comparison is an example of visualising sets of graphs classified according to a nominal variable. The problem of computing a stacking order across such a set of graphs is introduced as a new layout aesthetic for stratified graph visualisation.

The *Coordinated Visual Triangulation* system is an example of an interactive technique for navigating large sets of stratified graphs. It was designed in response to feedback from domain experts which questioned the scalability of the stratified-graph visualisation paradigm to larger sets of graphs. Initial feedback of the simple prototype coordinated visual triangulation system has been positive, but useful further work might involve a more in-depth user-study with a fully functional prototype to properly evaluate this paradigm.

8.3.3 Phylogenetic Trees

The application discussed in Chapter 7 is another example of stratified graph visualisation for comparing sets of graphs, but it is specific to the problem of comparing the structure of trees with common leaves. As such a new layout issue, the problem of ordering the leaves of each tree such that adjacent trees are arranged as similarly as possible, is introduced. This stratified leaf ordering problem (SLOP) is compared to the crossing minimisation problem found in hierarchical graph drawing. A heuristic for crossing minimisation from hierarchical graph drawing is adapted for the SLOP. The algorithm is demonstrated to obtain optimal solutions in polynomial time for the one-sided crossing minimisation problem.

8.4 Other Potential Application Domains

Generally, stratified graph visualisation should be applicable to any application domain involving an underlying evolving graph. The quintessential example given by Ferreira [63] of such a domain is the mobile ad-hoc network which is predicted to become an increasingly important communication medium in the future. In such a network mobile devices function as transient hosts and the topology of the network is extremely volatile. As such, visualisation tools supporting analysis of problems in these networks are likely to prove useful.

Software engineering has a number of promising applications for stratified graph visualisation. Object-oriented software architectures are commonly represented by class hierarchies or graphs of references between different software components. Throughout its development and maintenance software typically evolves to fit in with changing real-world requirements. Thus, the underlying architectural diagrams are evolving graphs, and the history of this evolution may be interesting to observe using stratified graph visualisation. Frequently reused architectural designs are known as design patterns [75]. Stratified graph visualisation could be used to compare different software components to aid in the search for such patterns.

The ideas for exploring high-dimensional evolving data-sets presented in the PORTFOLIOSPACE EXPLORER also potentially have broad application. In the social sciences, high-dimensional data-sets, for example experimental results from different subjects or groups, are common. The $2\frac{1}{2}$D "worm-view" style of visualisation may be applicable when the analyst wishes to compare the results from different groups or classes or when he or she wishes to analyse a set of results changing over time.

8.5 Closing Remarks

This thesis has presented visualisation paradigms and guidelines for implementation of these paradigms. These are intended to assist with the creation of information visualisation and graph visualisation applications that take advantage of 3D graphics in a way that is congruent with the limitations of human spatial perception. It is hoped that as technologies such as augmented reality and holographic displays become more sophisticated, knowledge of how to use them effectively for information visualisation will eventually lead to more widespread adoption of such techniques in different application domains.

Bibliography

[1] Adachi, J. and Hasegawa, M., *PROTML: Maximum likelihood inference of protein phylogeny*, in Computer Science Monographs, Institute of Statistical Mathematics, Tokyo, 1992, pp. 1–77. 1.4.3, 6.4

[2] Ahmed, A., Dwyer, T., Murray, C., Song, L., and Wu, Y. X., "Wilmascope graph visualization," *Proceedings of the IEEE Symposium on Information Visualization (InfoVis'04)*, Vol. To Appear, 2004. 4.1

[3] Alizadeh, F., Karp, R. M., Weisser, D. K., and Zweig, G., "Physical mapping of chromosomes using unique probes," *Proceedings of the 5th ACM-SIAM Symposium on Discrete Algorithms (SODA '94)*, 1994, pp. 489–500. 6.3.3

[4] Appel, R. D., Bairoch, A., and Hochstrasser, D. F., "A new generation of information retrieval tools for biologists: the example of the expasy www server," *Trends in Biochemical Science*, Vol. 19, No. 6, 1994, pp. 258–260. 1.4.2

[5] Ark, W., Dryer, C. D., Selker, T., and Zhai, S., "Representation matters: The effect of 3D objects and a spatial metaphor in a graphical user interface," *People and Computers XIII, Proceedings of HCI'98*, Springer, 1998, pp. 209–219. 1.3

[6] artLab, "The industry's foundation for high performance graphics," `http://www.opengl.org/`, 2004. 2.1

[7] Au, S. C., Leckie, C., Parhar, A., and Wong, G., "Efficient visualization of large routing topologies," *Int. J. Netw. Manag.*, Vol. 14, No. 2, 2004, pp. 105–118. 4.1

[8] Bar-Joseph, Z., Demaine, E. D., Gifford, D. K., Hamel, A. M., Jaakkola, T. S., and Srebro, N., "K-ary clustering with optimal leaf ordering for gene expression data," *Proc. Intl. Workshop Algorithms in Bioinformatics (WABI 2002)*, Vol. 2452 of *Lecture Notes in Computer Science*, Springer, 2002, pp. 506–520. 1.4.3, 6.4.2, 7.3

[9] Barabási, A.-L. and Bonabeau, E., "Scale-free networks," *Scientific American*, May 2003, pp. 50–59. 4.1

[10] Barth, W., Jünger, M., and Mutzel, P., "Simple and efficient bilayer cross counting," *Proceedings of the 10th International Symposium on Graph Drawing (GD'02)*, Vol. 2528 of *Lecture Notes in Computer Science*, Springer, Berlin, 2002, pp. 130–141. 4.3.2, 7.3

[11] Battista, G. D., Eades, P., Tamassia, R., and Tollis, I., *Graph Drawing: Algorithms for the Visualization of Graphs*, Prentice-Hall, Englewood Cliffs, NJ, 1999. 3.2, 3.2, 3.2.1, 3.2.1, 4.3.2, 7.3, 7.3

[12] Becker, M. Y. and Rojas, I., "A graph layout algorithm for drawing metabolic pathways," *Bioinformatics*, Vol. 17, No. 5, 2001, pp. 461–467. 6.1.2

[13] Booth, K. S. and Lueker, G. S., "Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms," *Journal of Computer and System Sciences*, Vol. 13, No. 3, 1976, pp. 335–379. 6.3.3

[14] Borg, I. and Groenen, P., *Modern Multidimensional Scaling: Theory and Applications*, Springer Series in Statistics, Springer, 1997. 3.2.1, 5.3

[15] Bragdon, C., *The Frozen Fountain*, Kessinger, 1924. 2.1

[16] Brandes, U. and Corman, S. R., "Visual unrolling of network evolution and the analysis of dynamic discourse," *Information Visualization*, Vol. 2, No. 1, 2003, pp. 40–50. 4.3.1

[17] Brandes, U., Dwyer, T., and Schreiber, F., "Visual triangulation of network-based phylogenetic trees," *Proc. 6th Joint Eurographics - IEEE TCVG Symp. Visualization*, 2004, to appear. 3

[18] Brandes, U., Dwyer, T., and Schreiber, F., "Visual understanding of metabolic pathways across organisms using layout in two and a half dimensions," *Journal of Integrative Bioinformatics*, Vol. 2, 2004. 2

[19] Brandes, U., Dwyer, T., and Schrieber, F., "Visualizing related metabolic pathways in two and a half dimensions," *Proceedings of the 11th International Symposium on Graph Drawing GD'03*, Vol. 2912 of *Lecture Notes in Computer Science*, Springer, 2003, pp. 111–122. 2

[20] Brandes, U., Kääb, V., Löh, A., Wagner, D., and Willhalm, T., "Dynamic WWW structures in 3D," *Journal of Graph Algorithms and Applications*, Vol. 4, No. 3, 2000, pp. 183–191. 3.2.1

[21] Brandes, U. and Wagner, D., "A bayesian paradigm for dynamic graph layout," *Proceedings of the 5th International Symposium on Graph Drawing (GD'97)*, Vol. 1353 of *Lecture Notes in Computer Science*, Springer, 1997, pp. 236–247. 4.3

[22] Brandes, U. and Wagner, D., "Dynamic grid embedding with few bends and changes," *Proceedings of the 9th Annual International Symposium on Algorithms and Computation (ISAAC'98)*, Vol. 1533 of *Lecture Notes in Computer Science*, Springer, 1998, pp. 89–98. 4.3

[23] Brandes, U. and Willhalm, T., "Visualization of bibliographic networks with a reshaped landscape metaphor," *Proceedings of the Joint EUROGRAPHICS - IEEE TCVG Symposium on Visualization (VisSym'02)*, ACM Press, 2002, pp. 159–164. 4.1

[24] Brands, S., Gallagher, D., and Looi, A., "Active investment manager portfolios and preferences for stock characteristics: Australian evidence," Tech. rep., The University of New South Wales, 2004. 5.1

[25] Brodbeck, D., Chalmers, M., Lunzer, A., and Cotture, P., "Domesticating bead: Adapting an information visualization system to a financial institution," *Proceedings of the IEEE Symposium on Information Visualization*, 1997, pp. 73–90. 1.4.1

[26] Brooks, F. P., "Grasping reality through illusion — interactive graphics serving science," *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM Press, 1988, pp. 1–11. 2.1

[27] Buchholz, A., Takors, R., and Wandrey, C., "Quantification of intracellular metabolites in escherichia coli k12 using liquid chromatographicelectrospray ionization tandem mass spectrometric techniques," *Analytical Biochemistry*, Vol. 295, 2001, pp. 129–137. 6.2

[28] Card, S. K., MacKinlay, J. D., Shneiderman, B., and Card, M., *Readings in Information Visualization: Using Vision to Think*, Morgan Kaufmann Series in Interactive Technologies, Academic Press, 1999. 1.1, 2.2.3, 4.4, 5.2

[29] Carmel, L., Harel, D., and Koren, Y., "Drawing directed graphs using one-dimensional optimization," *Proceedings of the 10th International Symposium on Graph Drawing (GD'02)*, Vol. 2528 of *Lecture Notes in Computer Science*, Springer, 2002, pp. 193–206. 3.2.1

[30] Cattell, R. B., "Factor analysis: an introduction to essentials," *Biometrics*, Vol. 21, 1965, pp. 190–215. 5.4.2

[31] Chen, C., "Bridging the gap: The use of pathfinder networks in visual navigation," *Journal of Visual Languages and Computing*, Vol. 9, No. 3, 1998, pp. 267–286. 4.1

[32] Chuah, M. C., Roth, S. F., Mattis, J., and Kolojejchick, J., "SDM: Selective dynamic manipulation of visualizations," *ACM Symposium on User Interface Software and Technology*, 1995, pp. 61–70. 4.1

[33] Cockburn, A., "Revisiting 2d vs 3d implications on spatial memory," *Proceedings of the fifth conference on Australasian user interface*, Australian Computer Society, Inc., 2004, pp. 25–31. 1.3

[34] Cockburn, A. and McKenzie, B., "3d or not 3d?: evaluating the effect of the third dimension in a document management system," *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM Press, 2001, pp. 434–441. 1.3

[35] Cockburn, A. and McKenzie, B., "Evaluating the effectiveness of spatial memory in 2D and 3D physical and virtual environments," *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM Press, 2002, pp. 203–210. 2.1

[36] Coren, Y., Carmel, L., and Harel, D., "ACE: A fast multiscale eigenvectors computation for drawing huge graphs," *Proceedings of the IEEE Symposium on Information Visualization (InfoVis 2002)*, IEEE, 2002, pp. 137–144. 3.2.1

[37] Dandekar, T., Schuster, S., Snel, B., Huynen, M., and Bork, P., "Pathway alignment: application to the comparative analysis of glycolytic enzymes," *Biochemical Journal*, Vol. 343, 1999, pp. 115–124. 6.3, 6.4.1

[38] Davidson, G. S., Wylie, B. N., and Boyack, K. W., "Cluster stability and the use of noise in interpretation of clustering," *Proceedings of the IEEE Symposium on Information Visualization 2001 (INFOVIS'01)*, IEEE Computer Society, 2001, p. 23. 4.1

[39] Delwiche, C., Kuhsel, M., and Palmer, J., "Phylogenetic analysis of tufa sequences indicates a cyanobacterial origin of all plastids," *Molecular Phylogenetics and Evolution*, Vol. 4, No. 2, 1995, pp. 110–128. 8, 6.17

[40] Diehl, S. and Görg, C., "Graphs, they are changing: Dynamic graph drawing for a sequence of graphs," *Proceedings of the 10th International Symposium on Graph Drawing (GD'02)*, Vol. 2528 of *Lecture Notes in Computer Science*, Springer, 2003, pp. 23–30. 4.3

[41] Diehl, S., Görg, C., and Kerren, A., "Preserving the mental map using foresighted layout," *Proceedings of Joint Eurographics – IEEE TCVG Symposium on Visualization (VisSym'01)*, IEEE, 2001, pp. 175–184. 4.3

[42] Diestel, R., *Graph Theory*, Springer, second edition ed., 2000. 3.1

[43] do Nascimento, H. and Eades, P., "User hints for directed graph drawing," *Proceedings of the 9th Symposium on Graph Drawing (GD'01)*, Vol. 2265 of *Lecture Notes in Computer Science*, Springer, 2001, pp. 205–219. 3.2

[44] Dodson, D., "Comaide: Information visualization using cooperative 3d diagram layout," *Proceedings of the 3rd International Symposium on Graph Drawing (GD'95)*, Vol. 1027 of *Lecture Notes in Computer Science*, Springer, 1995, pp. 190–201. 2.1, 3.2.1

[45] Dwyer, T., "Three dimensional UML using force directed layout," *Australian Symposium on Information Visualisation, (invis.au 2001)*, edited by P. Eades and T. Pattison, ACS, Sydney, Australia, 2001. 3.2.1

[46] Dwyer, T., "A scalable method for visualising changes in portfolio data," *Proc. of the Australian Symp. on Information Visualisation (invis.au)*, Vol. 24 of *Conferences In Research and Practice in Information Technology*, Australian Computer Society, 2003. 5

[47] Dwyer, T. and Eades, P., "Visualising a fund manager flow graph with columns and worms," *Proceedings of the 6th International Conference on Information Visualisation, IV02*, IEEE Computer Society, 2002. 4.3.1, 5, 5.5

[48] Dwyer, T. and Eckersley, P. B.1

[49] Dwyer, T. and Gallagher, D., "Visualising changes in fund manager holdings in two and a half dimensions," *Journal of Information Visualisation - to appear*, 2004. 5

[50] Dwyer, T., Rolletschek, H., and Schreiber, F., "Representing experimental biological data in metabolic networks," *Proceedings of the second conference on Asia-Pacific bioinformatics*, Australian Computer Society, Inc., 2004, pp. 13–20. 1

[51] Dwyer, T. and Schreiber, F., "Optimal leaf ordering for two and a half dimensional phylogenetic tree visualisation," *Proceedings of The Australian Symposium on Information Visualisation (InVis.au'04)*, Vol. 35 of *CRPIT*, 2004, pp. 109–115. 7

[52] Eades, P., "A heuristic for graph drawing," *Congress Numerantium*, Vol. 42, 1984, pp. 149–160. 3.2.1

[53] Eades, P. and Feng, Q., "Multilevel visualization of clustered graphs," *GD'96 Lecture Notes in Computer Science*, Vol. 1190, Springer-Verlag, 1996, pp. 101–112. 4.1

[54] Eades, P. and Kelly, D., "Heuristics for reducing crossings in 2-layered networks," *Ars Combinatoria*, Vol. 21.A, 1986, pp. 89–98. 4.3.2

[55] Eades, P. and Lin, X., "Spring algorithms and symmetry," *Theor. Comput. Sci.*, Vol. 240, No. 2, 2000, pp. 379–405. 3.2.1

[56] Eades, P. and Sugiyama, K., "How to draw a directed graph," *Journal of Information Processing*, Vol. 13, 1990, pp. 424–437. 3.2.1

[57] Eades, P. and Wormald, N., "Edge crossings in drawings of bipartite graphs," *Algorithmica*, Vol. 10, 1994, pp. 379–403. 4.3.2, 4.3.2, 7.3

[58] Eiseley, L. C., *The Immense Journey*, Vintage: New York, 1957th ed., 1946. 7

[59] Everitt, B. S. and Dunn, G., *Applied Multivariate Data Analysis*, Arnold London, 2001. 2.2.2, 5.3

[60] Falkenstein, E., "Preferences for stock characteristics as revealed by mutual fund portfolio holdings," *Journal of Finance*, Vol. 51, No. 1, 1996, pp. 111–135. 5.1

[61] Felsenstein, J., "Phylip - phylogenetic inference package," *Cladistics*, Vol. 5, 1989, pp. 164–166, See `http://evolution.genetics.washington.edu/phylip.html`. 1.4.3, 6.4, 6.4.3

[62] Feng, Q., *Algorithms for drawing clustered graphs*, Ph.D. thesis, University of Newcastle, 1997. 3.1

[63] Ferreira, A., "Building a reference combinatorial model for dynamic networks: Initial results in evolving graphs," Tech. Rep. 5041, INRIA, 2003. 4.2, 8.4

[64] Fetter, W. A., *Computer Graphics*, WA Benjamin, 1968, pp. 397–418. 1.2

[65] Fiehn, O., Kopka, J., Dörmann, P., Altmann, T., Trethewey, R. N., and Willmitzer, L., "Metabolite profiling for plant functional genomics," *Nature Biotechnology*, Vol. 18, 2000, pp. 1157–1161. 6.2

[66] Fisk, C. J. and Isett, D. D., ""accel" automated circuit card etching layout," *DAC '65: Proceedings of the SHARE design automation project*, ACM Press, New York, NY, USA, 1965, pp. 9.1–9.31. 3.2.1

[67] Fitch, W. M., "On the problem of discovering the most parsimonious tree," *The American Naturalist*, Vol. 111, 1977, pp. 223–257. 1.4.3, 6.4

[68] Forst, C. V. and Schulten, K., "Phylogenetic analysis of metabolic pathways," *Journal Molecular Evolution*, Vol. 52, 2001, pp. 471–489. 6.3, 6.3.1, 6.4

[69] Forster, M., Pick, A., Raitner, M., Schreiber, F., and Brandenburg, F. J., "The system architecture of the biopath system," *In Silico Biology*, Vol. 2, No. 3, 2002, pp. 415–426. 1.4.2, 6.1.3, 6.3.2

[70] Frick, A., Ludwig, A., and Mehldau, H., "A fast adaptive layout algorithm for undirected graphs," *Proceedings of the 2nd International Symposium on Graph Drawing (GD'94)*, Vol. 894, Springer, 1994, pp. 388–403. 3.2.1

[71] Friedrich, C. and Eades, P., "Graph drawing in motion," *Journal of Graph Algorithms and Applications*, Vol. 6, No. 3, 2002, pp. 353–370. 3.2, 4.4

[72] Friedrich, C. and Houle, M. E., "Graph drawing in motion ii," *Proceedings of the 9th International Symposium on Graph Drawing (GD'01)*, Vol. 2265 of *Lecture Notes in Computer Science*, Springer, 2001, pp. 220–231. 3.2, 4.4

[73] Fruchterman, T. and Reingold, E., "Graph drawing by force-directed placement," *Software Practice and Experience*, Vol. 21, No. 11, 1991, pp. 1129–1164. 3.2.1

[74] Gallagher, D. R., *Investment Manager Characteristics, strategy and performance*, Ph.D. thesis, The University of Sydney, 2002. 1.4.1

[75] Gamma, E., Helm, R., Johnson, R., and Vlissides, J., *Design Patterns*, Addison-Wesley, 1995. 8.4

[76] Gansner, E. R., Koutsofios, E., North, S. C., and Vo, K.-P., "A technique for drawing directed graphs," *IEEE Trans. Softw. Eng.*, Vol. 19, No. 3, 1993, pp. 214–230. 4.3.2, 4.3.2

[77] Gershon, N. and Eick, S. G., "Information visualization," *Computer Graphics and Applications*, Vol. 17, 1997, pp. 29–31. 1.3

[78] Gibson, J. J., *The Ecological Approach to Visual Perception*, Houghton-Mifflin, 1979. 2.2.2

[79] Gibson, W., *Neuromancer*, Ace Books New York, 1984. 1

[80] Giertsen, C. and Lucas, A., "3d visualization for 2d gis: an analysis of the users needs and a review of techniques," *Computer Graphics Forum*, Vol. 13, No. 3, 1994, pp. 1–12. 2.1

[81] Gilbert, E. W., "Pioneer maps of health and disease in england," *Geographical Journal*, Vol. 124, 1958, pp. 172–183. 1.1

[82] Goldberg, P. W., Golumbic, M. C., Kaplan, H., and Shamir, R., "Four strikes against physical mapping of DNA," *Journal of Computational Biology*, Vol. 2, No. 1, 1995, pp. 139–152. 6.3.3

[83] Goldman, N. and Narayanaswamy, K., "Software evolution through iterative prototyping," *Proceedings of the 14th international conference on Software engineering*, ACM Press, 1992, pp. 158–172. 1.5

[84] Goto, S., Okuno, Y., Hattori, M., Nishioka, T., and Kanehisa, M., "LIGAND: database of chemical compounds and reactions in biological pathways," *Nucleic Acid Research*, Vol. 30, 2002, pp. 402–404. 6.1.3

[85] Gresh, D., Rogowitz, B., Tignor, M., and Maryland, E., "An interactive framework for visualizing foreign currency exchange options," *Proceedings of the Conference on Visualization*, IEEE Computer Society Press, 1999, pp. 453–456. 1.4.1

[86] Gross, M., Sprenger, T., and Finger, J., "Visualizing information on a sphere," *Proceedings of the IEEE Symposium on Information Visualization (InfoVis'97)*, 1997, pp. 11–16. 1.4.1

[87] Hall, K. M., "An r-dimensional quadratic placement algorithm," *Management Science*, Vol. 17, No. 3, 1970, pp. 219–229. 2

[88] Hamming, R. W., *Numerical Methods for Scientists and Engineers*, McGraw-Hill, New York, 1962. 5

[89] Harel, D. and Koren, Y., "A fast multi-scale method for drawing large graphs," *Proceedings of the 8th International Symposium on Graph Drawing (GD2000)*, Vol. 1984, Springer, 2000, pp. 183–196. 3.2.1

[90] Harel, D. and Koren, Y., "Graph drawing by high-dimensional embedding," *Proceedings of the 10th International Symposium on Graph Drawing (GD'02)*, Vol. 2528 of *Lecture Notes in Computer Science*, Springer, 2002, pp. 207–219. 3.2.1, 5.4.2

[91] Hendrix, D., Cross, J. H., and Maghsoodloo, S., "The effectiveness of control structure diagrams in source code comprehension activities," *IEEE Transactions on Software Engineering*, Vol. 28, No. 5, 2002, pp. 463–477. 2.4.2, 2.4.5

[92] Heymans, M. and Singh, A. K., "Deriving phylogenetic trees from the similarity analysis of metabolic pathways," *Bioinformatics*, Vol. 19, No. Suppl. 1, 2003, pp. 138–146. 6.4

[93] Himsolt, M., "Graphlet: Design and implementation of a graph editor," *Software – Practice and Experience*, Vol. 30, No. 11, pp. 1303. 1

[94] Himsolt, M., "GML: A portable graph file format," Tech. rep., University of Passau, 1997. 1

[95] Hofestädt, R. and Thelen, S., "Qualitative modeling of biochemical networks," *In Silico Biology*, Vol. 1, No. 1, 1998, pp. 39–53. 6.1.1

[96] Holmes, E. C., Bollyky, P. L., Nee, S., Rambaut, A., Garnett, G., and Harvey, P. H., *New Uses for New Phylogenies*, chap. Using phylogenetic trees to reconstruct the history of infectious disease epidemics, Oxford University Press, 1996, pp. 169–186. 6.4

[97] Hong, S., "Drawing graphs symmetrically in three dimensions," *Proceedings of the 9th International Symposium on Graph Drawing (GD'01)*, Vol. 2265 of *Lecture Notes in Computer Science*, Springer, 2002, pp. 189–204. 3.2.1

[98] Hong, S. and Eades, P., "Drawing trees symmetrically in three dimensions," *Algorithmica*, Vol. 36, No. 2, 2003. 3.2.1

[99] Hong, S. and Murtagh, T., "Polyplane: a new layout algorithm for trees in three dimensions," Tech. Rep. IT-IVG-2003-01, University of Sydney, 2003. 3.2.1

[100] Hosobe, H., "A high-dimensional approach to interactive graph visualization," *Proceedings of the 19th Annual ACM Symposium on Applied Computing (SAC2004)*, Vol. 2, ACM Press, 2004, pp. 1253–1257. 3.2.1

[101] Howell, D. C., *Fundamental Statistics for the Behavioural Sciences*, Duxbury Press, 4th ed., 1999. 1.5, 2.4.2, 2.4.5

[102] Huang, M. L., Eades, P., and Wang, J., "On-line animated visualization of huge graphs using a modified spring algorithm," *Journal of Visual Languages and Computing*, Vol. 9, No. 6, 1998, pp. 623–645. 4.3

[103] Hughes, T., Hyun, Y., and Liberles, D. A., "Visualising very large phylogenetic trees in three dimensional hyperbolic space," *BMC Bioinformatics*, Vol. 5, No. 1, 2004, pp. 48–53. 1.4.3

[104] id software, "id history," `http://www.idsoftware.com/business/history/`, 2004. 2.1

[105] Jacobs, P. F., *Rapid Prototyping and Manufacturing: Fundamentals of Stereolithography*, Society of Manufacturing Engineers, 1992. 2.1

[106] Jick, T. D., "Mixing qualitative and quantitative methods: Triangulation in action," *Administrative Science Quarterly*, Vol. 24, 1979, pp. 602–611. 6.4, 6.4.2

[107] Jünger, M. and Mutzel, P., "2-layer straightline crossing minimization: Performance of exact and heuristic algorithms," *Journal of Graph Algorithms and Applications (JGAA)*, Vol. 1, No. 1, 1997, pp. 1–25. 4.3.2, 4.3.2, 7.6

[108] Kamada, T., *Visualizing Abstract Objects and Relations*, World Scientific, 1989. 3.2

[109] Kamada, T. and Kawai, S., "An algorithm for drawing general undirected graphs," *Information Processing Letters*, Vol. 31, No. 1, 1989, pp. 7–15. 3.2.1

[110] Kanehisa, M. and Goto, S., "Kegg: kyoto encyclopedia of genes and genomes," *Nucleic Acid Research*, Vol. 28, 2000, pp. 27–30. 1.4.2, 6.3.2

[111] Karp, P. D. and Paley, S. M., "Automated drawing of metabolic pathways," *Proceedings of the 3rd International Conference on Bioinformatics and Genome Research*, 1994, pp. 225–238. 6.1.2

[112] Kasmarik, K. and Thurbon, J., "Experimental evaluation of a program visualisation tool for use in computer science education," *Proceedings of the Australian Symposium on Information Visualisation, 2003*, Vol. 24 of *CRPIT*, Australian Computer Society, 2003, pp. 111–116. 2.4.2, 2.4.5

[113] Kaufmann, M. and Wagner, D., editors, *Drawing Graphs: Methods and Models*, Vol. 2025 of *Lecture Notes in Computer Science*, Springer, 2001. 3.2, 3.2.1, 3.2.1, 3.2.1, 4.3.2

[114] Keim, D. A., "Designing pixel-oriented visualization techniques: Theory and applications," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 6, No. 1, 2000, pp. 59–78. 6.3.3

[115] Keskin, C. and Vogelmann, V., "Effective visualization of hierarchical graphs with the cityscape metaphor," *Workshop on New Paradigms in Information Visualization and Manipulation*, 1997, pp. 52–57. 4.1

[116] Kirk, D. B., "The future of graphics computing," Tech. rep., NVIDIA Corporation, 2003. 1.2

[117] Klingner, J. and Amenta, N., "Case study: Visualing sets of evolutionary trees," *Proceedings of IEEE Information Visualization, 2002*, 2002, pp. 71–74. 1.4.3, 7.2.1, 7.6, 7.6.1

[118] Koike, H., "The role of another spatial dimension in software visualization," *ACM Trans. Inf. Syst.*, Vol. 11, No. 3, 1993, pp. 266–286. 4.2

[119] Koren, Y. and Carmel, L., "Visualization of labeled data using linear transformations," *Proceedings of the IEEE Symposium on Information Visualization (InfoVis 2003)*, IEEE, 2003, pp. 121–128. 5.3

[120] Kraak, M.-J., "Geovisualization illustrated," *ISPRS Journal of Photogrammetry and Remote Sensing*, Vol. 57, 2003, pp. 390–399. 1.3(b)

[121] Krikke, J., "A chinese perspective for cyberspace," *The International Institute for Asian Studies Newsletter*, 1996. 2.1, 2

[122] Kruskal, J. B., "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis," *Psychometrika*, Vol. 29, 1964, pp. 1–64. 3.2.1

[123] Kruskal, J. B. and Seery, J. B., "Designing network diagrams," *Proceedings of the First General Conference on Social Graphics*, U.S. Dept. of the Census, 1980, pp. 22–50. 3.2.1

[124] Levy, E., Zacks, J., Tversky, B., and Schiano, D., "Gratuitous graphics? putting preferences in perspective," *Symposium on Computer Human Interaction (CHI'96)*, ACM, 1996, pp. 42–49. 1.3

[125] Liao, L., Kim, S., and Tomb, J. F., "Genome comparisons based on profiles of metabolic pathways," *Proceedings of the 6th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES '02)*, 2002, pp. 469–476. 6.3, 6.3.1, 6.4

[126] Lipton, R. J., North, S. C., and Sandberg, J. S., "A method for drawing graphs," *Proceedings of the first annual symposium on Computational geometry*, ACM Press, 1985, pp. 153–160. 3.2.1

[127] Makarenkov, V., "T-REX: reconstructing and visualizing phylogenetic trees and reticulation networks," *Bioinformatics*, Vol. 17, No. 7, 2001, pp. 664–668. 1.4.3

[128] Manning, J., *Geometric Symmetry in Graphs*, Ph.D. thesis, Purdue University, 1990. 3.2.1

[129] Marr, D., *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*, W. H. Freeman, 1982. 2.1

[130] Marr, D. and Nishihara, H. K., "Representation and recognition of the spatial organization of three-dimensional shapes," *Proceedings of the Royal Society of London*, Vol. 200 of *B*, The Royal Society, 1978, pp. 269–294. 2.1

[131] Martin, G. E., *Transformation Geometry, an Introduction to symmetry*, Springer, 1982. 3.2.1

[132] Mayer, M. L. and Hieter, P., "Protein networks — built by association," *Nature Biotechnology*, Vol. 18, No. 12, 2000, pp. 1242–1243. 1.4.2

[133] McCormick, B. H., DeFanti, T. A., and Brown, M. D., "Visualization in scientific computing," *Computer Graphics*, Vol. 21, No. 6, 1987. 1.1

[134] McCracken, P. and Others, *Towards Full Employment and Price Stability*, Organisation for Economic Co-Operation and Development (OECD), 1977. 2.3

[135] Merrick, D., *Skeletal Animation for the Exploration of Graphs*, Master's thesis, School of Information Technologies, The University of Sydney, 2002. 3.2

[136] Michal, G., "Biochemical pathways (poster)," 1993. 1.4.2, 1.8, 6.3.2

[137] Michal, G., *Biochemical Pathways*, Spektrum Akademischer, 1999. 1.4.2

[138] Microsystems, S., "Project looking glass," `http://www.sun.com/software/looking_glass`, 2004. 2.1

[139] Misue, K., Eades, P., Lai, W., and Sugiyama, K., "Layout adjustment and the mental map," *Journal of Visual Languages and Computing*, Vol. 6, No. 2, 1995, pp. 183–210. 4.4

[140] Morris, S., Asnake, B., and Yen, G., "Optimal dendrogram seriation using simulated annealing," *Information Visualization*, Vol. 2, No. 2, 2003, pp. 95–104. 1.4.3, 7.3

[141] Munzner, T., "Exploring large graphs in 3d hyperbolic space," *IEEE Computer Graphics and Applications*, Vol. 18, No. 4, 1998, pp. 18–23. 3.2.1

[142] Munzner, T., Guimbretiére, F., Tasiran, S., Zhang, L., and Zhou, Y., "Treejuxtaposer: Scalable tree comparison using focus+context with guaranteed visibility," *Proceedings of SIGGRAPH*, 2003. 1.4.3, 7.2.1, 7.6.1

[143] Murgai, R., Fujita, M., and Krishnan, S. C., "Data sequencing for minimum-transition transmission," *Proceedings of the 9th IFIP International Conference on Very Large Scale Integration (VLSI '97)*, 1997. 6.3.3, 6.3.3

[144] Nakao, M., Bono, H., Kawashima, S., Kamiya, T., Sato, K., Goto, S., and Kanehisa, M., "Genome-scale gene expression analysis and pathway reconstruction in kegg," *Genome Informatics*, Vol. 10, 1999, pp. 94–103. 6.2

[145] Nei, M. and Kumar, S., *Molecular Evolution and Phylogenetics*, Oxford University Press, 2000. 7.6.1

[146] Nesbitt, K., *Designing Multi-sensory Displays for Abstract Data*, Ph.D. thesis, University of Sydney, 2003. 1.4.1

[147] Nesbitt, K. and Friedrich, C., "Applying gestalt principals to animated visualizations of network data," *Proceedings of the Sixth International Conference on Information Visualisation (IV'02)*, IEEE Computer Society, 2002, pp. 737–745. 3.2

[148] North, S. C. and Woodhull, G., "Online hierarchical graph drawing," *Proceedings of the 9th International Symposium on Graph Drawing (GD'01)*, Vol. 2265 of *Lecture Notes in Computer Science*, Springer, 2002, pp. 232–246. 4.3

[149] Ogata, H., Fugibuchi, W., Goto, S., and Kanehisa, M., "A heuristic graph comparison algorithm and its application to detect functionally related enzyme clusters," *Nucleic Acids Research*, Vol. 28, No. 20, 2000, pp. 4021–4028. 6.3.1

[150] Page, R. D. M., "Treeview: An application to display phylogenetic trees on personal computers," *Computer Applications in the Biosciences*, Vol. 12, No. 4, 1996, pp. 357–358. 1.4.3

[151] Parker, G., Franck, G., and Ware, C., "Visualization of large nested graphs in 3d: Navigation and interaction," *Journal of Visual Languages and Computing*, Vol. 9, No. 3, 1998, pp. 299–317. 2.4.2, 3.2, 3.2.1

[152] Pinnuck, M., "Stock preferences and derivatives activities of australian fund managers," *Accounting and Finance*, Vol. 44, No. 1, 2004, pp. 97–120. 5.1

[153] Purchase, H., "Which aesthetic has the greatest effect on human understanding," *Proceedings of the 5th International Symposium on Graph Drawing (GD'97)*, Vol. 1353, Springer, 1997, pp. 248–261. 3.2, 3.2.1

[154] Quigley, A. and Eades, P., "Fade: Graph drawing, clustering, and visual abstraction," *Proceedings of the 8th International Symposium on Graph Drawing (GD2000)*, Vol. 1984, Springer, 2000, pp. 197–210. 3.2.1

[155] Reddy, V. N., Mavrovouniotis, M. L., and Liebman, M. N., "Petri net representations of metabolic pathways," *Proceedings of the 1st International Conference on Intelligent Systems for Molecular Biology (ISMB'93)*, 1993, pp. 328–336. 6.1.1

[156] Reingold, E. and Tilford, J., "Tidier drawings of trees," *IEEE Transaction on Software Engineering*, Vol. SE-7, No. 2, 1981, pp. 223–228. 3.2.1

[157] Roberts, J. C., editor, *Proceedings of the conference on Coordinated and Multiple Views In Exploratory Visualization*. IEEE Computer Society, 2003. 6.4.2

[158] Robertson, G. G., Czerwinski, M., Larson, K., Robbins, D. C., Thiel, D., and van Dantzich, M., "Data mountain: Using spatial memory for document management," *ACM Symposium on User Interface Software and Technology*, 1998, pp. 153–162. 1.3, 2.1

[159] Robertson, G. G., Mackinlay, J. D., and Card, S. K., "Cone trees: animated 3d visualizations of hierarchical information," *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM Press, 1991, pp. 189–194. 3.2.1, 4.1

[160] Robinson, A. H., "The thematic maps of Charles Joseph Minard," *Imago Mundi*, Vol. 21, 1967, pp. 95–108. 1.1, 1.3(a)

[161] Robinson, D. and Foulds, L., "Comparison of phylogenetic trees," *Mathematical Biosciences*, Vol. 53, 1981, pp. 131–147. 7.6.2

[162] Roessner, U., Wagner, C., Kopka, J., Trethewey, R. N., and Willmitzer, L., "Simultaneous analysis of metabolites in potato tuber by gas chromatography-mass spectrometry," *Plant Journal*, Vol. 23, 2000, pp. 131–142. 6.2

[163] Roll, R., "A mean/variance analysis of tracking error," *Journal of Portfolio Management*, Vol. 18, No. 4, 1992, pp. 13–22. 5.4.1

[164] Rost, U. and Bauer-Bornberg, E., "Treewiz: interactive exploration of huge trees," *Bioinformatics*, Vol. 18, No. 1, 2002, pp. 109–114. 1.4.3

[165] Ryall, K., Marks, J., and Shieber, S. M., "An interactive constraint-based system for drawing graphs," *ACM Symposium on User Interface Software and Technology*, 1997, pp. 97–104. 3.2

[166] Saitou, N. and Nei, M., "The neighbor-joining method: a new method for reconstructing phylogenetic trees," *Molecular Biology and Evolution*, Vol. 4, No. 4, 1987, pp. 406–425. 6.4.3

[167] Scaife, M. and Rogers, Y., "Extenal cognition: How do graphical representations work," *International Journal of Human-Computer Studies*, Vol. 45, No. 2, 1996, pp. 185–213. 4.4

[168] Schacherer, F., Choi, C., Gotze, U., Krull, M., Pistor, S., and Wingender, E., "The TRANSPATH signal transduction database: a knowledge base on signal transduction networks," *Bioinformatics*, Vol. 17, No. 11, 2001, pp. 1053–1057. 1.4.2

[169] Schreiber, F., "High quality visualization of biochemical pathways in biopath," *In Silico Biology*, Vol. 2, No. 6, 2002, pp. 59–73. 1.4.2, 6.1.2

[170] Schreiber, F., "Visual comparison of metabolic pathways," *Journal of Visual Languages and Computing*, Vol. 14, No. 4, 2003, pp. 327–340. 6.3.2, 6.12

[171] Sebrechts, M. M., Cugini, J. V., Laskowski, S. J., Vasilakis, J., and Miller, M. S., "Visualization of search results: a comparative evaluation of text, 2d, and 3d interfaces," *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, ACM Press, 1999, pp. 3–10. 5

[172] Shiozawa, H., ichi Okada, K., and Matsushita, Y., "3d interactive visualization for inter-cell dependencies of spreadsheets," *Proceedings of the 1999 IEEE Symposium on Information Visualization*, IEEE Computer Society, 1999, p. 79. 4.1

[173] Shneiderman, B., "Tree visualization with treemaps: A 2-D space filling approach," *ACM Transactions on Graphics*, Vol. 11, No. 1, 1992, pp. 92–99. 1.4.1

[174] Sirava, M. ., Schäfer, T., Eiglsperger, T., Kaufmann, M., Kohlbacker, O., Bornberg-Bauer, E., and Lenhof, H. P., "Biominer – modeling, analyzing and visualizing biochemical pathways and networks," *Bioinformatics*, Vol. 18, No. Suppl. 2, 2002, pp. S219–S230. 6.1.2, 6.3.2

[175] SmartMoney, "Market map," http://www.smartmoney.com/marketmap/. 1.4.1

[176] Soukup, T., *Visual Data Mining: Techniques and Tools for Data Visualization and Mining*, John Wiley and Sons Inc, 2002. 1.1

[177] Spence, R., *Information Visualization*, ACM Press, 2000. 1

[178] Spencer, R., "The streamlined cognitive walkthrough method," *Symposium on Computer Human Interaction (CHI'2000)*, ACM, 2000, pp. 353–359. 1.5

[179] Stewart, C., Hart, D., Berry, D., Olson, G., Wernert, E., and Fischer, W., "Parallel implementation and performance of fastDNAml — a program for maximum likelihood phylogenetic inference," *Proceedings of SC2001*, ACM, 2001. 1.4.3, 7.1, 7.2.2

[180] Sugiyama, K. and Misue, K., "Graph drawing by the magnetic spring model," *Journal of Visual Languages and Computing*, Vol. 6, No. 3, 1995, pp. 217–231. 3.2.1

[181] Sutherland, I. E., "The ultimate display," *Proceedings of the IFIP congress*, 1965, pp. 506–508. 1.2

[182] Swan, R. C. and Allan, J., "Aspect windows, 3-d visualizations, and indirect comparisons of information retrieval systems," *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, ACM Press, 1998, pp. 173–181. 4

[183] Tavanti, M. and Lind, M., "2d vs 3d, implications on spatial memory," *Proceedings of the IEEE Symposium on Information Visualization 2001 (INFOVIS'01)*, IEEE Computer Society, 2001, pp. 139–146. 1.3

[184] Tegarden, D., "Business information visualization," *Communications of the Association of Information Systems*, Vol. 1, No. 4, 1999. 1.4.1

[185] Tohsato, Y., Matsuda, H., and Hashimoto, A., "A multiple alignment algorithm for metabolic pathway analysis using enzyme hierarchy," *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB '00)*, 2000, pp. 376–383. 6.3, 6.3.1

[186] Tufte, E. R., *Envisioning Information*, Graphics Press, 1990. 2

[187] Tufte, E. R., *Visual Explanations*, Graphics Press, 1997. 1.1

[188] Tufte, E. R., *The visual display of quantitative information*, Graphics Press, 2001. 1, 1.3, 2.3

[189] Tutte, W. T., "How to draw a graph," *Proceedings of the London Math Society*, Vol. 13, No. 3, 1963, pp. 743–768. 3.2.1, 7.5

[190] Varshney, A. and Kaufman, A., "Finesse: A financial information spreadsheet," *Proceedings of the IEEE Information Visualization Symposium*, Vol. 125, IEEE, 1996, pp. 70–72. 1.4.1

[191] Wainer, H., *Visual Revelations*, Lawrence Erlbaum Associates, 1997. 1, 1.2(b), 1.4.1

[192] Walshaw, C., "A multilevel algorithm for force-directed graph drawing," *Proceedings of the 8th International Symposium on Graph Drawing (GD2000)*, Vol. 1984, Springer, 2000, pp. 171–182. 3.2.1

[193] Ware, C., *Information Visualization: Perception for Design*, Morgan Kaufmann, 1999. 2.3, 2.5

[194] Ware, C., "Designing with a $2\frac{1}{2}$d attitude," *Information Design Journal*, Vol. 10, No. 3, 2001, pp. 258–265. 2.1

[195] Ware, C. and Franck, G., "Viewing a graph in a virtual reality display is three times as good as a 2-D diagram," *IEEE Conference on Visual Languages*, 1994, pp. 182–183. 5.4.2, 6.5

[196] Ware, C. and Franck, G., "Evaluating stereo and motion cues for visualizing information nets in three dimensions," *ACM Transactions on Graphics*, Vol. 15, No. 2, 1996, pp. 121–140. 3.2

[197] Ware, C., Purchase, H., Colpoys, L., and McGill, M., "Cognitive measurements of graph aesthetics," *Information Visualization*, Vol. 1, No. 2, 2002, pp. 103–110. 3.2

[198] Wegenkittl, R., Löffelmann, H., and Gröller, E., "Visualizing the behavior of higher dimensional dynamical systems," *Proceedings of the 8th conference on Visualization '97*, IEEE Computer Society Press, 1997, pp. 119–126. 2.2.3

[199] Wharton, C., Rieman, J., Lewis, C., and Polson, P., *Usability Inspection Methods*, chap. The Cognitive Walkthrough Method: A Practioner's Guide, John Wiley and Sons, Inc, 1994. 1.5

[200] Wittig, U. and Beuckelaer, A. D., "Analysis and comparison of metabolic pathway databases," *Briefings in Bioinformatic*, Vol. 2, No. 2, 2001, pp. 126–142. 6.1.3

[201] Wohlers, T., "2.5D systems," `http://rapid.lpt.fi/archives/rp-ml-1996/1044.html`, 1996. 2.1, 4

[202] Wolf, D., Gray, C. P., and de Saizieu, A., "Visualising gene expression in its metabolic context," *Briefings in Bioinformatics*, Vol. 1, No. 3, 2000, pp. 297–304. 6.2

[203] Wright, W., "Research report: information animation applications in the capital markets," *Proceedings of the 1995 IEEE Symposium on Information Visualization*, IEEE Computer Society, 1995, p. 19. 1.4.1

[204] Zhou, Y. and Liu, J., "AVA: visual analysis of gene expression microarray data," *bioinformatics*, Vol. 19, No. 2, 2003, pp. 293–294. 6.2

format.

APPENDIX A

Sample Interview Transcriptions

The following are sample transcriptions of typical interviews. Three interviews have been chosen which capture feedback related to the three different case study domains. Source materials have been stored and are available upon request.

The left most column in each table identifies the speaker. "Int." is the interviewer, "A." is the answer or response. When more than one subject was present in the interview answers from each participant are labelled "A1.", "A2.", etc. The middle column is the transcription text. The last column shows notes made in the course of analysing the feedback.

A.0.1 Fund Manager Movement

The participant was a researcher in the field of fund manager performance analysis. He has been briefed on the "cognitive walkthrough" design of the study, i.e. that there will be a guided use of the tool to explore the data-set, and asked to comment on the visual paradigm. Some explanation about the systems by the interviewer has been omitted from the transcription. The interview lasted about one hour.

Spkr	Transcription	Notes
Int.	Each of these separate lines is a Fund Manager and the way it's arranged is that at each point in time which corresponds to the third dimension I've arranged the Fund Managers relative to the Index so that the closer a dot representing Fund Manager is to be indexed	*demonstrating "worm" fund manager movement visualisation*
A.	Where is the Index?	

Int. The Index is the blue dot in the middle of the 2D view, and in the 3D view if there wasn't such a cluster of Fund Managers around the Index, then you'd be able to see it as a blue bar in the middle. So the closer a Fund Manager is physically placed to the Index, then the more similarly weighted the portfolio is at that point in time. I won't explain the intricacies of how I've projected the high dimensional portfolio space down to the 2D location, but it's only important for you to understand that the more closely situated two points are in space, then the more closely weighted their portfolios are.
Each of these spikes is placed on a 2D plane relative to where a portfolio that is fully weighted in one market sector would be placed. So they act as markers for comparing the weightings of portfolios. So this yellow spike corresponds to the banking sector, so a portfolio 100% weighted in the banking sector would also be placed in this position. And the height of the spike corresponds to the bar on the left-hand list of market sectors which shows the percentage of market share of that market sector. So for the banking sector, 17% of all Fund Managers' holdings are in the banking sector. The lines representing Fund Managers wiggle which indicate that their weighting is changing over time, either due to trading or changing share value in each market sector. I can move the water level up and down which corresponds to the time period shown in the 2D cross section and you can see that our data-set runs from September, 2001 to August, 2002. We can select an area closer to the Index.

A. All these points are return, right?

return relative to the index turns out to be an important requirement

Int. No, it's the total weighting of the portfolio so a portfolio that's worth a billion pounds will be positioned in the same place as a portfolio that's worth a hundred thousand pounds if they have the same balance across market sectors.

A. Oh I see.

Int. So here I'm sweeping out an area closer to the Index to eliminate all the outlying portfolios, so that we can get a closer look at the portfolios more closely weighted to the Index. That brings up a new window showing a smaller number of Fund Manager worms and we can zoom in to see more detailed movement. You can see that the Index is now more clearly visible as a blue column in the centre.

A. What's to the left, and what's to the right and what are the axes?

typical response — it takes some time for people used to scatter plots or charts directly mapping the relationship between two variables, to understand the concept of dimensional scaling. Where only the relative distance between points is important.

Int.	There actually isn't an axis. Remember that portfolios are positioned in 2D purely based on their weightings across market sectors. This set of portfolios appears to be spread out more along one axis, that's because there is one dimension which captures much more movement than the other dimension. That is producing most of the variance in the data. I'd say the dimension is probably one of these market sectors here, or a combination of the market sectors represented by the spikes lying along this axis. So if we bring up a more conventional 3D chart view of one of the portfolios lying along the axis, then we can see the change in weightings in more detail.	
A.	So the previous graph that you showed me shows the weighting at a point in time.	*referring to the 3D area chart view.*
Int.	No, again it's across the entire time period. So each of the colour areas is one market sector and time is the depth axis.	
A.	So for each depth position in the graph, there is one corresponding position in the plane shown in the 3D window?	
Int.	Yes, exactly. So the chief components of the variance shown here will be these market sectors along here. Is all that making sense?	
A.	I still don't get .... You have the FTSE in the middle ... so what does it mean for this fund (pointing) relative to that fund? What's the left and the right?	*still struggling to understand dimensional scaling*
Int.	All it means is that they are spaced relative to difference between the portfolios. By distance I mean that we actually calculate a measure of difference between the weightings of the portfolios and the Index, so the portfolio you pointed to first is weighted differently to the Index, but the portfolio you pointed to on the right is also weighted differently to the Index, but more differently to the first portfolio. So if we bring up the chart view of the two portfolios that you pointed to .... we can see that they are indeed weighted differently in a number of market sectors. So more than that, you can see that in the 3D view as time goes on, the two portfolios come closer together, and looking at the chart views we can see that this is reflected in their relative weightings. They become more similarly weighted. So does that make any sense now?	
A.	Yep.	
Int.	So we have explored a data set a little bit. If you were going to explore the data set further and you had this tool available to you, what would you be interested in looking at next?	*attempting to guide a cognitive walk-through*
A.	OK firstly, can we go back to the view of the entire data set that you had initially.	
Int.	So, what do you notice about this picture? What strikes you as worthy of further investigation?	
A.	So looking at this, the first thing I'd like to know is what are the outliers? And I'd like to know what is the return for each of them, what is their profit relative to the Index? Because then you can tell which one is performing better, and which ones are performing worse.	*return*
Int.	So the way I am showing return here is through colour, and the thickness of the column. So the total value of each portfolio is represented by the thickness of the column, and increase or decrease in value is shown by colour, so green means that the portfolio has increased in value (made a profit) , and red means that the portfolio has decreased in value (made a loss) and black means staying about the same.	*should probably show profit or loss relative to the index, i.e. the analysts definition of return for a passive fund*

A.	Right well I'd like to be able to drill down and see the actual value of the portfolio, and not just that but what you really want to know is which one did best and which one did worst, and why. So who is the best performer in the group? So the colour tells me that a little bit, but I can't tell which one is best.	*another recurring theme is the need for more flexible interactivity*
Int.	So I guess you need to be able to filter it in some way.	
A.	Yes maybe by changing colouring to show different attributes. Maybe by marking the best performer in a colour that stands out, like yellow. And you want to know at what time it was the best performer, so for example if they make 100% return in the first quarter, but then make a loss for the rest of the period, that's information that you need to know. And you want to know the time series for their return. I can't look at all of these and know specific details about them. I need somewhere to get more detail.	*need for interactive browsing, flexible data query / filtering / mapping of visual attributes to data attributes*
Int.	OK, so let's look at it in a more detailed section. Let's go back to a section closer to the Index because the quality of this data set means that many of the outliers are probably due to incomplete data.	*a further detailed explanation of sources of errors in the data follows but has been omitted from transcription*
A.	Yes, half the time Fund Managers themselves don't know what they're holding.	
Int.	Yep, that's a bit scary. Now that we've zoomed into the portfolios more closely weighted to the Index, we have eliminated a lot of the smaller outlying funds, and one would hope that the quality of the data should on average be better. So we can see the variance in more detail. Is there anything here, any patterns that you find interesting, that you'd like to know more about?	
A.	So obviously I'd like to know what that thick one standing on its own is doing. What kind of fund is it? Active or passive? Because the performances of funds that track the Index are measured as their deviation from the Index. So if this is an Index fund, then I'd say by definition the best performers are the ones most closely situated to the Index. But if these are active funds, then we're more interested in their return relative to the Index.	*return / interactivity*
Int.	Unfortunately, our data set doesn't give us that kind of information. It doesn't tell us what the goals of the various funds are. That's part of the purpose of this tool, to help us discover what fund managers are trying to do. The way it is oriented around the Index, I guess it's more geared towards passive traders.	
A.	So there's a couple of other things I'd like to know. Can you tell me which one is the best performer in terms of return?	*interactivity*
Int.	Probably not easily from this view by looking at the colour, but we could easily add some logic to the programme to determine return by actually calculating from the data.	
A.	So if you click on that guy, can you tell me his holdings? I'd like to know what that kink in the line means.	
Int.	Yes we can bring up the chart view again to get a closer look. (brings up chart view) It looks like the kink is due to incompleteness of the data. The data is missing for these market sectors before about March, 2002.	

A. That's good because I would want to know that too. It's important to know where the problems in the data are. So looking at this portfolio, it looks like banks make up the largest part of the holding. I'd like to know now, are all the portfolios that we're examining that hold banks performing in the same way? And I'd like to know which banking stocks they're holding. I'd like to be able to drill down to this level of detail just to show how actual stocks are moving. I'd like in your chart view to be able to click on one of the market sectors and bring up a desegregated view showing the movement of portfolios just within the banking sector, in terms of their component stocks.

interactivity / flexible data-filtering/mapping

Int. OK, yes that's a good idea. The other thing that I'm trying to do in this study though is to evaluate the actual paradigm, the style of visualisation, visualising a spread of high dimensional data in a 3D visualisation. So looking at this visualisation, how does it compare to the visualisation or charting tools that you'd usually use to do this type of analysis?

A. I'm coming from a research approach myself, so usually our research is driven by an idea or a hunch that we form based on experience. And then we use data base queries and visualisation of charts and so on to test the theories. So it seems like this tool gives us a new approach in terms of visualising the entire data set naively in order to search for anomalies that we may not have expected.

It would be nice to be able to study the Index addition collision, that is, the stocks that are in the Index change all the time. And so one of the things that we're interested in is when a stock is removed from the Index or added to it. What strategy do the fund managers use to try and match the weighting in the Index. So the Index changes occur suddenly and with significant effect on the weighting which is impossible for traders to match exactly, and they may have to take a loss in order to match the Index. Normally we'd study time series charts of an individual manager's holdings in one stock or sector at a time. I'm not sure how that sort of behaviour would look in this type of visualisation.

Int. So what I'm trying to do with this visualisation is to condense all the movements from a number of portfolios and a large number of stocks or market sectors into a single display. So for example, if we look at this particular group here, you can see that at the end of the time period they all seem to move towards the Index at the same time. Is seeing that kind of behaviour interesting to you?

A. Yes, now that you point it out, it would be interesting to know if there is actually a pattern there that they all move towards the Index. The first question that I come up with is, I want to know when. Because there might be some issue that might induce that. It could be at the end of a quarter for their reporting they want to get as close as possible to the Index in order to make their performance look better. That's the first question, but if that doesn't answer it, I don't know what it could be, but it would be interesting to find out. It might be a topic for further research. We'd probably need to break it down into what they sell, what they underweight and what they overweight.

So I assume you'll be able to tell me that to some degree from this tool, so can we find out what sector they have changed their weighting in, by clicking on one and bringing up the chart?.

Int. It looks like there's a significant change in media and photography.

Operator brings up chart for one fund manager

A.	OK, I don't know how the industries are broken down, but it could be that there's something drastic happened to the value of one stock. We'd have to break it down further to find out in detail.	*interactivity*
Int.	Would I be right in saying that for you the most important thing in this kind of visualisation is flexibility in terms of being able to break down the data in any number of different ways in order to answer different questions that arise in your analysis?	
A.	Yes, I normally approach this kind of analysis top down. Is there a way that you can zoom down into industry sectors to get a detailed view by stock?	
Int.	Yes, it's just a matter of changing the way the data is aggregated, but unfortunately it requires a bit of programming, so I can't do it right now.	
A.	It looks like one or two of the spikes representing market sectors are much further away from the centre than others. Why would that be?	
Int.	It's probably that those sectors account for more of the variance than others, remember that their position relates to the position where a portfolio that was completely weighted in one of those sectors would be placed.	
A.	But how does that reflect the changes in market share of the sector over time?	
Int.	Actually that's a good point. We could probably show them in a similar way to the portfolios themselves, as 'wiggly' worms changing position and thickness throughout the entire time period, rather than just an aggregate, single position and height as they are now.	
A.	Yes, because changes in stocks in particular industries may account for a large amount of movement across a number of portfolios, for example, if something happens to NAB in the banking sector, for example they go bankrupt or merge, then that will affect everyone who is holding NAB.	
Int.	Yes, that's a really good point and also the way I'm using height of the spikes to show average market share of the sectors over the entire period is overloading the third dimension which breaks the metaphor a bit and is probably confusing.	*alternative mapping for sector markers*
A.	Yes, it's probably better to show the sector value using thickness the same way you did for portfolios. You have two issues here, one is the cross sectional position, the other is the time series and being able to integrate them is quite powerful.	
Int.	There's another visualisation tool that I'd like to show you which shows a more detailed view of the movement within an individual portfolio. This captures similar detail to the 3D chart view of one portfolio that we showed earlier, but presents it in a different way.	*Shows the column view visualisation tool. More detailed explanation follows but is omitted from transcription*
A.	Looking at that, what I want to know is, I can see that there seems to be some sort of correlation between the average stock price in each market sector and the movement of the fund manager between them. [Pointing to individual sector columns in the visualisation]. To me there's enough evidence there to show that there's some sort of story. And the next thing I want to know is, does it apply to all the funds? It's good if there's something there in this tool that suggests what I need to look for. The next thing is, as a researcher, I need to ask if there's something there, does it apply to all the funds, and if yes, then I need to know what events does it relate it, what stocks, what industry, is it because of insider trading or predictability, there's a lot of issues that explain why. That's good, I like that.	

Int.	I've observed this (movement from sectors decreasing in value to sectors increasing in value) in a number of portfolios. Unfortunately, because of the granularity of the data, it is impossible to tell whether the movement comes before the change or after the change.	
A.	Yes, that would be a very useful thing to know. So once we've observed this at a particular point in time for a particular portfolio, we probably need to go back and get more data somehow, but if you had really good quality data, then this would be an excellent tool for examining this type of movement. If you had daily data or even intra day data, then this would open up a whole lot of areas of study in terms of not only movement within the portfolio, but also market impact of large trades, so for example, if a trader makes really substantial trades, how does that affect the total value of the stock or the market or the sector? And we might even make some money from it! (Laughs)	
Int.	I've presented these visualisation tools to you mostly as a way of exploring a data set, but there's another use for visualisation tools which is for explaining data to somebody else when you've already studied it and they've learned something about it. You can present those ideas to somebody else using visualisation as a kind of reporting tool.	
A.	Yes it's possible that this might be useful for explaining my research to somebody. But to me it's more important to use this as a research tool to present to somebody. You can do anything. I like the idea that using this thing, you can summarise all your data set into a single picture and then I want to know why I'm seeing what I'm seeing, in other words I think it's a good way to stimulate research into areas you might not have otherwise expected.	

A.0.2 Metabolic Pathways Visualisation

Two biologists were present in this interview. Their responses are labelled "A." and "A2.". The biologists were not native English speakers and there were some problems with transcription due to their accents.

Spkr	Transcription	Notes
Int.	This is a set of 7 metabolic pathways for different organisms and I've stacked them into the third dimension so that you can easily compare differences between the different pathways. And the idea is that by stacking them in an order such that adjacent pathways are as similar to each other as possible, and that way it emphasises the differences, for example here it's fairly easy to see that this reaction occurs only in one organism.	
A.	To really see that you need multiple windows?	
Int.	Yes possibly, but you should be able to see it. It would be nice to have better labelling, but this is just a prototype. So what we want to evaluate today is more the general principles of the visualisation of this idea of stacking into the third dimension.. So that's one application. Visualising a set of related metabolic pathways. Another visualisation application is with just one metabolic pathway	
A.	Can you show the top view so that we can see it's just one pathway?	

Int.	Just one metabolic pathway but we've added some data from experiments, times series data. So the principle is fairly simple, we have times series data collected from several days in the course of our experiment, ten or 12 different samples. And we show this changing data in the third dimension. And then maybe before we start I can get you to describe what it is that you do. What your research into metabolic pathways is, and whether this style of visualisation would be appropriate.	
A.	We are dealing with carbohydrate metabolisms mainly. This contains of course different metabolisms, The interaction between different tissues. The problem with visualisation for us is that we have many, many data from different transgenic lines that we'd like to compare together. And we need to be able to see that very directly, and very easily. So something like that is very nice we can see that very nicely, but sometimes it's difficult to see what's going on.	*It becomes increasingly apparent that the biologist is concerned by the visual complexity induced by trying to visualise his large volume of data in this way*
Int.	The principle of this one is visualising the data in situ in the metabolic pathway. Is that useful for you?	
A.	Yes it is, one part of that. The other part is that we'd like to compare different plants together in one picture, different plants together and different compounds together within one pathway. That makes it a little bit complicated to show easily because we have 4 or 5 lines which we'd like to compare together, and maybe 10 metabolites and compounds which are changing.	*large volumes of data and a combination of both our applications, i.e. showing experimental data for a number of time periods across multiple species*
Int.	So there are 4 or 5 sets of data	
A.	5 lines means one wild type where each line is a plant, per plant you have 20 metabolites, and then you have these metabolites for 10-20 days or something like that.	
Int.	So you need to compare hundreds of metabolic pathways? What would be more useful, to place 5 plants side by side or to make an animation where you go from one to the next?	
A.	No all together. That is best if you go from one compound to the next you bore all the people. And it takes a long time to go through, but if you have it in one picture you can compare it and see the differences between wild type plant and transgenic lines directly.	
Int.	So you want to have one picture and the option to choose what ...	
A.	What is changing? What is changing, what is comparable? What is really happening in each pathway? One easy example would be glycolysis, sugar goes to sugar substance, goes down to some phosphor-related metabolites. And at the end we use them for expiration, for example, for amino acids. Now we have wild type plants and for example 3 transgenic lines with any number - and we have some changes in this pathway, for eg. modifying an enzyme here and we should see some changes here, and the question is now, how can we make it easier to understand that they are altogether here, these pathways, wild type, line 1, line 2, line 3 within these pathways?	(Draws diagram of a number of time series charts)
Int.	So you'd be interested in being able to compare just 4 pathways at a time?	
A.	Yes, the next step would be to see this one and this one and compare the total change of all the pathway metabolites within all the plants, all changes of the metabolites which occur.	
Int.	That's for one day?	

A.	For one day, that would be the easiest way. We normally do that, like when we compare them together we did it up to now, we just compare one metabolite like that - which are for example, changes in sugars, but in that case we have many, many graphs and it's likely to be complicated.	Drawing series of charts
Int.	Are there likely to be links, for eg. changes in one metabolite affect changes in another metabolite?	
A.	Within one pathway, yes. Because when we change here something, sugar may be accommodate it	
Int.	Would it maybe help if we draw the pathway but each of the objects in this pathway is a kind of histogram. Just show the graphical changes within the pathway. Effectively, that is what we have here (points at screen). The other advantage of this three D paradigm is that if there are changes in the quantity of these reaction enzymes, then you can show that with the thickness of the arrow.	Draws pathway with histograms at each node
A.	Yes. OK I see. You could do this in a 2D picture, but then you would have many charts, and the picture becomes confusing, the problem with the 3D picture is that we have no experience with this kind of visualisation, and so we cannot see very easily or fast the changes. You can see that, but not us, so here if you have different kinds of arrows, you do not know what that means, and we cannot see the changes and differences and so on.	
Int.	If we make it a bit bigger, then it's a bit easier to see the changes.	
A.	Yes, that's much nicer, and if you move up and down you can see really the changes	(moving cross section viewer)
Int.	The other idea that we're thinking about trying is that if you have another dimension of data, so if you have different lines and from each line you have a set of experimental data, so that each line might give you a picture like this, and you had a method for stepping through the different lines and replacing the data so that this 3D picture would change, as you chose which line and then you get another dimension of data.	
A.	Yes. In that case changing from line to line is not really so interesting, we just like to compare a control with all transgenic lines. So that the changes that happen in one line we can see happening in all transgenic lines compared to the wild type. The reason why we might want to have them altogether is that we can see that fast. That is easiest, but it gets more complicated when we want to take more lines on different days, but that would be very nice to have the first part finished and then think about the next set. And then the last part would be to join these pathways with the other and compare them together so that any changes in that pathway have any effects on the other pathways. That kind of picture would be good to have them altogether, that would be very useful and if you can change the type of diagram here or you can select another set of data here. I think this has some advantages. The disadvantage is that if you are not used to working in 3D, the advantage is that it's a kind of animation. It shows for you in a better way than if you just had a histogram. And then it's a bit like an animation going up and down. With only the first one it is very difficult, but with this one it's very clear.	
A2	But there must be some kind of focus, it's nice to see the overview, but if you go into 3D, there should be more, I think.	
Int.	More?	

A2	More data. Or something, I don't know. The advantage of 3D is that you can go away and see the whole, when you go very near you see only the box. This navigation and moving around must be somehow used.	
Int.	I think also the problem here is that we have a very restricted interface, in an ideal environment, for eg. in Sydney we have a 3D wall with stereo classes, and you have the impression that you are on a holodeck, like Startrek, or in virtual reality. On this small screen it's a bit difficult to use and get the illusion of 3D.	
A2	Yes it must be a little bit interactive I think. If you go to something and you want more info about this, that should be somehow possible. Only a static 3D image is perhaps not enough to get this. It would be nice if you could click on one of these columns and get a detailed view so that you can easily see how it changes over time. That would an idea. If you just go to the pathway and know exactly where the changes are, then you can click on it and you have the changes.	
Int.	We can also set it up so that the colour of the column at each point is dependent on the changes in the column.	
A.	Yes, the idea would be so that where there are changes you know them immediately, and then to get detail you pop up this window and see exactly - ah - here in the middle, maybe the name of the metabolite appears or something like that. And then you know exactly where the change is in the transgenic lines compared to the wild type. That would be very nice. This should be maybe coloured differently.	
Int.	Another idea that we've been playing with is using this facing so that the idea here is that these are stacked in an order according to a distance measure between the pathways, and so it would be possible herer to change the spacing based on how similar they are, so that the really similar pathways are close together, or you could change the spacing according to any other difference measure. Would that be useful to you in any way?	
A.	That would be good for one line if I understand it right.	
Int.	No we just space the levels to show difference or any parameter at all, so we can space them according to the time when the samples were taken. So if the samples weren't taken at even intervals,	
A.	So somehow like that, that would be then control and here transgenics or something like that, is that right? So 3 lines, 1, 2, 3?	draws picture of stacked pathways
Int.	Yes you could stack them that way. And you could change the height.	
A.	Yes, for eg. if all the sugars for transgenics are higher compared to here	
Int.	Well - this is the usual distance, and if there is a bigger gap, then there is a bigger interval between the data samples.	(pointing to gaps between pathways in diagram)
A.	Yes that's completely different to the other one. But this situation we don't encounter very often. Because what we normally do is a comparison between transgenic lines and wild type under different conditions. Although one of my colleagues is doing both, they are also doing experiments over many days and weeks and so on, on one line. So that could be used in this case?	
Int.	So you still often need to compare results from pathways? So another thing that I've been explaining to people is that in this example the ordering that we're showing is fixed, but there's no reason why you couldn't have an interface which actually selects which pathways are shown, and you could select half a dozen of them just by clicking on them.	
A.	Yeah, yeah, that's very good (getting excited).	

Int. So this is what I think you were saying before, we need to consider a lot of interactive features as well.

A. Yes, that's true. That would be then the next step really, that we have a list of different pathways and compare them, a small number at a time. That's very good. Although the problem with the animation is that we're not normally dealing with animation, though for talks or presentations it would be very good. If I do that in a seminar or a talk that's very good, but if I do it in our group, then they like to see the changes of all the metabolites. I like that very much myself, but I'm sure that the others are getting confused. One you know what's going on, that's very good. But it's just to explain it to those people, that's the challenge.

A2 So yes I think that's the next step to train people and implement different interaction, so that you can give them different ways to choose whatever they want to see in the context.

A. I think this is a very good idea. I like that very much, just to go through different pathways.

Int. At the moment we have to go through click, click, click, but it would be easy to make that automatic, like a kind of automation.

A. That would be very good.

A2 For me what would be very good is if we have this 2D net which you select from the 3D, and then if you click on one metabolite then you get a histogram or a table of the data.

A That is really a very good idea, if you have this pathway here you know glycolysis and then you click on it and you see the changes, and the changes can be marked by different colours for example. And then when you go through different pathways, get the changes for the other compounds, that's very nice.

Int. There's a theory that there's a limit to how much information you can show in one picture. Seven different entities.

A. That's not much.

Int. So it seems natural that there's going to have to be some way of zooming in on areas of interest.

A. So I have this idea that if you look at it, then the nodes such as black boxes are just black boxes, but then if you click on it you see more detail, so we can look at the overview and go with the mouse, so you don't see everything at one moment. If you look at very complicated graphics, then you have to look very carefully at what's going on.

Int. A way to look at small areas in the network is to be able to draw a box around it.

A. That would be perfect. That idea is very important because when you have large networks, when you change for eg. a specific enzyme here, then you expect to have changes only in this part normally. That would be the first thing we would have to look at, and we could just magnify that and see whether we have any changes, and then the second step would be to ask the question whether these changes have any effect on the other parts of the pathway.

A2 Another thing with the layering, it would be nice to pull the layers apart so that you can see between them. And the colouring between the layers, here it is the same, but it should be completely different so that you can easily contrast the layers.

A. To make it clear where the changes are.

Int. Yes, and the colours could be matched to this other view.

A. Yes, that would be perfect.

A2 If the colour has no other meaning, that could be also the case.

A.	Yes, we are coming very close to the solution.	
Int.	What time is it?	
A2	Lunchtime.	

A.0.3 Phylogenetic Tree Visualisation

Again two biologists were present.

Spkr	Transcription	Notes
Int.	So the idea is, this is just a prototype, we show one way to compare different organisms so we've got 7 different metabolic paths from 7 different pathways, and they've been stacked on top of each other in 3D. In order to compare the differences between them, you can see the top pathway is quite different in terms of the reactions that appear compared to the bottom pathways. For eg. if you look kat this reaction in the top pathway, the only other pathway that shares is the next one down, but then it appears in no other pathways. So we've been discussing this type of application with a number of biologists and getting some very useful feedback about the various different ways we can improve it. Another application shows just one metabolic pathway but with experimental data. For example, another biologist obtains a metabolite profile across 20 metabolites on different days, so you can see it's just one pathway, but then we've included the experimental time series data in 3D in the width of columns. Also the width of the arrows is an indication of the strength of the reaction, possibly how much enzyme is used or involved in the reaction, that sort of thing. This is what I'm calling a 2 1/2 D network visualisation metaphor. So we have a set of networks and by stacking it this way we're extending it into the third dimension. A few months ago I found a paper in which they were comparing a set of philogenetic trees. When they generate the trees they visualised the output of their clustering algorithm which inferred the phylogenic trees from DNA sequence data, but of course that process takes a very long time, and it would output a phylogenic tree every hour or so, gradually converging on optimal solutions to the problem. So they visualised that by taking each of the phylogenic trees and stacking them in the third dimension, and the idea was that you could see how the algorithm progressed over time. This visualisation I'm showing you now is based on data from a biologist in Sydney.	
A.	This is just to understand it. This is the time development in a Markov chain or Monte Carlo simulation which gives you over a time period an approximation.	

Int. That's right, actually this data set is a bit different. The output of his algorithm is a set of hundreds of phylogenic trees and they're ranked in order of what the best approximation to the actual tree is. So here we show the top 10 or so estimates, and they've been stacked in 3D in order to compare them. According to the biologist it's still interesting to see what the second best, third best, etc. estimates are. because the biologist may be able to use his expert knowledge to understand trees and find out if the differences are actually interesting and give more insights into the data. So if I show it to you from the top down you can see it's the same tree except for slight differences in certain areas. In the original paper when I saw them first use with 2 1/2D stacking method, I noticed that if we look at it like this, quite often there would be a lot of movement, and it would be quite hard to compare adjacent trees because they could be arranged quite differently. So I've applied an algorithm to arrange the trees by switching the branches in the tree in order to make the leaves appear in an order which is as similar as possible from one tree to the next. And then we link up similar leaves with these coloured lines and you can see that once I've run the algorithm, it's arranged these leaves in such a way that there are a minimal number of crossings between the lines.

A. So the coloured lines connect same taxa.

Int. Yes, that's right. So this is just a rough demonstration. There are many problems with it, for eg. there are no verbals, so you can't see exactly what the species are. But it's just intended to give you an idea of what it is that I'm doing so that I can get your feedback. One thing I'd like to show you is a slightly different rendering of that set of data. So here we have the same set of phylogenic trees, but arranged with the leaves around the perimeter of a circle. I've seen renderings of phylogenic trees in two dimensions using a similar radial layout. That's quite common isn't it?

A. Yes, this is called a circular tree or something like that.

Int. So the idea of doing this in three dimensions is that you get this barrel structure, where on the outside you always have the leaves representing the species and that might make it a little bit easier to track movements in the species across trees. The other thing is that I've link the root of each tree with this spine which you can see here simply to try and show whether there are significant structural differences or changes in structure between adjacent trees. So here you can see there's quite a big kink in the spine, so to look closer you can probably see significant structural changes between these two trees. So these are the ideas that I've been playing around with, and now I just want to get as much feedback from you as possible (laughing). So first, if you wouldn't mind just telling me how it is that you use phylogenic trees and their representations or drawings in your work. And if you can see any parallels?

A. Normally it's just drawings. And you have just two dimensional, the possibility to show it two dimensionally. You sometimes have the possibility of adding colour to show an extra attribute, but normally it is just this. And the problem we often have is, if you calculate these trees you get different results, and then you have to show all of them side by side, so we have to go through every one and compare, and you can manually flip it so that these taxa are here and you can draw it in this way. And sometimes you have the possibility to add some lines here so that the reader can see much faster the connection. But I think this (the demonstration) is appealing. You analyse say 5 or 6 different genes and everyone gives you a little bit different picture of the relationship of the organisms, and when you have to show up everything, here, here, here (pointing to diagram). And it is a little bit complicated for the reader to get the idea. And the first time I saw this picture I thought this would be a fine possibility to show gene trees, and at the end to infer the species the phylogeny afterwards. Because you have to integrate over 5 or 6 different gene trees to arrive at the end of a conclusion which will tell you what it is, the phylogeny of the organisms. And I think this would be a really fine thing. If you have, just like this metabolism network at the very beginning, you can see all the possibilities, but at the end you have a backbone which sums up everything in this backbone connection. In this way you could visualise and you could track, here is a different pathway, and this is another possibility.

A2 I think we would need to pick on one species and see where it appears in the tree.

Int. So you'd like to see more interactive possibilities in the system?

A2 Just a descriptive feedback, so that if you click on one node it highlights the other links so that you can see where it appears on every tree. I think this would help in combining molecular morphological data.

Int. What does that mean?

A2 We have molecular data that makes some trees, and we try to reconstruct the evolution from DNA or whatever, and then people work on the actual plant and try to refer back to the evolutionary pathways. And they see that you have a simple flower at first and then a more compound flower derived. Then the tree is much more simple but it would be absolutely interesting to map the simple trees on the complex molecular trees and then define molecular branches that are supported by the outside shape of the plant. And that is the main interest at the moment, combining data.

Int. Do you have an example of a data set of one of these trees?

A There are two different approaches. You can calculate an additional tree like here, or trees from several different genes, and there is in every tree a little bit different outcome. This one is calculated from morphological data. This one has red flowers, this one has blue flowers and this one has yellow flowers. And what you can do is to calculate these are just three characteristics, and you have often not enough morphological characteristics to get a really good statistical support for the entire tree. And the idea at the moment is to calculate this phylogeny, the evolution of the organisms with molecular data because you have an infinite number possible data, and you get very robust trees from the molecular side and then you just plot on these phylogenies different morphological characteristics. You can use, if this is the real organisms phylogeny, then you can go on and look where are these blue flowers, and you maybe see they are occurring here, and here and this gives you an immediate indication that blue colour arises two times independently during the evolution of the flowers and then you can go back and say, Oh why, where is the specific link, why have these two a blue flower and you find maybe these are pollinated by these and these are able to see blue better than red, something like this. To get more information about this interaction between environments and morphological structures, we often use the molecular tree and just plot it on, and this would be a nice thing to see.

A2 But in terms of geography, seeing that sometimes some branches are only African or we do it with chemical compounds, it depends on what is useful to our group.

Int. So how do you do it at the moment? Do you get one of these programmes and produce one of these trees with it and colour it by hand?

A2 Yes.

Int. So when you draw them, the software doesn't give you the option to change the ordering of the leaves if you wanted to make sure that all the leaves corresponding to some geographical attribute, for eg. if you wanted to ensure that all the species which occurred in Australia were placed together

A2 There are programmes that allow us to do that interactively. You can click and it swaps the branches. In a programme called Mac Clade you can change the topology of the tree just by dragging a branch from one point to another.

Int. But it doesn't happen automatically. You can't look at the tree and then press a button and have it group the leaves according to some parameter.

A2 No. But it can do other things, like with this flower colouring. It calculates where is the yellow or blue. OK I can't integrate it.

Int. So if I understand it, say from looking at the root, say this ancestor here is the first one which has the colour characteristic?

A2 Yes. But we can also click on the tree. What are the features, what characteristics for this branch, and there could be 10 different characteristics supporting this group here.

Int. So it tells you this in terms of the underlying data?

A2 Yes

Int. But is it possible, like with this other one we showed you, to combine or compare trees?

A2 No, but this would be really great. But also sometimes we have data which looks like a network where you can't decide immediately if a particular character state is the right choice. It looks like this because there are different types, and we might have types in our data where we are not able to decide. Is it derived on this way, or is it derived from this way. So we get networks. But it happened also that you might also get hybrid species where you have two species that combine. So we often have the problem to visualise networks,

Int. So you are not able to do this?

A2 No, it is normally really hard. My solution is I calculate all the non-hybrid species and then just add my hybrids where they belong. And so this is one parent and this is another parent, and my hybrid is derived from this. And I think this would be a promising part to include when we have ambiguous data, and it is very much like this when we have different metabolic pathways and you could show the hybrids in another dimension, and to be able to show that there are bifurcations they join together. This is much more biological than to assume that at every point we have a bifurcation.

Int. So the length of these branches shows evolutionary time? So when you have these hybridisations, must the total length of the branch on either side be the same?

A2 It depends on how you compute it and it's not trivial. There are different rates in different groups. And if you want to trace back and find it this happened at a particular point of time then you have to use some really good algorithms to calculate, and what the problem is is how to really integrate hybridisation and time. I can visualise it relatively easily and say this is not much time ago, maybe a million years, or this is a million.1 years, but as soon as you add these reticulate structures and to integrate them somehow in a logical way.

Int. So I think we have two different applications. The one that we have in our example is where the biologist computes all of the possibilities, and then chooses the best to visualise, and the second one is where you have trees from different sources, different sequences or different enzymes or whatever, and you want to combine these trees to again give a visual comparison and find out which is the most likely one. And in the second one the stacking ordering is arbitrary.

A2 It would be nice to bring it in such a way that the order shows you the more similar results are near together, and the more distant solutions are further apart.

Int. That's in fact the same problem that we showed you at the beginning for the metabolic pathways because the pathways are ordered in a way that the most similar are close together in stack. From the computational point of view the difficulty is you have an exponential solution space of possibilities, but when there are a dozen or so trees that's not a problem. We could use for the trees a similar approach to find an appropriate order for the trees, and then the other problem with how to deal with this network-like structure.

A2 Yes I think if you have a good solution for the metabolic pathway then it should be easy to transfer it for tree combinations. This is basically the same question that you have to answer there. This one with the reticulate structures in the phylogenic trees, this is a major problem for a lot of people. There is no good idea for how to visualise it.

A1 For this metabolic pathway this looks great. It would great to show ambiguities or differences, and from the top this is the place where there are the most connection. For this problem of the reticulate structures there is no solution currently, and I think this would be really helpful.

Int. This is just a technical question. The file formats for the trees that you use are standard text format. What file format do you use for these sort of reticulate structures.

A2 Just Powerpoint diagrams.

A1 There is no way to see immediately from the data which one is the hybrid. There are programmes that will say you have some problems in your structures, they are just not bifurcations. They cannot handle reticulate structures in the data. We do this manually. This is shown like this (draws picture). There is a rectangle in the tree. It gives you an indication that it is not just a bifurcation, it is just not this branch first and then these too, it might be 3 or even 5. This symbol shows that they have some special relationship.

Int. So the trees are not always binary, but they can be branching in three ways or more?

A2 Yes this is an indication that there may be 4 characteristics and only one is intermediate. And then on the other side there may be only two so they unite again.

All [Discussion about details of diagram]

Int. So it's an interesting question. So there are two problems, first we have to draw these trees with their reticulate structures and then find an ordering for sets of trees.

A1 Sometimes we show these changes in the trees with overlays in overhead projector slides or Powerpoint animations to capture the different aspects in the data. But it's all done by hand. It would be nice to show something like this (the 3D tree stack) when you're giving a presentation. So the points in the trees where there are different internal structures are interesting from the biology, why are there these differences in the data?

Int. Another leaf ordering that I've seen referred to a seriation. Underlying this is the data set in some dimension analogy, and they derive the trees by clustering species which are similar in the underlying data space, and I've seen papers where they talk about trying to order the leaves of a tree so that adjacent leaves of the tree are as close as possible in the underlying data space.

A2 But then there are crossings within the tree?

Int. No, they do it such that the plenarity of the tree is maintained because if you draw it with an arbitrary ordering of the leaves, then it's possible that leaves that are children in separate sub-trees and therefore are quite different or far apart in the underlying data space can be placed next to each other in the tree but in the tree drawing. Do you use this sort of ordering ever?

A2 No, but we may use an ordering to show the geographical relationships.

Int. Do you do this by hand by manually switching leaves?

A2 Yes, the programme lets you switch neighbouring branches.

A1	In principle it's a problem. What would be the best order and you can just flick around the branches without changing the top topology of the tree? You have to get it in the right order to preserve similarities, or to be able to combine or compare similar trees. And this is really a point, there is no programme which will do this automatically. You have to get an output from your programme and then bring it into a graphics programme, and then manually turn and arrange everything so that you can easily show that these two groups, there are some differences but they are mostly the same. And if you don't do it, then you have a completely unordered set of trees and really no one can read it.	
All	[discussion about exchanging data for further collaboration]	
Int.	So how do you label the trees in your diagrams? Do you label only the taxa, or do you label the internal nodes as well?	
A1	No we only label the leaves because they represent the actual organisms and everything deep down here is hypothetical and we have no names for them.	
A2	To give you an example I have just mapped chemical compounds onto the leaves and on the internal branches of maps their common ancestors.	
Int.	So the big branches have the same chemical compounds as the children?	
A2	Yes, some common alkaloids. So that would be something just as an example. The ancestors here develop one compound and all the leaves in this sub-tree have the same compound to give an idea.	
A1	We use this kind of parsimony to explain the distribution of characteristics with as few as possible mutations. We don't want to invent too much hypothesis. So in the common ancestors we show the lowest possible number of changes.	

APPENDIX **B**

Included CD-ROM and Software Description

The included CD-ROM contains software implemented to test ideas presented in this thesis. All systems were implemented in Java[1] and Java3D[2]. The CD-ROM contains ZIP files which should be unpacked onto a local drive. The resultant directories contain README files which explain how to run the various programs. Note that these software systems should be run on computers with fast processors (Pentium 4 or better) at least 512 megabytes of main memory and graphics cards with at least 128 megabytes of memory.

B.1 The WILMASCOPE Three-Dimensional Graph Visualisation System

WILMASCOPE was designed as a generic stand-alone system for three-dimensional graph visualisation and as a toolkit to be used by other programs. See Figure B.1 for a screen-shot. WILMASCOPE features:

- a hierarchical-clustered graph model;
- plugins for different glyphs to represent graph elements;
- an extensible object-oriented design allowing easy extensibility for new layout algorithms or graphics environments;
- the ability to create high-resolution still images or to record animations;
- open-source distribution under the Lesser GNU Public License[3]

[1] Available from `http://java.sun.com/j2se/`.
[2] Available from `http://java.sun.com/products/java-media/3D/`.
[3] See `http://www.gnu.org/copyleft/lesser.html`.

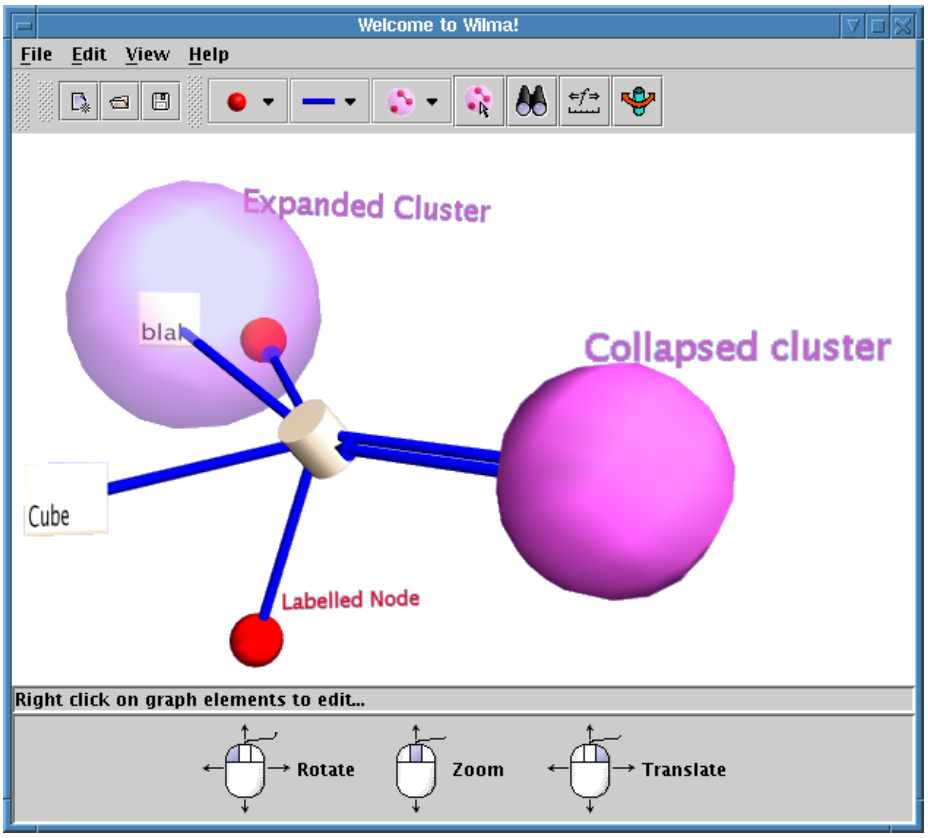


Figure B.1: A WILMASCOPE screen-shot showing some important features.

More details on WILMASCOPE are available from the homepage: `http://www.wilmascope.org` and is also discussed at greater length in [48].

The distribution on the CD-ROM comes with a number of examples described in this thesis. The following files can be found in the default data directory:

`sectorFMMove.xwg` — An example of a PORTFOLIO COLUMNS stratified-graph visualisation showing movement in a fund-manager's portfolio as described in Section 5.6.

`metabolicPathwayWithData.xwg` — A visualisation of experimental data shown *in-situ* in the underlying metabolic pathway as described in Section 6.2.3.

`metabolicPathway.xwg` — The metabolic pathway comparison example from Section 6.3.

`phylotree.xwg` — A stratified phylogenetic tree visualisation with minimal inter-strata leaf-edge crossings as described in Section 7.2.2.

`phylotreeCyl.xwg` — A stratified phylogenetic tree visualisation using the Barrel layout method as described in Section 7.5.

Use the `File->New` menu option to remove the current graph before loading a new example. The examples may be examined with a cross-sectional viewer by activating the `View->Show Axis Plane` menu option.

B.2 The PORTFOLIOSPACE EXPLORER System

The PORTFOLIOSCAPE EXPLORER system, as described in Chapter 5, is also included on the CD-ROM. Once it has been run (see the included README file) the user is presented with a window allowing them to choose the data-set that is loaded, see top-left window in Figure B.2. Choose either a data-set from the drop-down menu box and press the “Add new PCA” button to initiate the visualisation.

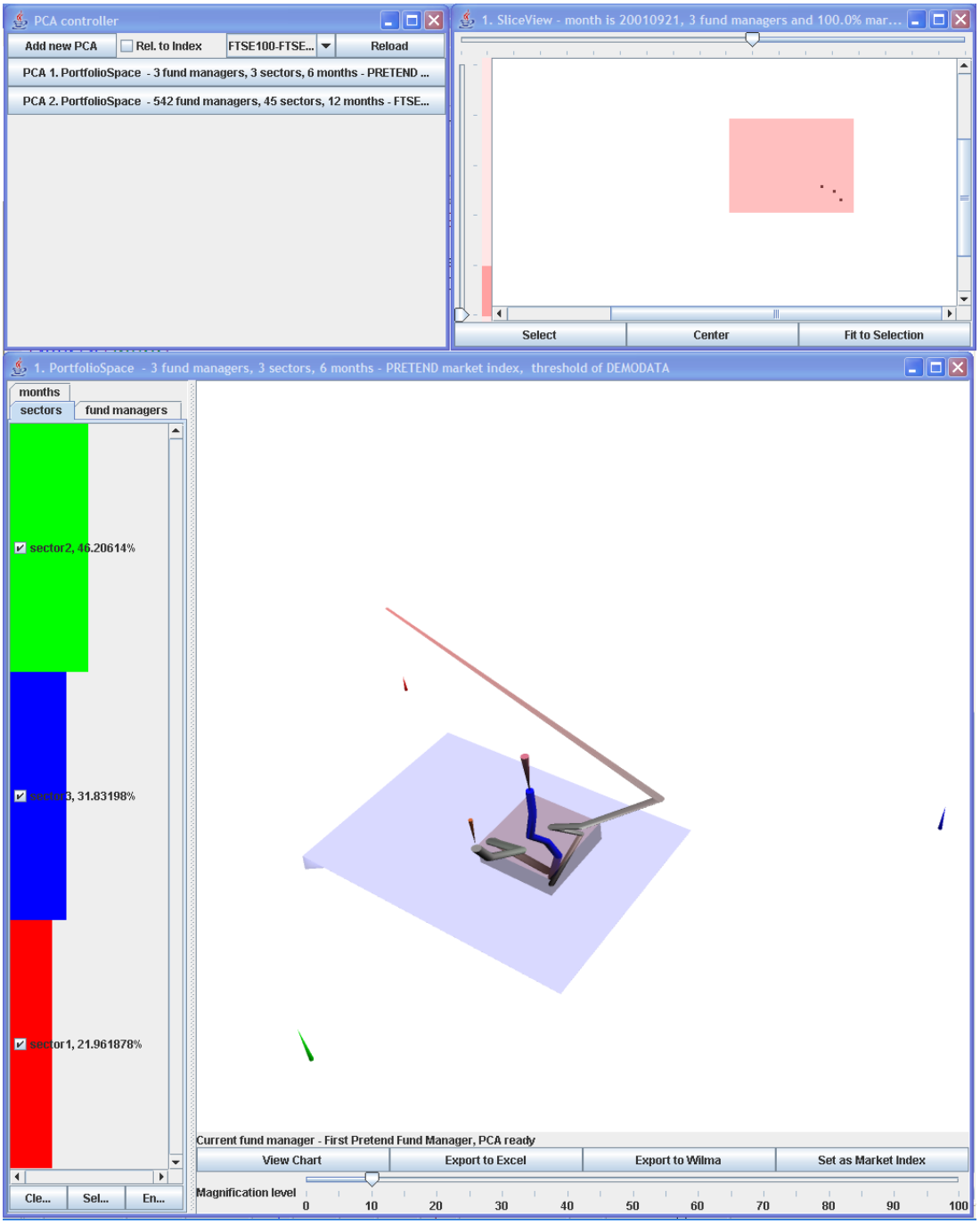


Figure B.2: The PORTFOLIOSPACE EXPLORER system.