IMPROVING AUTOMATED LAYOUT TECHNIQUES FOR THE PRODUCTION OF SCHEMATIC DIAGRAMS

A THESIS SUBMITTED TO
THE UNIVERSITY OF KENT
IN THE SUBJECT OF COMPUTER SCIENCE
FOR THE DEGREE
OF DOCTOR OF PHILOSOPHY.

By
Daniel Chivers
November 2014

Abstract

This thesis explores techniques for the automated production of schematic diagrams, in particular those in the style of metro maps. Metro map style schematics are used across the world, typically to depict public transport networks, and therefore benefit from an innate level of user familiarity not found with most other data visualisation styles. Currently, this style of schematic is used infrequently due to the difficulties involved with creating an effective layout – there are no software tools to aid with the positioning of nodes and other features, resulting in schematics being produced by hand at great expense of time and effort.

Automated schematic layout has been an active area of research for the past decade, and part of our work extends upon an effective current technique – multi-criteria hill climbing. We have implemented additional layout criteria and clustering techniques, as well as performance optimisations to improve the final results. Additionally, we ran a series of layouts whilst varying algorithm parameters in an attempt to identify patterns specific to map characteristics. This layout algorithm has been implemented into a custom-written piece of software running on the Android operating system. The software is targeted at tablet devices, using their touch-sensitive screens with a gesture recognition system to allow users to construct complex schematics using sequences of simple gestures.

Following on from this, we present our work on a modified force-directed layout method capable of producing fast, high-quality, angular schematic layouts. Our method produces superior results to the previous octilinear force-directed layout method, and is capable of producing results comparable to many of the much slower current approaches. Using our force-directed layout method

we then implemented a novel mental map preservation technique which aims to preserve node proximity relations during optimisation; we believe this approach provides a number of benefits over the the more common method of preserving absolute node positions. Finally, we performed a user study on our method to test the effect of varying levels of mental map preservation on diagram comprehension.

Acknowledgements

My main thanks go to my supervisor, Dr. Peter Rodgers, for his time over the duration of my Ph.D and whose advice, guidance and support have taught me invaluable research, presentation and writing skills; without which I would not have been able to complete this thesis.

I would also like to express my appreciation to my parents for their unwavering support and encouragement throughout my time in higher education; and to my brother Stefan, whose inspiration and guidance when I was younger helped change my perspective on education and spark a passion for knowledge. It was from following in his footsteps that I discovered the field of computer science which has lead me to where I am today.

To my close friend Kristy, for her resolute belief in my abilities – far exceeding my own, and who encouraged me to undertake a Ph.D in the first place; and to Dr. Mole, who went though the whole process simultaneously – thanks :)

I would also like to extend my gratitude to Max Roberts, Martin Nöllenburg, Helen Purchase, and everyone else who expressed particular interest in my work and provided valuable advice and feedback on various aspects. Finally, I would like to thank all members of the computing department, in particular my colleagues in SW104 "The Zoo", for providing an enjoyable and rewarding working environment during my time at the University of Kent.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis addresses the problem of automated metro map layout. Current published maps seen around the world are drawn by hand, requiring a great deal of time and skill to create. Research in automated layout techniques for creating this style of diagram has progressed over the past decade, but the uptake of these methods into schematic diagram creation has been slow. Possible reasons for this include the lack of availability of software for design and layout, and that the output quality of current automated techniques does not yet equal that of a well-designed hand drawn schematic. Our work in this field aims to address this problem by furthering the current research in automated schematic layout. We have created two pieces of software for supporting user's needs of an automated layout system, and developed a new layout method offering a number of benefits over existing techniques.

Current metro maps, that can be seen in cities all around the world (see Section 2.1.1 for world metro examples), share a number of characteristics which appear to originate mostly from a map of the London Underground drawn in 1931 by Harry Beck (Figure 2). Beck worked as a technical draftsman and produced the map taking inspiration from electrical schematics, in particular octilinear line angles (multiples of 45°) and roughly equidistant station spacing in place of geographical accuracy. These features were a radical change from both the existing map (Figure 1) and its predecessors, which aimed to preserve geographic positions of stations, and used curved lines to mimic the physical line

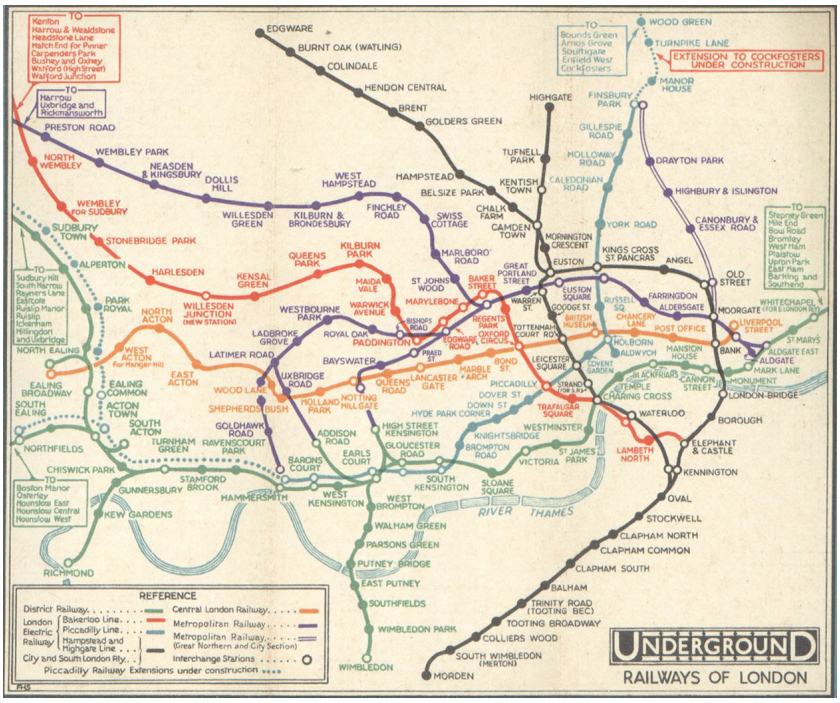


Figure 1: London Underground map, 1932 – the year before introduction of Harry Beck's design.

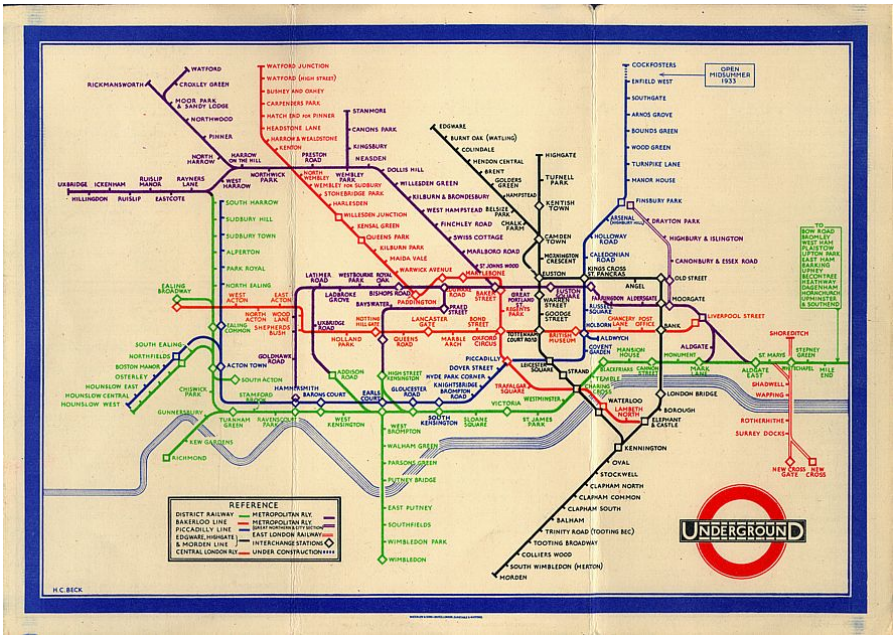


Figure 2: London Underground map, 1931 (design) – Harry Beck. Image shows pamphlet produced in 1933.

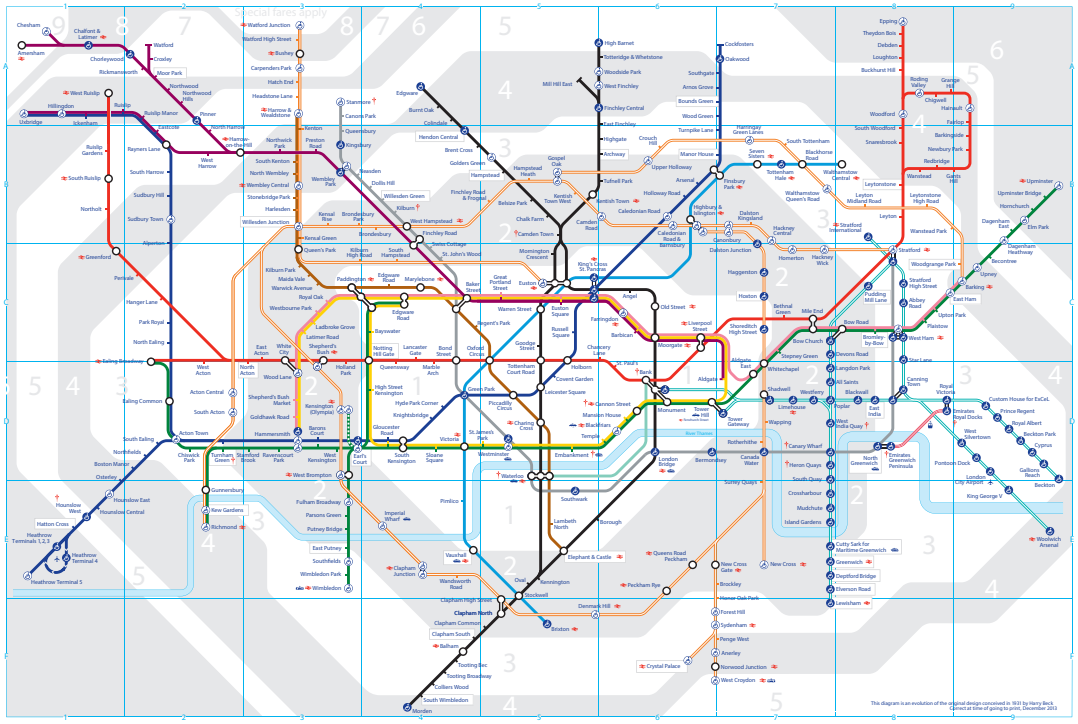


Figure 3: London Underground map, 2014 – Copyright Transport for London.

paths. These previous maps were also often seen drawn over a ground-level map, further emphasising their geographical accuracy. Beck believed that users of the underground system were not interested in geographical accuracy but rather how to get where they were going and where to change trains, and so he removed this characteristic in favour of only maintaining map topology – all nodes and connections are accurate, but geographical accuracy is not ensured. The move from geographic to topological accuracy afforded Beck the ability to move stations, thereby allowing straighter lines, approximately equidistant station spacing, and expansion of dense areas in order to improve the primary objective of a schematic – readability. The river Thames also features in Beck's design, albeit following an octilinear representation, to preserve some geographic reference for users. Beck's design was initially declined in 1931 by London Underground for being too radically different, and not indicating distances between stations; but after a successful trial run in 1932 the map was given its first full publication in 1933. The positive reaction from customers proved it was

a sound design, and a large reprint was required after only one month (Beck 2014).

Beck's map has been continually updated since its initial inception to accommodate new lines and additional station information, and the underlying design elements have been kept roughly in line with the original. The modern London Underground map (Figure 3) has become an iconic design and symbol of London itself, and its effective yet simple method of conveying complex information is mimicked in many other published metro maps.

1.1 Motivation

It is intuitive that the design of a schematic has an impact on its readability – a well-designed schematic will allow the user to interpret the presented information faster. In the case of a metro map this refers to tasks such as the speed at which a user can locate a station, or plan a route from station a to station b. There are no defined rules to ensure a metro map is easy to use, and as such most are drawn by graphical designers who are experienced in producing clean, easy to use diagrams using appropriate software for the task, typically a vector graphic design application. Even so, metro maps such as that of London contain hundreds of stations making the creation of diagrams very difficult; because of this it has become necessary to have not just good graphic design skills, but an understanding of the psychology of map reading in order to design an effective layout. Figure 4 shows the published Madrid metro map in 2014 and is a good example of bad layout decisions. It can be seen that the designer(s) of this diagram have used many characteristics also found in the London Underground map, such as octilinearity and even station distribution, but have chosen to retain a level of geographical accuracy. The result of this is that many line sections contain unnecessary bends, only serving to make the diagram harder to read. For example, the leftmost red periphery section bends downwards slightly, before bending back upwards twice as it extends outwards; a similar strange layout can be seen in the southernmost circular line. One might assume that these layout decisions are to maintain geographical accuracy, but as the entirety of

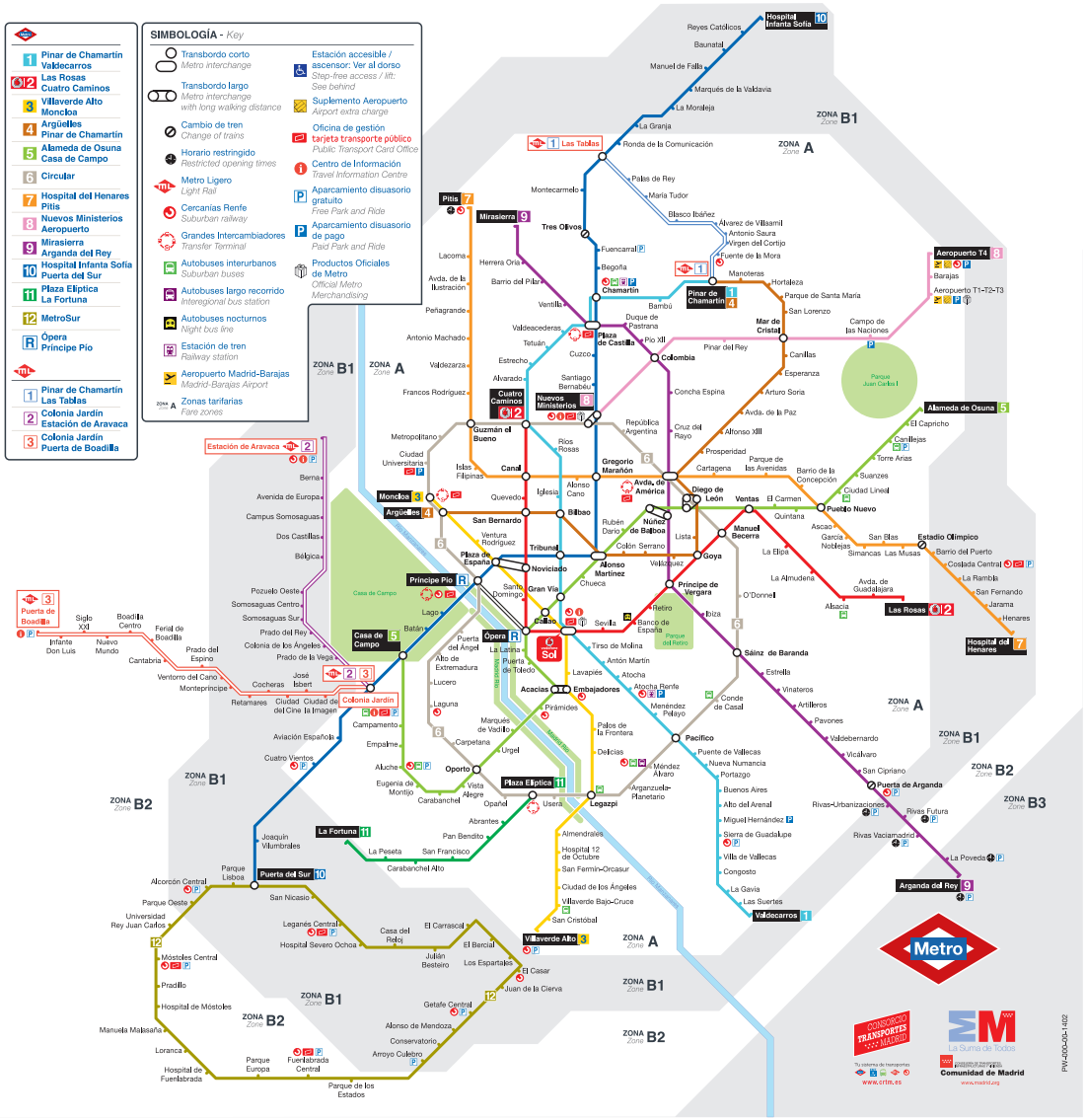


Figure 4: Madrid metro, 2014 – Copyright Metro de Madrid, S.A.

the schematic is not geographically accurate, it would be wrong to infer that these line sections are. This means that the line bends do not convey any additional information and serve no purpose – it would have been more suitable to use straight lines with minimal bends, and the same can be said of many other design choices in the schematic.

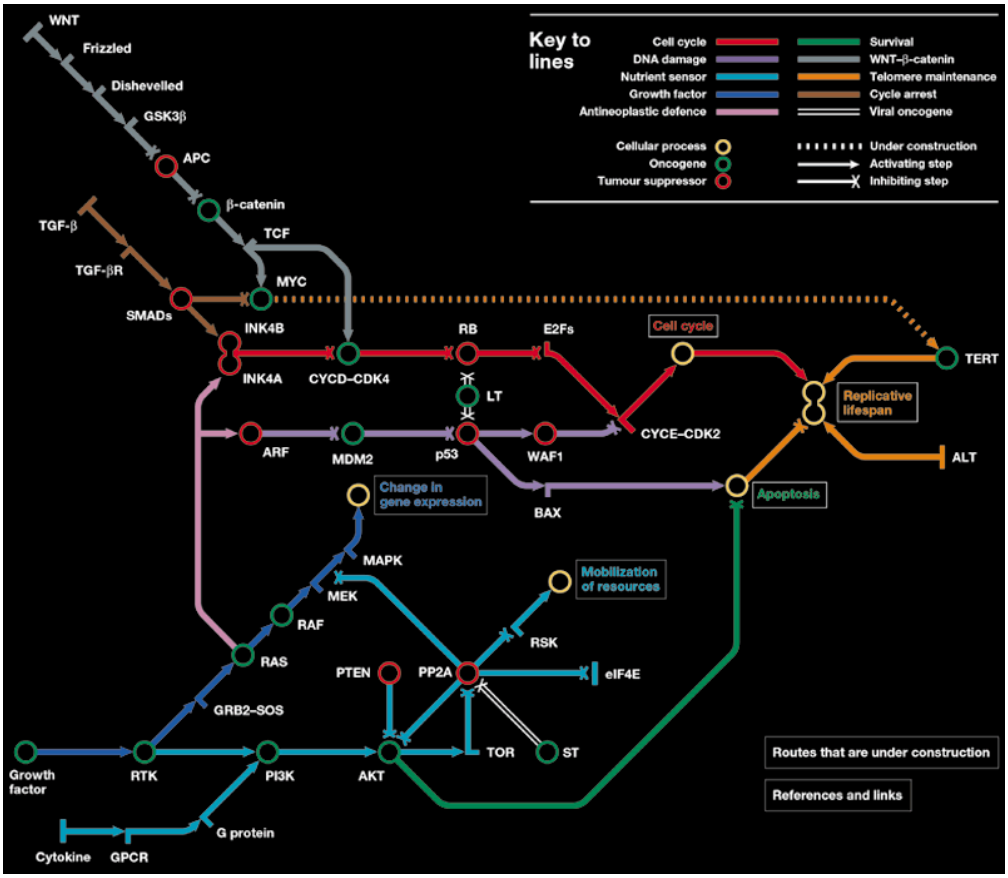


Figure 5: Metro map style cancer pathway visualisation by Hahn and Weinberg (designed by Claudia Bentley) (Hahn and Weinberg 2015).

As this visualisation technique is used for most published metro maps, it holds an advantage of public familiarity over other styles. For this reason it can also be used as a very powerful visualisation for many other types of data, benefitting from a higher base level of public comprehension. There are a wide variety of examples of such usage on the Internet, for example those shown in Figures 5 and 6; these figures show a metro map style layout of cancer pathways and musical artists respectively. These two figures, along with many other privately created maps (more examples can be found in Section 2.1.2), indicate that this visualisation technique has great potential beyond transportation systems and that there is a desire for it in many application areas. However, the difficulty currently involved with the creation of such maps unfortunately

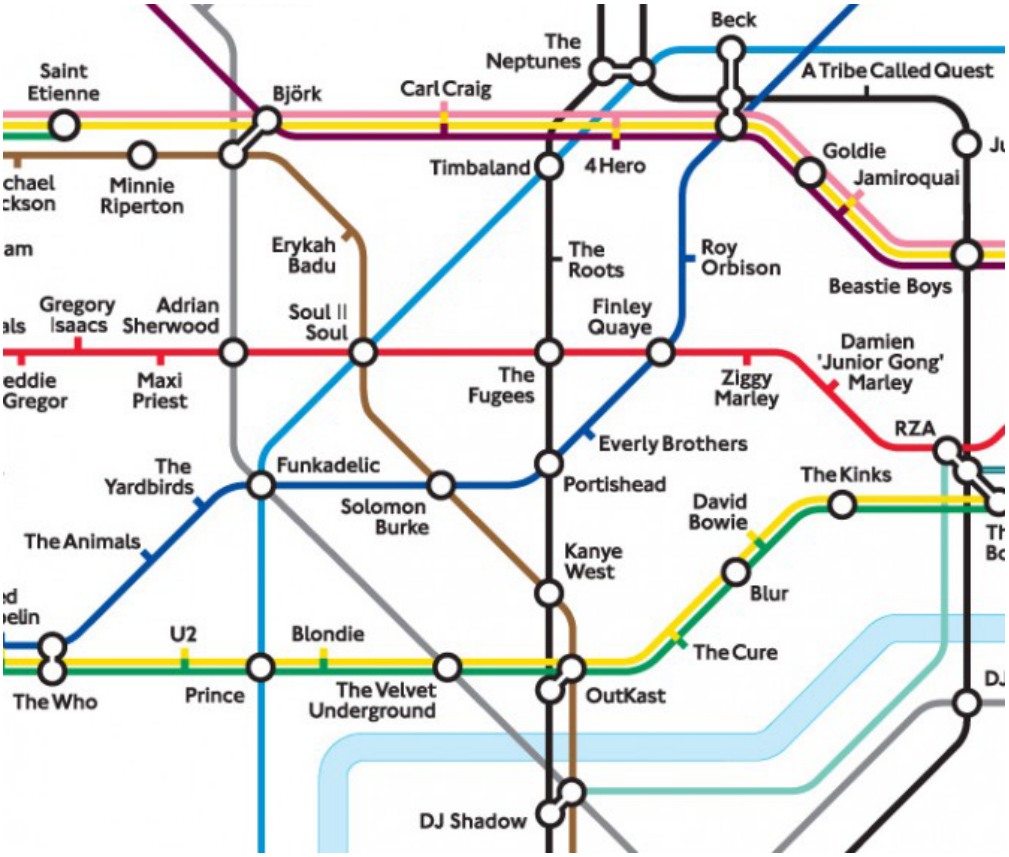


Figure 6: Metro map of music by Dorian Lynsky (Lynskey 2015) (clipping). Lines represent music genres, stations represent bands/artists.

puts a limit on the use of this visualisation style – an effective design is very time consuming and difficult to create. As a result of this, there are many examples of hard to read and/or poorly designed metro map schematics. An alternative approach to the creation of these diagrams is to take an existing schematic and simply modify the labels; this method was used to create the metro map of music (Figure 6) from the London Underground map – even the river Thames can still be seen. Although this method of creation is somewhat easier as it removes the responsibilities of node placement and line routing, the creator is subsequently restricted on label placement and must be flexible with their data to ensure the schematic works. This is not ideal for the vast majority of custom data sets.

Besides the creation of static diagrams, the recent popularity of smartphones

and mobile tablets offer new and exciting opportunities for automated layout algorithms. A fast layout method could allow real-time interaction with the user providing custom-generated maps – for example a transport map could be manipulated to emphasise a particular route, making it both easier and faster for users to navigate.

These points provide motivation for our work in automated metro map layout. We believe that a software tool combining effective drawing tools and an automated layout method would benefit many users – both professional designers for a fast initial layout that can be further improved, and amateur designers who may struggle with creating a good quality layout. Such a tool will hopefully lead to a wider adoption of this underused visualisation style.

1.2 Research Goals / Contributions

The work described throughout this thesis can be broken down into multiple smaller research goals, each of which is explained in this section. These goals introduce multiple new concepts and/or techniques for automated metro map layout, however most have not been empirically tested. The rationale for this approach is that in the current environment of automated metro map layout, there is still plenty of room for experimentation with novel layout techniques to discover methods that yield effective results and operational advantages. We believe that the field is not yet at a stage at which, although it may be interesting, detailed empirical comparisons between methods are necessary, as mostly a subjective visual comparison can provide compelling evidence of the effectiveness of a technique. For this reason we believed it more beneficial to explore a wider range of techniques with shorter subjective evaluations.

1. To develop a gesture-based input system for mobile devices to facilitate the creation and modification of schematics in a metro map style. We achieved this goal by producing an Android application which allows users to quickly draw schematics by using a series of gestures to create schematic objects. This section of work is described in Chapter 3 and published in (Chivers and Rodgers 2011).

2. To implement and improve a current multi-criteria optimisation method, as well as examine how modifying optimiser parameters affects the resulting fitness value. We achieved this goal by implementing a published multi-criteria optimiser into our Android application, and using a number of techniques to improve its performance. We then ran a large number of tests to attempt to determine a link between optimiser parameters and the final layout. This section of work is described in Chapters 4 and 5 and published in (Chivers and Rodgers 2013)(Chivers and Rodgers 2014a).

3. To develop a new force-directed octilinear layout method which would improve upon the existing implementation. Our method is outlined in Chapter 6, and example layouts are shown. Our results show that our new technique produces superior results than that of its predecessor. This section of work is explained in Chapter 6 and published in (Chivers and Rodgers 2014b).

4. To develop a mental map preservation technique based upon node proximity relations, as opposed to the standard absolute node position methods. We used a Delaunay triangulation to achieve this goal, and implemented the technique into our graphing software for combination with our force-directed layout method. This section of work is explained in Chapter 7 and published in (Chivers and Rodgers 2014b).

5. To carry out a user study to determine if mental map preservation has an effect on diagram comprehension after a modification. This study was to provide further valuable information to the current inconclusive research in this area. This section of work is outlined in Chapter 8 and published in (Chivers and Rodgers 2014b).

1.3 Statement of Publications

The following is a list of publications, including their related chapter(s) and goal(s):

- **Gesture-Based Input for Drawing Schematics on a Mobile Device** (Chivers and Rodgers 2011). Daniel Chivers and Peter Rodgers, In *Proceedings of the* 15^{th} *International Conference on Information Visualization (IV11)*, pages 127–134, July 2011. Goal 1; Chapter 3.

  ▷ This paper covers the implementation of our gesture-based schematic input software along with details of the implementation of a multi-criteria hill-climbing layout optimiser.

- **Exploring Local Optima in Schematic Layout** (Chivers and Rodgers 2013). Daniel Chivers and Peter Rodgers, In 19^{th} *International Workshop on Visual Languages and Computing (VLC), proceedings of Distributed Multimedia Systems (DMS 2013)*, 28 pages, August 2013. Goal 2; Chapter 5.

  ▷ This paper explores how varying optimiser parameters of a multi-criteria hill climber affects the resulting fitness.

  ▷ This paper received the DMS 2013 best paper award.

- **Improving Search-Based Schematic Layout by Parameter Manipulation** (Chivers and Rodgers 2014a). Daniel Chivers and Peter Rodgers, In *International Journal of Software Engineering and Knowledge Engineering*, in print, 2014. Goal 2; Chapter 5.

  ▷ This paper is an extended journal version of (Chivers and Rodgers 2013), which we were invited to submit after presentation at DMS 2013.

- **Octilinear Force-Directed Layout with Mental Map Preservation for Schematic Diagrams** (Chivers and Rodgers 2014b). Daniel Chivers and Peter Rodgers, In 8^{th} *International Conference on the Theory and Application of Diagrams (Diagrams 2014)*, pages 1–8, July 2014. Goals 3, 4, 5; Chapters 6, 7, 8.

  ▷ This paper covers our force-directed octilinear layout algorithm, our Delaunay triangulation mental map preservation, and our study into the effect mental map preservation has on diagram comprehension.

All of the above publications were peer reviewed.

1.4 Summary of Chapters

This thesis is divided into chapters as described here:

2. **Background**. This chapter provides a background in schematic layout and how schematics are designed, including details on previous and current automated layout methods.

3. **Improving The User Interface**. This chapter details a piece of software written to allow creation of metro map style schematics using a series of gestures.

4. **Multi-criteria Hill Climbing Optimiser**. This chapter covers the implementation of our multi-criteria hill climbing optimiser, explaining improvements that were made.

5. **Exploring the Effects of Parameter Manipulation**. This chapter covers our testing process that was performed to determine the effect of optimiser parameters on the final layout.

6. **Force-directed Octilinear Layout**. This chapter describes the steps taken in the development of our method to enforce octilinearity in force-directed layout.

7. **Force-directed Layout & Mental Map Preservation**. This chapter details how a Deluanay triangulation is used to constrain node movement during optimisation and thus help preserve the users' mental map.

8. **Mental Map Preservation Comprehension Study**. This chapter provides the procedure and results of our study into how mental map preservation in dynamic schematic layout affects comprehension.

9. **Conclusion & Future Work**. This chapter provides a conclusion to our work and covers a number of ideas for future research in this area.

1.5 Summary

Metro map design is currently inaccessible to a wide audience due to the difiiculty involved in both design and layout. This is due to a lack of programs designed to aid users in their creation by providing functionality such as automated layout and construction tools specific to the metro map style. Along with this, the current multi-criteria hill climbing technique for metro map layout leaves room for improvement in the form of additional new criteria and performance improvements. It is also a local optimum method and is very sensitive to its initial configuration, although there has been no research into any apparent trends dependent on this.

Search based techniques can produce good quality results but this comes at the cost of performance, making them unsuitable for interactive applications. The existing force-directed method, which does run fast enough to be suitable for interactive applications, does not produce results of a comparable quality and leaves plenty of room for improvement.

Finally, although current literature in the area of mental map preservation has shown no significant effect, the techniques used in its implementation have all been based on the absolute movement of nodes. A proximity-based implementation has not been tested, and may produce alternate results.

This thesis will address these specific problems and shortcomings.

Chapter 2

Background

This chapter will provide a background to the area of schematic layout, in particular that of metro maps. We examine a selection of current metro maps from around the world, explain current design processes used in their production, and discuss how there is vast potential for improvement. Besides official metro maps, we explain the metro map metaphor and why this visualisation style is not more frequently used. We then examine a number of techniques which have been developed for the automated layout of graphs, leading specifically into metro map style schematics.

2.1 Schematic Mapping

This section presents a selection of official metro maps from around the world. These maps are up-to-date as of July 2014, found within metro stations, and are used by the majority of customers. We examine the key characteristics of each and summarise how these maps share a number of common features employed to aid comprehension. Many more examples can be seen in Mark Ovenden's book, *Metro Maps of the World* (Ovenden 2005). We discuss the design processes in place to create such maps for official use, and identify the advantages and disadvantages of the current approach. Following this we explain the metro map metaphor and provide a number of examples to show alternate uses of the visualisation style; we then discuss why it is not more frequently seen.

2.1.1 Published Metro Maps

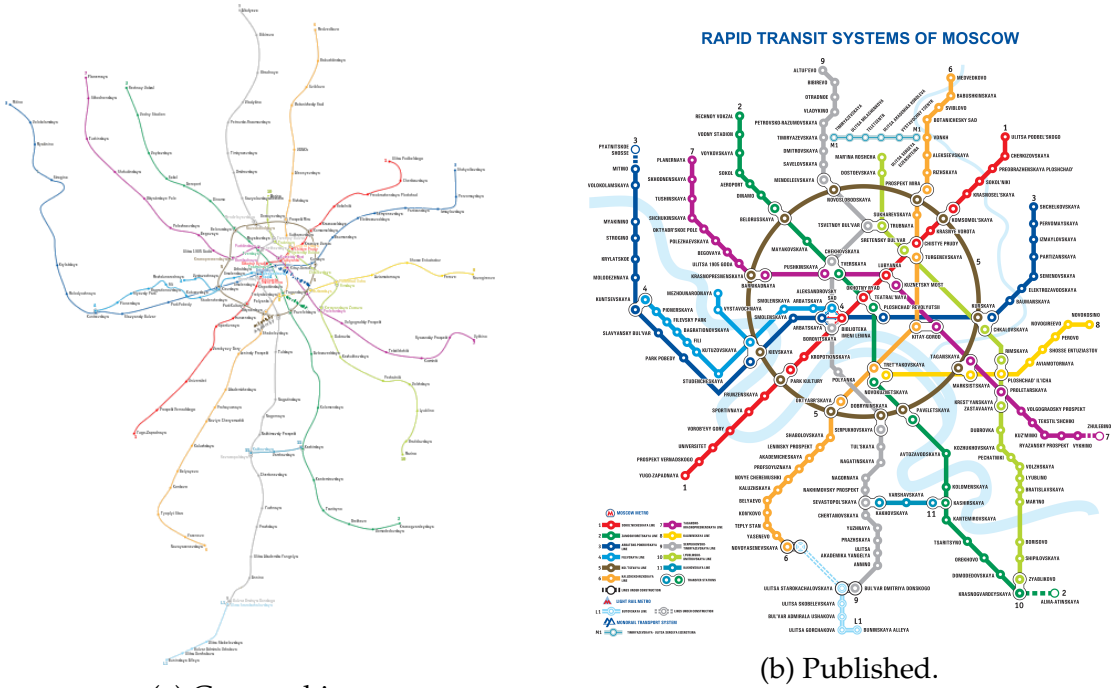
This section will focus on the following four examples: Moscow (Figure 7), Sydney (Figure 8), Vienna (Figure 9), and Washington (Figure 10); maps for London (Figure 3) and Madrid (Figure 4) have also been previously presented. These maps have been chosen as they represent good examples of typical metro maps, as well as include a number of unique and interesting features.

2.1.1.1 Features

When comparing the example schematics, many common features can be identified. An obvious but important feature of metro maps is the use of multiple line colours. This feature is found in all published metro maps and is very effective at differentiating between lines. Black and white versions of metro maps have been produced, and under these circumstances the use of colours is replaced by other distinguishing features such as dashed lines or patterns.

Another is the restricted resolution of edge angles – in all four maps it can be seen that edges have been restricted to angles that are a multiple of 45°. One exception to this is seen in Moscow, which chooses to use a circular ring line as a stylistic feature; however, all other edges follow this rule. This design feature is commonly referred to as octilinearity (specifically 45°), and is found in a large number published schematics. A smaller number of schematics use alternate resolutions for edge angles including 90° (rectilinearity) and 30°; even fewer do not apply any restriction on edge angles.

Station distribution in the published schematics has been made even across the diagram. This is especially apparent when comparing published schematics to their original geographic map, in particular for large networks such as Moscow and Sydney. Typically, stations in the city centre are close together, and this distance increases in the suburban periphery map sections. The effect of this is a complicated centre with clearer periphery sections; this is the opposite of what is desired as the city centre will most likely carry the most traffic and therefore be read most frequently. Designers deal with this by expanding dense areas whilst contracting sparse sections to create an evenly distributed



(a) Geographic.

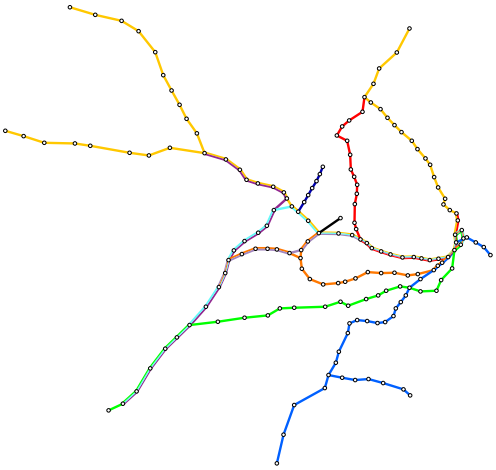
(b) Published.

Figure 7: Moscow metro - Copyright Moskovsky Metropoliten.

schematic. Along with an even distribution of stations, in most cases the distance between stations along lines is kept as equal as possible.

Lines in a geographic map will often contain many bends. Schematics aim to simplify this by removing all possible bends to keep lines as straight as possible. A good example of this can be seen in Moscow, in which the red line bisects the circular line from bottom-left to top-right, with only a single small bend at the end. The argument for removing bends is that the eye can then follow along the lines faster and with more accuracy. Vienna and Washington appear to not apply this technique due to their more geographic layout, explained below.

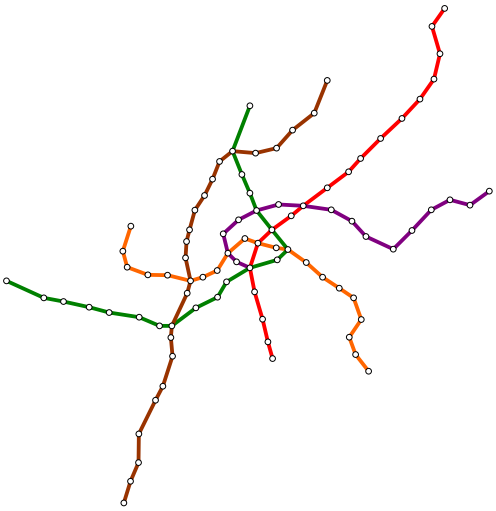
Many cities have prominent features such as rivers, parks, or seas that can be used as orientation tools to help create a reference between the abstract station positions and the world at ground level – these can be incorporated into the schematic in one of two ways. Either the feature is distorted and fitted around schematic lines as in Moscow and Sydney, or the feature remains less distorted and the schematic is adapted to work around it, as in Vienna and Washington. If



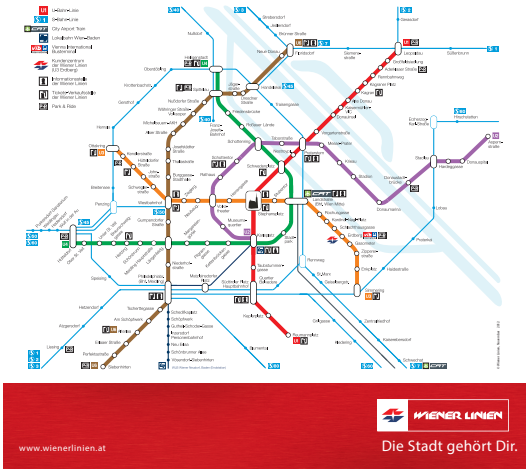
(a) Geographic.

(b) Published.

Figure 8: Sydney metro - Copyright RailCorp.

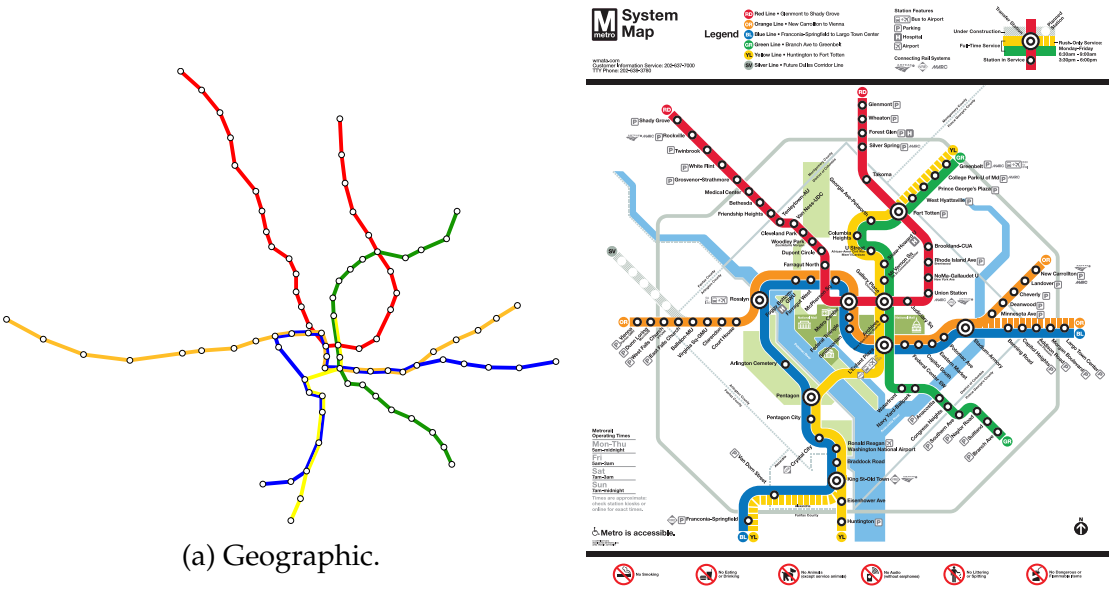


(a) Geographic.



(b) Published.

Figure 9: Vienna metro - Copyright Wiener Linien.



(a) Geographic.

(b) Published.

Figure 10: Washington metro - Copyright Washington Metropolitan Area Transport Authority.

done well, this can be beneficial as it is useful to know where you are positioned at ground level. However, often designers have allowed geographic features to alter the schematic to an extent that has a detrimental affect on comprehension, such as in Figure 4 – the Madrid metro map.

In line with including geographic features on schematics, it has become increasingly common for maps to abandon a purely schematic layout in favour of a semi-geographical approach. The Washington schematic is a particularly good example of this, as it contains many line bends which follow the line geography but go against typical schematic layout design.

The layout features explained here make up the metro map "fingerprint", and are applied as criteria in most automated layout techniques. This is explained in greater detail in Sections 2.2 and 2.3.

2.1.1.2 Production

Published metro maps, such as those shown here, are currently produced manually by graphical designers. The design of these maps is a very difficult task,

as it requires the designer to have expertise in both the psychology of map reading and schematic design. Besides this, metro maps can often contain hundreds of nodes and it's up to the designer to decide on effective positions to create the layout. This takes a great deal of time to complete, and many designers are also not experienced in design considerations pertaining to how metro maps are used.

For these reasons, the production of metro maps leaves much room for improvement. An automated layout technique capable of producing results of comparable quality to those currently drawn by hand would be a great benefit in terms of time and difficulty saved during production, and would also be developed to follow empirically tested layout criteria shown to aid comprehension, reducing the number of metro maps with bad design features; this benefit will result in faster and more accurate journey planning for passengers. Even if the automation of metro map layout is not able to produce an output of as high a quality as a human design in terms of aesthetics, it can still be used as an initial layout design step for designers, alleviating them of what is arguably the most difficult part in map production. After producing a good layout, designers could then finalise aesthetics to meet the transport systems' specifications.

2.1.2 The Metro Map Metaphor

The metro map metaphor refers to the use of a metro style schematic being used to visualise an abstract data set rather than a transport system. Examples of this use can be found in a wide range of subjects, produced by both private individuals and large organisations. This section examines a few such examples.

Figure 11 shows a schematic of our home galaxy, the Miky Way. Stations represent stars, nebulae, and points of interest, and lines represent arms of the galaxy. According to the designer, care has been taken to position stations in their actual positions, and lines have been drawn to follow the natural spiral of the galaxy. It is not clear if the lines represent recognised groupings of objects, or if they are simply drawn for aesthetic purposes in the shape of the galaxy, as suggested by line names such as *Express* and *Suburban*. Although this schematic is relatively small the layout is clear, consistent, and provides an easy to read

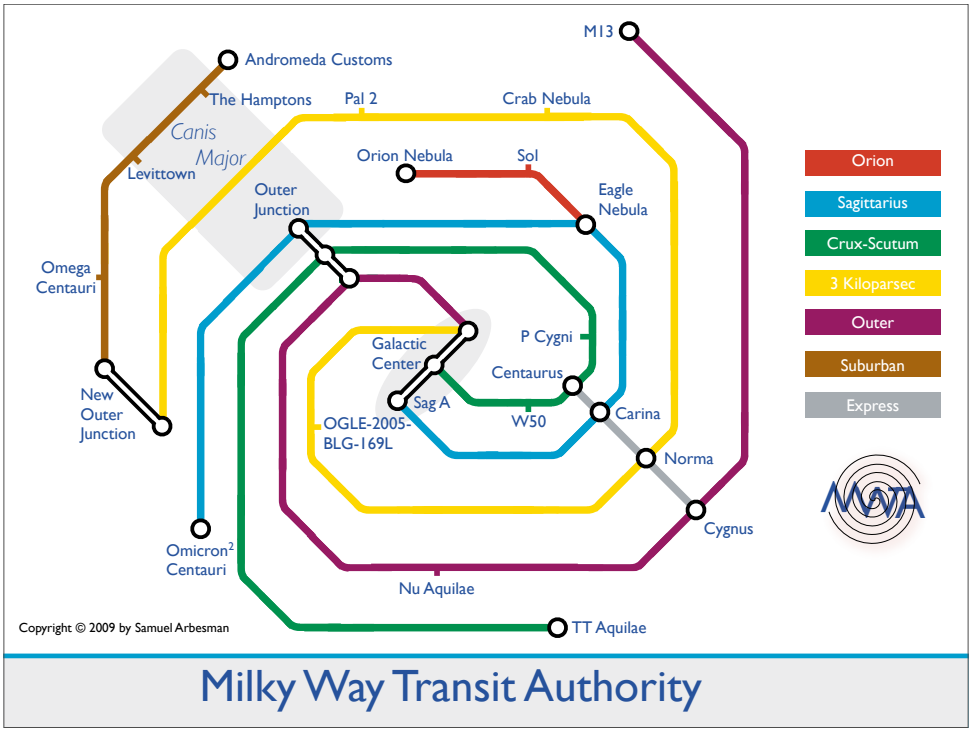


Figure 11: Stars and points of interest in our Milky Way system. Lines represent arms of the galaxy. (Arbesman 2014)

diagram.

Figure 12 is a schematic produced by technical consulting company Real Story, illustrating used vendors and their fields of expertise. This schematic is not based on geographical data, and therefore the positioning of the nodes has no significance. Despite this potential benefit of no node position constraints, the designers have chosen positions along irregular line angles, instead of conforming to any subset such as octilinearity. Along with this, lines are both straight and curved, and line bends are both curved and angular. The centre of the schematic is also unclear with no apparent junction alignment or positioning, introducing many line crossings in a confined area. A lack of uniformity across this schematic, combined with a confusing centre, makes this schematic very unordered and chaotic. However, this type of data is a perfect match for a metro map style schematic and in spite of its flaws, it still presents the information in an appealing and easy-to-read way.

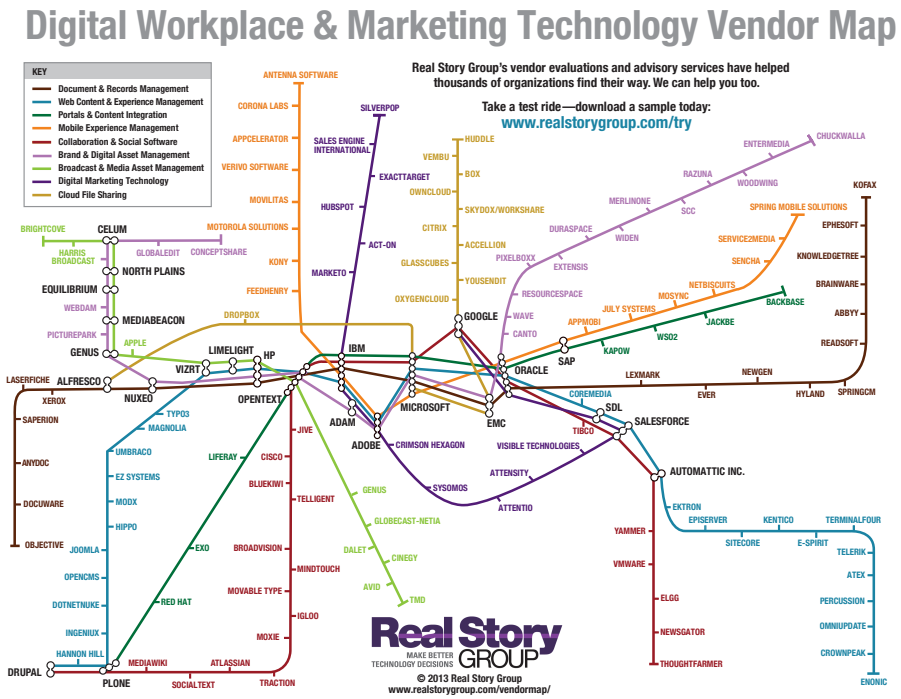


Figure 12: Technology venders used by a consulting firm. Lines represent the vendors field of expertise. (Group 2014)

Figure 13 is a metro map style schematic illustrating metro systems around the world. The lines in this diagram do not represent any connection between the nodes, and have been included purely for aesthetic purposes. At a first glance it may appear that the schematic is geographically correct, but further inspection shows only a very rough accuracy in city (node) positions. There are a large number of similarities between the layout of this schematic and that of the official London Underground (Figure 3), which indicates that it is a modification based upon an existing well-lived layout.

As discussed previously, published schematics used for transport system visualisations, such as the current London Underground map, are drawn by designers typically using vector graphics software packages. Providing a good level of expertise this can produce very high quality diagrams, however it does nothing to aid the designer with the key aspect of schematic design – the layout.

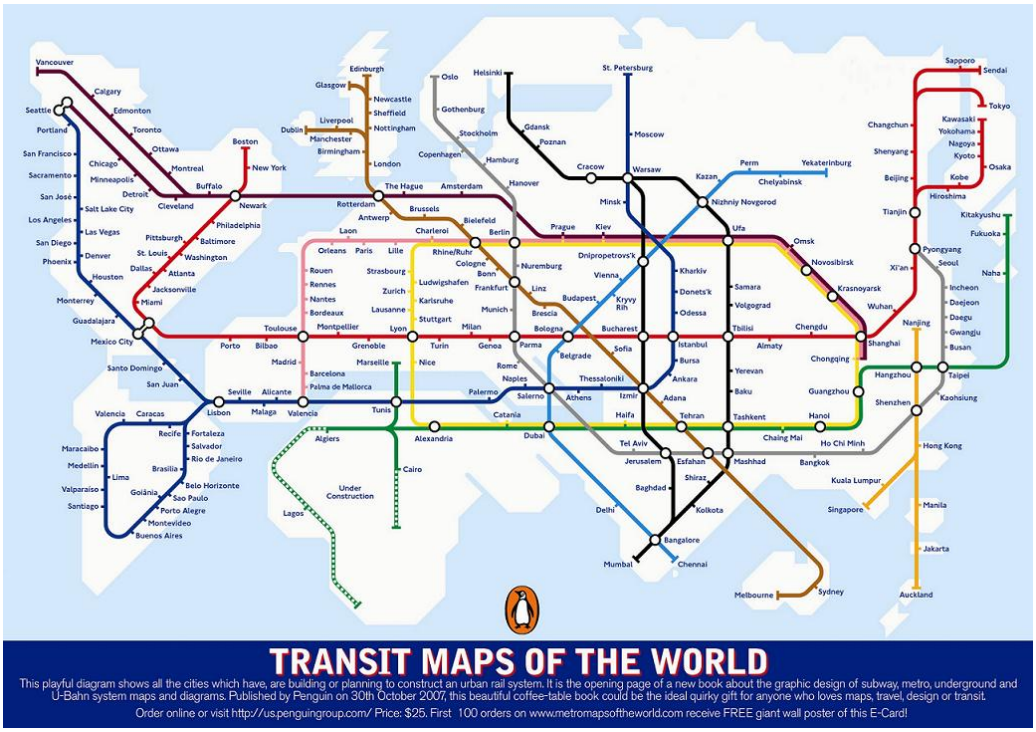
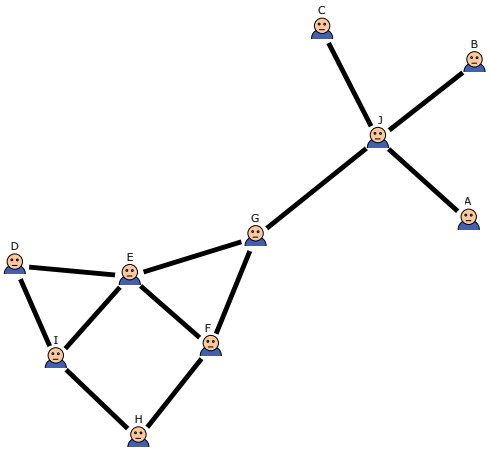


Figure 13: Metro map of metro maps from (Ovenden 2005). Lines are purely for aesthetic purposes.

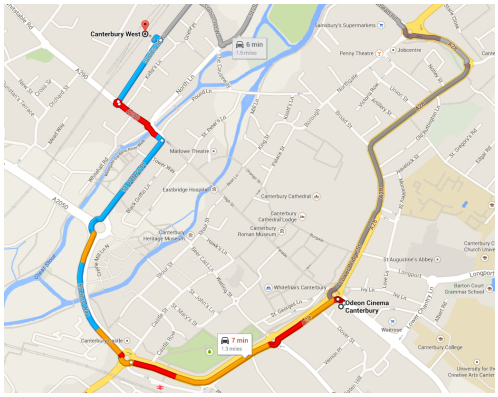
As there is no help in the creation of the layout, this results in many poor layouts, or schematics being based on previously drawn examples as in Figure 13. This section shows that there are many potential areas in which the visualisation technique can be applied, and many people have attempted to do so with varying levels of success, but the difficulty involved with layout design severely hinders its adoption as a visualisation technique. An system capable of automatically producing good layouts will therefore help amateur designers as well as, if not more so than, professionals.

2.2 Graph Drawing

Graphs play a very important role in a wide variety of disciplines; from visualisation tools for network structures such as social networks (Figure 14a) or transport networks as illustrated in the previous section, through to their use as



(a) Graphs can be used to depict social networks.



(b) Navigation systems rely on an underlying graph structure to perform routing tasks. Image from Google maps.

Figure 14: Examples of common graph usage.

data structures underlying other concepts such as in navigation systems (Figure 14b).

Traditionally, graphs were drawn by hand, but much research has been carried out on their automated layout. This section examines a selection of notable graph drawing methods. We focus specifically on 2D, undirected and unweighted node-link graphs, although most of these techniques are extensible to other graph subtypes. Automated layout techniques can be divided into two main categories: force-directed (2.2.1) and search-based (2.2.2).

Graph layout is an important foundation for research in automated schematic layout, as many schematic layout techniques lean upon existing methods in this area. The reason for this is that schematics are often represented as graphs, and so modified versions of graph drawing techniques can be used for the layout process.

In the modern world there are vast quantities of data that have the potential to be visualised as graphs, many of which frequently change. For example social networks or computer-based visualisations which respond to user interaction. The process of adjusting a layout to reflect changes in its underlying data set is called dynamic layout, and is further explained in Section 2.2.4. We also cover a concept commonly associated with dynamic re-optimisation known as

mental map preservation.

2.2.1 Force-directed Layout

The initial automated graph layout methods are typically variants of a force-directed layout. This layout technique relies on information contained within the graph structure, rather than external factors to perform the layout. Graphs are modelled as physical systems in which nodes and/or edges exert forces upon each other in order to move into desirable positions. Force-directed layout methods typically produce aesthetically pleasing layouts, but are limited in scalability due to the physical model being unable to escape local optima.

Typical run times of force-directed layout methods on modern machines using small to medium sized graphs are very fast, often producing an output within a fraction of a second; this is highly beneficial for use in interactive applications as users are able to optimise graphs without having to wait. Performance is a major advantage of force-directed layouts over their search-based counterparts explained in the following section which can take minutes, if not hours, to run. A disadvantage of these techniques is that new criteria can only be enforced by applying additional forces to the nodes, thereby causing them to move differently. This makes it very difficult to strongly enforce additional criteria as nodes are moved by summing all their forces in each iteration; the resulting composite force fully satisfies none of the applied criteria, and nodes are moved to unoptimal positions.

Later research in the technique extended both the performance and scalability of this type of method up to many thousands of nodes by using a multi-levelled approach, in which the graph is represented by a series of progressively simpler graphs.

2.2.1.1 Barycentric Tutte Embedding

The first methodology for graph layout was proposed in (Tutte 1963). The method uses a system of linear equations to position vertices at the barycentre of their neighbouring nodes, within the bounds of a fixed convex polygon.

This technique is categorised as force-directed layout as the method iterates towards a minimum energy vertex configuration. In his paper, Tutte proves that the method is capable of calculating a straight-line, crossing-free drawing for any given 3-vertex-connected planar graph, and that every face of the embedding is convex.

The method takes an input graph consisting of a number of vertices and edges $G = (V, E)$. In this graph, vertices V must be split into two groups, V_0 and V_1. Group V_0 must contain at least three vertices selected at random, these are to become fixed vertices; group V_1 contains the remaining vertices, to be positioned by the algorithm. For the most effective layout, group V_1 should contain more vertices than V_0. Along with these two groups of vertices, a convex polygon with $|V_0|$ vertices is required. During the algorithm, V_1 vertices will be positioned within this polygon. With the required input, the algorithm pseudocode is as follows:

1. `Place each vertex of` V_0 `at vertices of the convex polygon, and each vertex of` V_1 `at the origin.`
2. `For each vertex` v `in` V_1`:`

   $x_v =$ mean x`-value of all connected vertices.`

   $y_v =$ mean y`-value of all connected vertices.`
3. `Repeat step 2 until the` x `and` y`-values for each free vertex converge to a solution.`

The formula for calculating x_v and y_v for a given vertex is shown in Equation 1.

$$(x|y)_v = \frac{\sum_{(u,v)\in E} (x|y)_u}{degree\,(v)} \tag{1}$$

where v is a connected vertex.

Figure 15 shows an example output from a Tutte embedding of the mesh of a cube. The red vertices around the edge make up the fixed group V_0 and form the initially chosen convex polygon; the remaining white vertices belong to the free group V_1. The algorithm has successfully placed all free vertices at the barycentre of their neighbours.

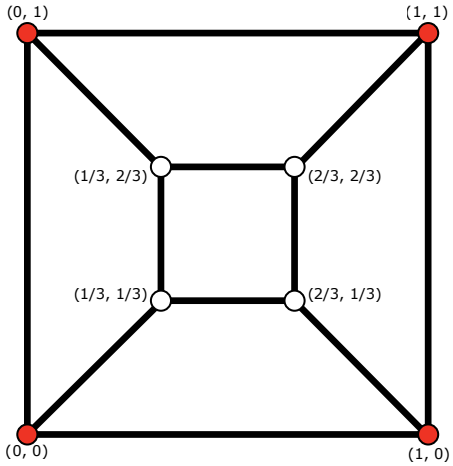


Figure 15: Tutte embedding of a cube. Red vertices indicate the initial fixed convex polygon (a unit square). Vertices are labelled with their position within the unit square.

As mentioned previously, an advantage of Tutte's method is a guarantee of finding a straight-line, crossing-free drawing for any given 3-vertex-connected planar graph. There is also only a single global solution meaning that the method does not produce inferior local optima results, as opposed to the more recent force-directed approached discussed next. Disadvantages of the method include the requirement that the graph must be planar and 3-vertex-connected, restricting it's potential use as a more general graph layout method. Another issue is that when applied to larger graphs, node spacing can be very unevenly distributed across the output as shown in (Eades and Garvan 1996).

2.2.1.2 Force-directed Layout

In 1984, Peter Eades presented a new force-directed layout method, providing an effective method for small to medium sized graph layout (Eades 1984). This method has since become the basis for subsequent force-directed techniques, and is still commonly used in graph layout. Eades attempts to meet two criteria with his method: edge lengths should be kept roughly equal across the graph, and the layout should display as much symmetry as possible. The justification for attempting to meet these two criteria is that they are considered "aesthetically pleasing" in a wide variety of application areas.

In the force-directed layout method, the graph is modelled as a mechanical system where vertices act as steel rings and edges act as springs between these rings. The vertices are then positioned in some initial layout before being released so that the spring forces move the system into a minimal energy state. The algorithm outputs the vertex positions from this stable state.

Two force types are used during the algorithm; the first of which are edge forces which act upon vertices. The force exerted on a vertex by an edge is calculated using Equation 2. Eades explains that a logarithmic function is used, as linear springs are too strong when the vertices are far apart.

$$f_a = C_1 \log\left(\frac{d}{C_2}\right) \tag{2}$$

where d is the length of the spring, C_1 is the spring strength, and C_2 is the ideal spring length.

Secondly, nonadjacent vertices repel each other using an inverse square law force (Equation 3).

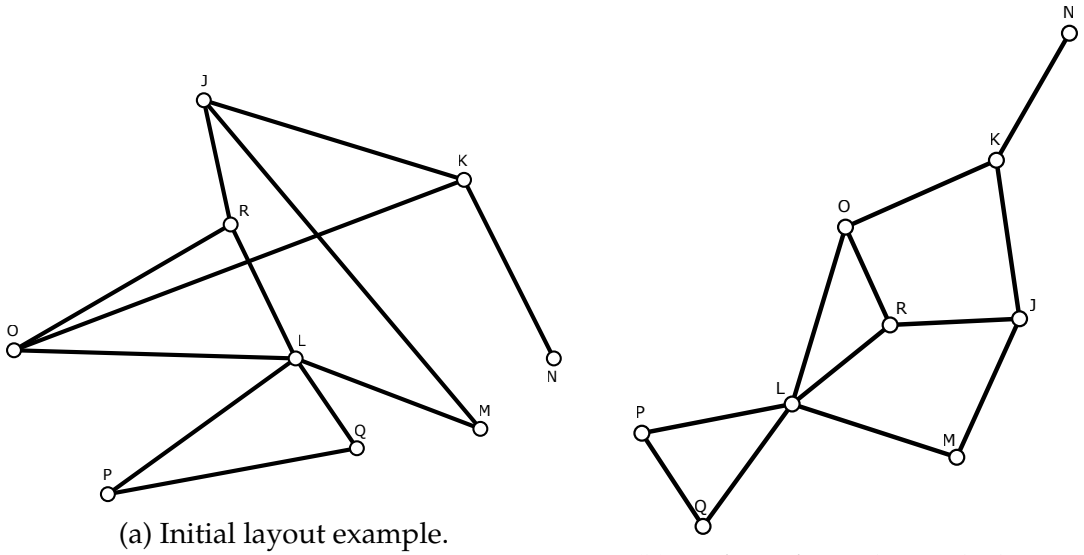
$$f_r = \frac{C_3}{d^2} \tag{3}$$

where d is the distance between the vertices, and C_3 is the repulsion strength.

Vertices are moved each iteration by $C_4 \times$ (force on vertex). C_1, C_2, C_3 and C_4 are constants in the algorithm, set at values 2, 1, 1 and 0.1 respectively. According to Eades, these values are appropriate for most graphs and almost all graphs achieve a minimal energy state after the simulation step is run 100 times.

Figure 16 shows the effect of a force-directed layout on a small graph containing nine nodes. Figure 16a is the input to the algorithm; the nodes in the input diagram can be either randomly positioned (as in this example), or be positioned by another criterion such as geographic location. Figure 16b shows the resulting layout after the force-directed method is applied and illustrates how the algorithm output meets the two initial target criteria; edge lengths have been roughly equalised when compared to the input graph, and elements of symmetry can be seen throughout.

Eades notes that there are several classes of graphs for which the algorithm produces poor layout: dense graphs, graphs with dense subgraphs, or graphs



(a) Initial layout example.

(b) After force-directed layout method is applied.

Figure 16: Example of a force-directed layout method. Note: labels have been manually positioned to cross-reference nodes and are unaffected by the layout algorithm.

with a small number of bridges (graph edges whose removal disconnects the graph). However, for a wide class of graphs the method can be used to produce fast, good quality layouts with roughly equalised edge lengths and improved symmetry.

2.2.1.3 Performance Optimisations

The first significant performance modification to the basic algorithm was detailed in (Quigley and Eades 2001) and was based upon a previous technique from (Barnes and Hut 1986). This modification originated from research into N-body simulations – modelling how multiple celestial bodies with individual gravitational fields interact together, and predicting future interaction. For the most accurate simulation there must exist a force between every pair of bodies (modelled as vertices), and as such has a time complexity of $O\left(N^2\right)$ – meaning that performance will decrease drastically as more vertices are added. Force-directed layouts, which are similar to N-body simulations as they also contain

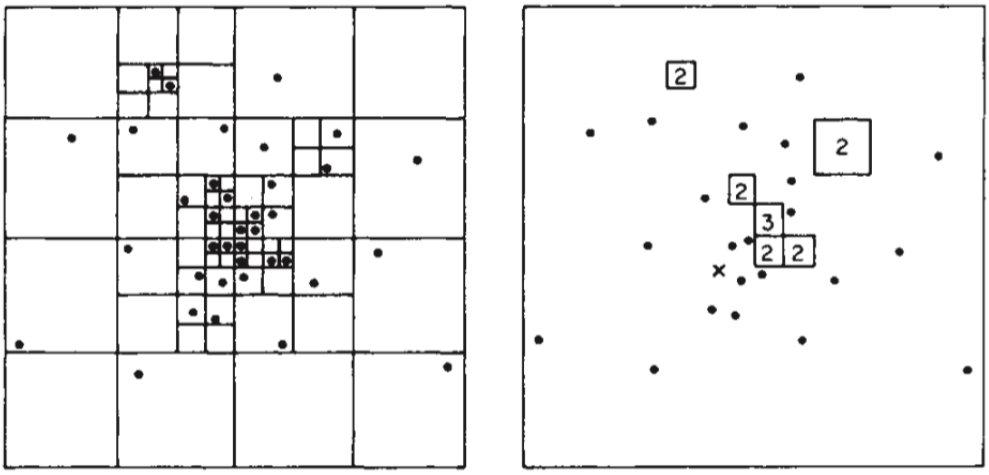


Figure 17: Hierarchical boxing (left) and force calculation (right) in two dimensions of the Barnes-Hut optimisation from (Barnes and Hut 1986).

multiple vertices and forces between all node pairs, share this same problem and this performance optimisation can greatly speed up the algorithm when more vertices are added.

Due to the inverse-square function used to calculate forces between vertices, the magnitude of the force between two distant vertices is very small. There are also often many distant vertices in a similar vicinity, for which the standard algorithm individually calculates forces for. The Barnes-Hut approach relies on grouping close vertices together into a single pseudo-vertex. This pseudo-vertex contains the total mass of the combined vertices and is positioned at their barycentre. This vertex can then used to calculate the force between the current vertex and all vertices in the group simultaneously, at the cost of a small loss in precision.

Figure 17 illustrates how the area is partitioned to create groupings (left), and how groups are used to calculate forces on vertex x (right). Vertex groups are calculated at the start of each iteration as follows; an initial empty cell large enough to contain all vertices is created, then each vertex contained within the cell bounds is added to the cell. If a cell contains more than one vertex, that cell is subdivided into four smaller child cells. This is repeated until each cell contains at most one vertex. This process creates a quad-tree of cells, from the root

cell containing all vertices down to child cells containing zero or one vertices (Figure 17, left). The final step in constructing the tree is to tag each cell with the location of it's centre of mass and the total mass of the particles contained within it.

Having constructed the cell-tree, the force on any particle p may then be approximated by a simple recursive calculation. Starting from the root cell which contains all vertices, let s be the width of the cells region and d the distance from the cells centre of mass to p. If $(s/d < \theta)$, where θ is a fixed accuracy degree parameter, then use the calculated force between this cell and p. Otherwise, resolve the current cell to its four children and recursively examine each in turn.

The authors tested their method using a 4096 particle body simulation and results indicated that the number of two-body interactions computed by their technique was 0.1 times the amount required by a direct-summation force calculation. They also comment regarding precision loss and that in practice, forces computed even with a large θ value (around 1.0) are still accurate to approximately 1% with little dependence on the number of nodes.

Another technique for the optimisation of force-directed layout was proposed in (Tunkelang 1998) and uses a method based upon an inexact line search to improve the speed at which nodes converge to their local optimum. Force-directed methods typically do not specify an objective function, but rather express its negative gradient for each node in the form of forces, the magnitude of which is mostly used as the distance that each is moved by. Using these forces to move nodes requires many iterations to achieve convergence on the local optimum, as nodes are only moved a small distance towards it each iteration. If an algorithm was capable of calculating, or even estimating, the position of the local optimum based upon the objective function gradient, convergence could be achieved much faster by moving nodes a greater distance each iteration.

The conjugate gradient method, along with line search, can be used to minimise an objective function in a single step based upon its gradient. However, because the objective function of a force-directed method is not quadratic, the conjugate gradient method cannot be used. An accurate polynomial interpolation line search is possible, but such an approach would require many gradient

recalculations. To this end, the author states that (page 6):

> *Our approach uses an adaptive step size. We us* [sic] *an empirically determined initial step size, and we increase or decrease this step size on each iteration based on the previous one. We perform a gradient evaluation to ensure that we do not overshoot; if our step size is acceptable, we use this computed gradient for the following iteration.*

Unfortunately, it is unclear how the step size is increased or decreased as no further details are provided on these or the gradient evaluation procedure. However, the empirical evaluation timings show a significant increase in optimiser performance for meshes and trees of various sizes. The author also tested hypercubes, for which performance improvements were unclear.

2.2.1.4 Graph Theoretic Distance

Kamada and Kawai presented a variation of Eades' force-directed layout method in (Kamada and Kawai 1989). Rather than using graph edges as fixed length springs between connected vertices, the authors add virtual springs between all pairs of vertices. These springs are then weighted by calculating their graph theoretical distance – this distance is found by calculating the shortest path through the graph between each pair of vertices. The algorithm functions by attempting to minimise the difference between the graph theoretical distance, and the Euclidean distance for each virtual spring. To this end, the basic constant attraction and repulsion forces used by Eades are not used, but rather each virtual spring will attract or repel it's vertices depending on its current Euclidean length proportional to its graph theoretical distance.

Figure 18 shows example output graphs from the Kamada and Kwai method. These graphs look very similar to a typical output by Eades' force-directed method, and share many desirable properties – symmetric drawings, equidistant edge lengths, and minimal edge crossings. An interesting point made by the authors is that *"The experiments have shown that the initial positions do not have a great influence on the resultant pictures."* – and that due to this, their method can be used as a fast visual check for isomorphism. This statement goes against common understanding that force-directed techniques are prone to local optima

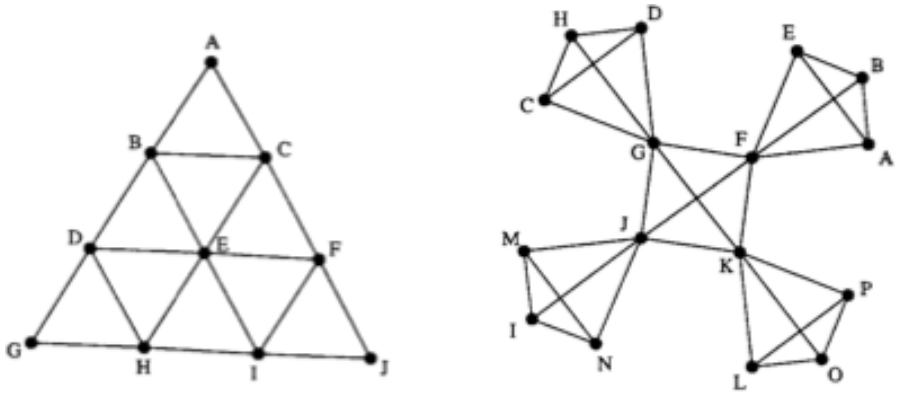


Figure 18: Example graph outputs for Kamada & Kwai graph theoretic distance method from (Kamada and Kawai 1989).

and that initial vertex configuration has a large effect on the output. Unfortunately, there is no further discussion or evidence to support this.

Kamada and Kawai put forward an novel alternative to Eades' force-directed method, but it is unclear if this technique provides any beneficial advantages over the existing method. In terms of map comprehension, it could be argued that in certain situations a Euclidean distance between vertices based upon their graph theoretic distance could be beneficial, although the accuracy of this criterion is not guaranteed and so can only be taken roughly. The algorithm is more computationally expensive, so there is a trade-off for this potential advantage.

2.2.1.5 Force-directed Layout Extension

Fruchterman and Reingold presented a variant of Eades' force-directed layout model in (Fruchterman and Reingold 1991) in which a number of modifications were made. The authors note how Eade's formulas do not reflect Hooke's law, and therefore came up with new formulas for attraction (Equation 4) and repulsion (Equation 5) which they claim are more closely related.

$$f_a = \frac{d^2}{k} \tag{4}$$

$$f_r = \frac{-k^2}{d} \tag{5}$$

where d is the distance and k is the optimal distance between vertices.

Unlike Eade's implementation which uses constants to define spring strengths which affect the least-energy distance between vertices, Fruchterman and Reingold calculate an optimal distance between vertices based upon the area of the graph and its number of vertices. This optimal distance k is calculated using Equation 6.

$$k = C\sqrt{\left(\frac{area}{number\ of\ vertices}\right)} \tag{6}$$

where C is a constant found experimentally.

When the attraction and repulsion formulas are plotted on a graph of force against distance, the point at which the sum of the two crosses the x-axis is where the two forces would exactly cancel each other out, and this happens when $d = k$. Using these formulas Fruchterman and Reingold state *"We have achieved results similar to those of Eades, as we will show, but we rejected his formula for* f_a [attraction] *since it was inefficient to compute."*

Along with changes to the force calculations, the authors introduce a global "temperature" level. The temperature value is initialised to a value determined by a function of the graph area and is used to cap the movement allowed by vertices, decreasing to zero over the duration of the algorithm. This modification changes the termination behaviour of Eades' algorithm from being reliant on vertices to find equilibrium positions, to being user-defined depending on a given cooling schedule (cooling schedule refers to the operation of the function used to manipulate temperature over the duration of the algorithm). The authors note that they conducted subjective experiments with two types of cooling schedule: the first starts at a high temperature and cools rapidly at a constant rate, whilst the second is set at a constant low temperature. They concluded that results from a constant low temperature schedule were superior and required less iterations.

As with other force-directed layout methods, the major time consuming aspect is calculating repulsion forces between each pair of vertices. The authors use a combination of two methods to address this problem. Firstly, the graph area is divided into a grid of resolution $2k$ – twice the optimum vertex-vertex

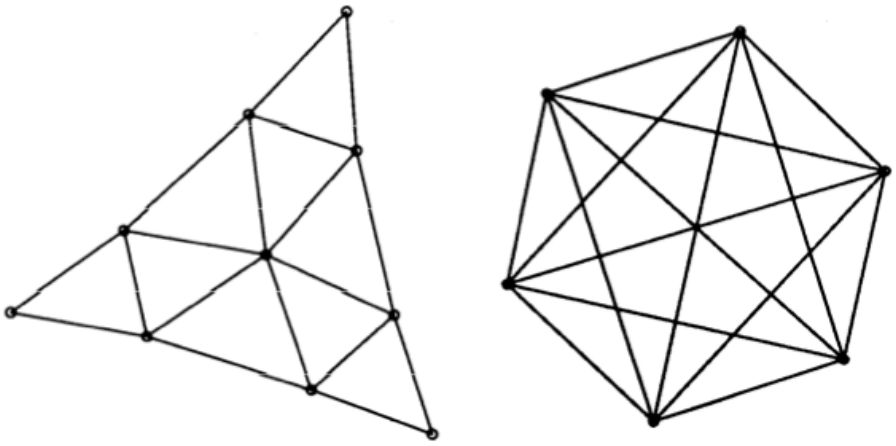


Figure 19: Example graph outputs for Fruchterman & Reingold force-directed layout (Fruchterman and Reingold 1991). Note: the left graph is the same structure as that in Figure 18 by Kamada & Kwai.

distance found in Equation 6. Vertices are placed in their grid squares at each iteration, and repulsive forces between vertices are only calculated if they lie in neighbouring cells. Using this technique, the authors state that *"In practice, we found that the square of the grid boxes caused distortion"* and to solve this a further check was introduced, after the neighbouring cell check, in which repulsion is ignored if the distance between the vertices is $> 2k$.

Figure 19 shows example graph outputs from this algorithm. Much like alternate force-directed layouts, the results are aesthetically pleasing with good symmetry and node separation. When comparing the left graph to the same graph produced by Kamada and Kwai (Figure 18), a slight difference can be seen in the perimeter edge – the edges in the Fruchterman and Reingold approach have been pulled in, compromising the straight edge. This could be due to the graph theoretic distance approach resulting in more even distribution across all vertices, rather than limiting forces to a localised area.

2.2.1.6 Scalable Force-directed Layout

One of the main drawbacks for basic force-directed layout algorithms is the lack of scalability; a good layout can be produced for up to approximately 40 vertices, but scaling beyond this is problematic for two reasons. Firstly, the time

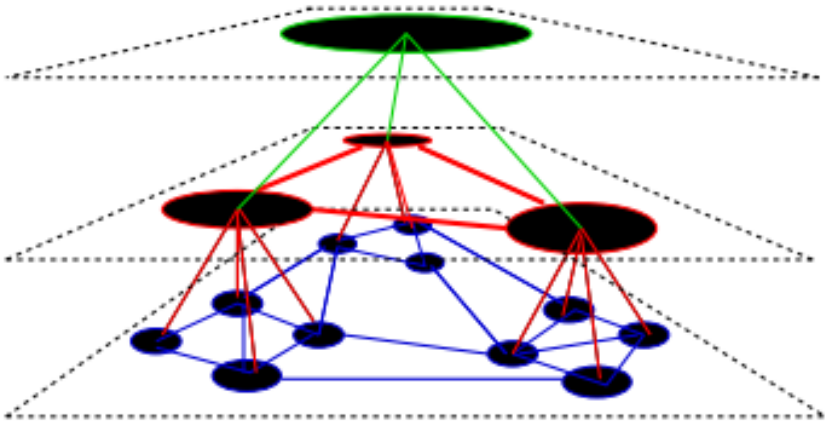


Figure 20: An example illustrating the layers of abstraction in a multilevel graph from (Eades and Feng 1997).

complexity of these algorithms is typically $O(N^2)$ and as such adding too many nodes is infeasible in terms of performance. Secondly, as the size of the graph increases, the number of possible suboptimal configurations also increases – raising the likelihood of a poor result.

Chris Walshaw presented a scalable force-directed layout technique in (Walshaw 2001), extending the work on multilevel graph visualisation in (Eades and Feng 1997). A multilevel visualisation of a graph uses a number of levels of abstraction to group together nodes in a similar vicinity. Figure 20 illustrates a multilevel graph with two levels of abstraction; the blue graph (bottom) is the most detailed version of the graph including all vertices. The green graph (top) shows the coarse graph as a single vertex and the red graph (middle) shows how close vertices are grouped together.

This concept is used by Walshaw to simplify large graphs to the point where they are suitable for a force-directed layout. The process of grouping nodes to form clusters is repeated until the number of vertices falls below some threshold. A force-directed method is then applied to the simplified graph before ungrouping vertices to their clusters of the previous abstraction level. This process is repeated until the original graph is optimised.

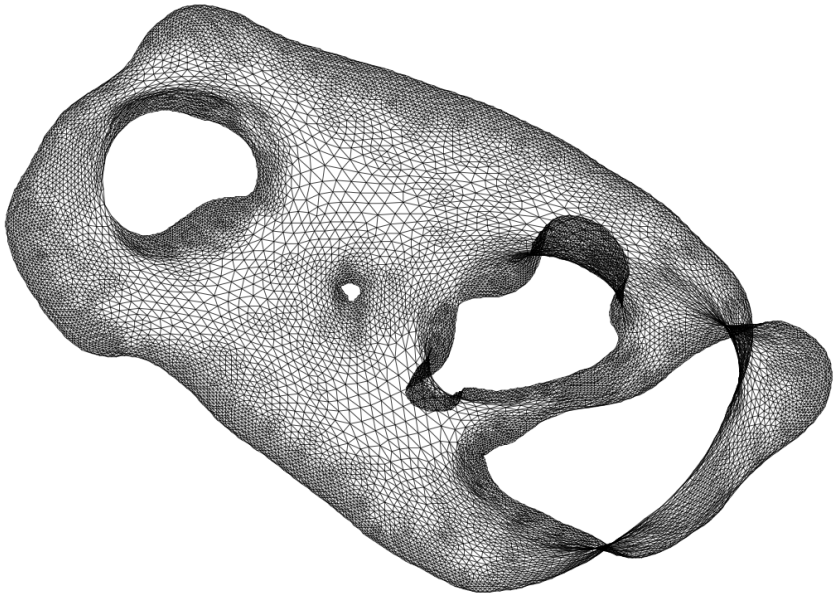
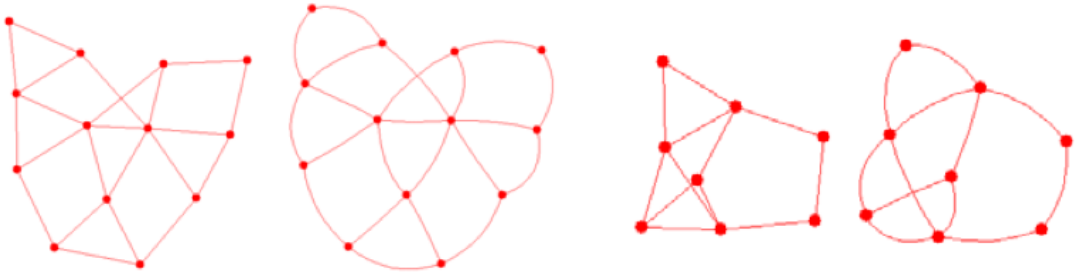


Figure 21: Example of Walshaw's multilevel optimiser on a graph with many vertices from (Walshaw 2001).

Using this method, Walshaw's algorithm is capable of producing good optimisations of very large graphs with up to 100,000 vertices. Optimisation timings for such large graphs are understandably larger – 30 seconds for a 10,000 vertex graph to around 10-20 minutes for the larger graphs; however, when applied to a smaller graph, the method is no slower than any other technique as none or very few abstraction levels are required. Figure 21 shows an example of Walshaw's multilevel optimiser on a very large graph. The result shares characteristics seen in previous force-directed layouts on smaller graphs; approximately equal edge lengths with regular vertex positioning, and symmetry amongst vertices. The data for this graph was taken from a fluid simulation, and as such displays an extremely high density of nodes at certain edges.

2.2.1.7 Curvilinear Graphs

The large majority of layout techniques focus on graphs using straight edges between vertices, but there has also been research into producing layouts using curved edges. Using curved edges provides an additional level of flexibility in



(a) (b)

Figure 22: Layout improvements obtained by Finkel and Tamassia's curvilinear force-directed method from (Finkel and Tamassia 2005).

graph drawing, introducing the ability to modify the angular resolution of a vertex without affecting the position of adjacent vertices. High angular resolution is a desirable trait for a graph layout as it provides a clean look, increasing readability – this is evidenced by the numerous methods implementing this constraint.

Initial attempts at curvilinear graphs focused on planar polyline drawings (Gutwenger and Mutzel 1998) and edge routing between fixed vertex positions generated using existing layout methods (Dobkin et al. 1997)(Brandes and Wagner 1998). A common approach was to use an edge routing method to construct a polyline, and subsequently use the vertices to construct a Bèzier spline (Goodrich and Wagner 1998). These approaches are capable of producing curved connections between nodes resulting in an improved angular resolution, but still leave room for improvement as they are unable to move the vertices that they connect.

Finkel and Tamassia present a method which combines curvilinear edges into a force-directed layout method in (Finkel and Tamassia 2005). They embed Bézier spline control points along edges as dummy vertices which are then included in the force-directed layout method. Figure 22 shows two example graph comparisons between a straight-line layout and Finkel and Tamassia's curvilinear force-directed method. These examples show how effective using curved edges can be in increasing the angular resolution of edges around vertices; in particular, the layout of Figure 22b has been much improved.

2.2.2 Search-based Layout

Search-based graph layout methods use mathematical optimisation techniques to improve layout using an objective function which is used to quantify the aesthetic quality of a graph. The layout is then refined by repeatedly performing small modifications to the graph configuration to optimise the objective function. These methods typically take much longer to run than a force-directed approach, as many possibilities are evaluated in order to find a better layout. However, the extensibility of the objective function calculation in these methods allows for much easier implementation of additional criteria. There are too many possible layout configurations to perform an exhaustive search, and therefore these methods use a number of heuristics to limit the search space; this often results in the method only obtaining a local optimum, although certain methods such as simulated annealing have attempted to mitigate this.

2.2.2.1 Simulated Annealing / Multi-criteira Hill Climbing

Davidson and Harel propose the use of a simulated annealing optimisation technique for automated layout of undirected graphs (Davidson and Harel 1996). The authors claim that their method produces good results for graphs of a modest size, comparable to those produced by alternate methods including Eades' force-directed technique.

The authors' approach implements several simple criteria, listed below, many of which are also present in the force-directed approach taken by Eades:

1. Even distribution of nodes.
2. Uniform edge lengths.
3. Minimal edge crossings.
4. Keeping nodes away from edges.

A cost function is defined for each criterion, which provides a numerical measurement to be taken of how well a schematic adheres to it. These cost functions typically produce values which vary by orders of magnitude, making them incomparable. In order to work around this, the authors add a weighting to each

function. Once the cost functions are comparable they can be summed to produce a value indicating the total fitness of the schematic to all criteria; this is called the objective or loss function and can be subjected to a general optimisation method – in this case simulated annealing.

After choosing some initial schematic configuration most iterative methods alter the layout at each iteration, re-evaluate the objective function and possibly replace the previous configuration with it, for example if the new layout reduces the objective function. This process continues over a set number of iterations or until some termination condition is satisfied, for example if there are no more moves that improve the objective function. The procedure ends in a minimum energy configuration, but it is generally a local minimum rather than an optimal global minimum.

Simulated annealing differs from standard iterative improvement methods by allowing "uphill" moves – moves that spoil, rather than improve, the temporary solution. This is controlled using a temperature value which decreases over the duration of the method, lowering the likelihood of choosing an uphill move as the number of iterations increases. Pseudocode for the simulated annealing method is as follows:

1. Choose an initial configuration σ and an initial temperature T.

2. Repeat the following (usually some fixed number of times):

    a) Choose a new configuration σ' from the neighbourhood of σ.

    b) Let E and E' be the values of the objective function for σ and σ' respectively:

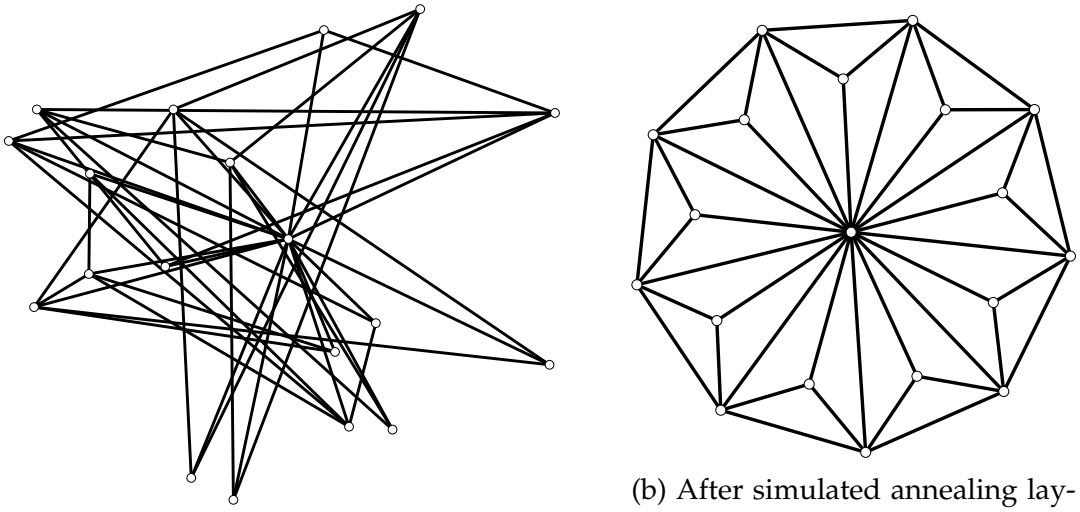
        $\triangleright$ if $E' < E$ or *random* $< e^{(E-E')/T}$ then set $\sigma \leftarrow \sigma'$.

    c) Decrease temperature T.

    d) If the termination rule is satisfied, stop.

(In step 2b, *random* stands for a real number between 0 and 1, selected randomly.)

This simulated annealing approach has advantages over force-directed methods including the ability to easily add new criteria, as long as a cost function can



(a) Initial layout example.

(b) After simulated annealing layout method is applied.

Figure 23: Example of the simulated annealing layout method from (Davidson and Harel 1996).

be defined for numerically quantifying it. Criteria weights can also be adjusted to allow control over the strength of individual criteria. Another advantage is that simulated annealing is able to escape from local optima, allowing the method to find a "better" result and be less sensitive to initial configuration. Disadvantages of the simulated annealing approach include a long optimisation time, typically much longer than that of a force-directed method. This is largely due to the requirement of re-calculating the objective function at each new configuration, most of which will be discarded. It should also be noted that, due to the randomised aspect of configuration choice, the obtained result is nondeterministic.

Figure 23 shows the result (23b) of the simulated annealing method applied to a typical input graph (23a). The result is an aesthetically pleasing layout which manages to follow all algorithm criteria – nodes are evenly distributed, edge lengths have been equalised somewhat, edge crossings have been avoided, and nodes have been kept away from other edges. Besides this, the method has managed to produce a planar graph without prior knowledge of planarity, which can be attributed to the addition of the edge crossings criterion. The algorithms' ability to escape local optima greatly aids when finding a planar layout

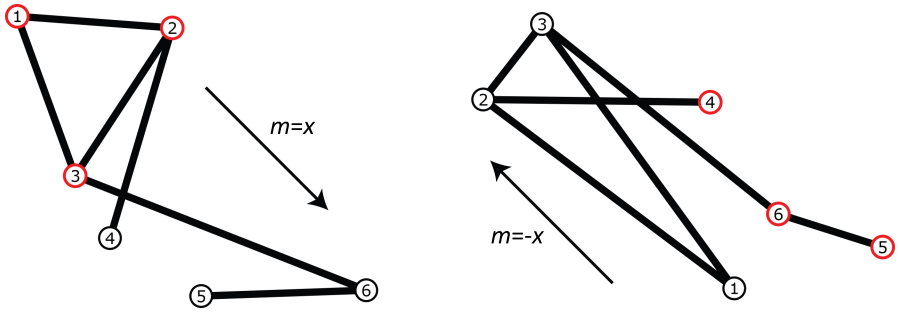
from an input with many crossings, which is something that a force-directed approach can struggle with. The output also has a high level of symmetry present – an emergent property from the defined criteria.

In conclusion, the simulated annealing approach by Davidson and Harel is an effective layout technique for small to medium sized graphs. It is capable of producing aesthetically pleasing schematics which conform to a number of user-defined criteria. A major benefit of this method is the ease at which new criteria can be added and their strength adjusted, providing much more scope for extension to suit a wide range of applications. However, these benefits come with a performance trade-off, so it cannot replace a force-directed method in all situations.

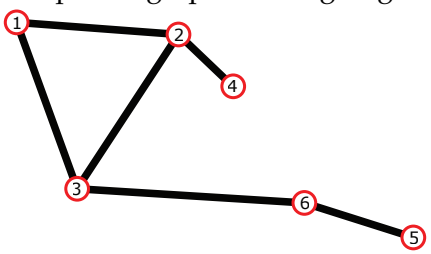
Simulated annealing can be simplified by removing the concept of temperature and disallowing moves that produce a worse result – a technique known as multi-criteria hill climbing. A new configuration is only accepted if the value of its objective function has been improved over that of the existing solution. The benefits of this simplification are that it will converge on a solution much faster. It does, however, restrict the layout to a local optimum result. There are a number of variations on the hill climbing technique including next ascent and best ascent methods. Next ascent performs random neighbourhood changes and chooses the first configuration that has a lower objective function value (and is non deterministic). Best ascent fully searches the local neighbourhood and selects the best configuration found (and is deterministic).

2.2.2.2 Hybrid Genetic Algorithm

General genetic algorithms have been successfully used in a range of graph-related optimisation problems such as pathfinding and network optimisation (Piggot and Suraweera 1995)(Schweitzer et al. 1997). However, these systems use one-dimensional encodings which makes them unsuitable for applying multiple global criteria. Hobbs and Rodgers propose the use of a hybrid genetic algorithm for aesthetic graph layout in (Hobbs and Rodgers 1998). Based on five well known measurable aesthetic criteria for graphs, the authors use the following criteria in their objective function:



(a) The two parent graphs undergoing crossover.



(b) The resulting child graph.

Figure 24: Example of graph crossover in a genetic algorithm layout approach from (Hobbs and Rodgers 1998). Arrows indicate vertex selection gradient, red vertices indicate the selected vertices from each parent.

1. Total edge length – minimise the sum of all edge lengths.
2. Graph area – minimise the area of vertex-bounding rectangle.
3. Vertex overlap – penalty for close or overlapping vertices.
4. Angular resolution – penalty for acute angles between edges connected to the same vertex.
5. Edge crossings – penalty for crossing edges.

As with other layout techniques utilising an objective function, these criterion are weighted before summation to ensure comparability.

An initial population of graph configurations is randomly generated from a given graph of vertices and edges. From these, four are randomly selected and the two fittest graphs of these are chosen for reproduction. The authors use a geometrically-based gradient crossover function which adds vertices to the child graph by selecting vertices from the parents along a randomly generated

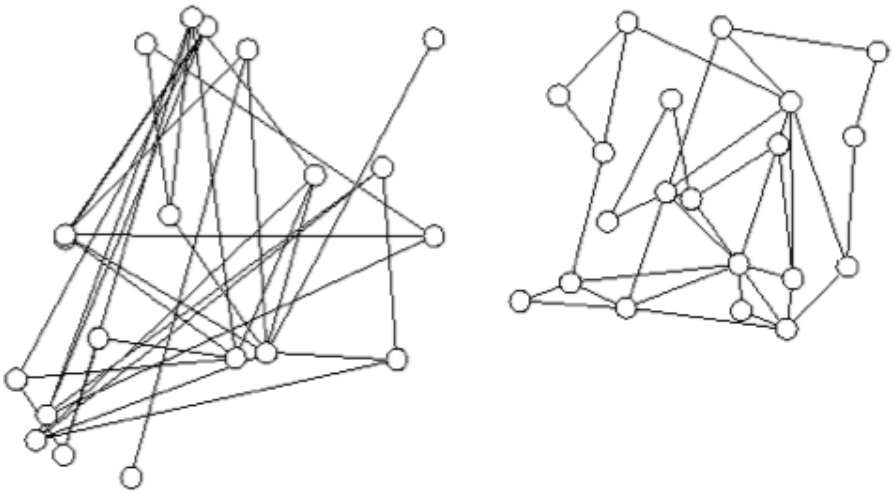


Figure 25: An initial graph (left) and aesthetically optimised result (right) from (Hobbs and Rodgers 1998).

gradient – one parent ascending and one descending the gradient. A vertex is selected from each parent in turn along the gradient and added to the child graph; if the chosen vertex has already been selected, the next unused vertex is used. This procedure, illustrated in Figure 24, creates a new child graph whilst preserving spacial relationships and topological features. Based upon the new child graph, a new population of graph configurations is then generated by mutation. A random selection of vertices from the graph are moved a random distance along both the x and y-axes. Mutation likelihood and size of movement are controlled by the user.

Figure 25 shows an example of the method applied to a randomly generated graph with 159 edge crossings. The result of the algorithm, shown on the right, was selected from the final population and has only four edge crossings remaining. This result was obtained by running the GA with a population of 20 for 1000 generations, although the authors state that most improvements were seen by around 250 generations. This is a positive result, as the method has shown to be effective in removing edge crossings. However, the graph does appear quite irregular; this is perhaps caused by criteria 1 and 2 used to minimise edge lengths and graph area respectively, and could possibly be improved by including additional criteria for uniform distribution and minimum distance

between vertices/edges as in Davidson and Harel's simulated annealing approach.

Benefits of this method include a greater ability to escape local optima than force-directed or simulated annealing techniques, along with a more scalable time-complexity to achieve good results. This latter point allows the method to optimise much larger graphs, which may be infeasible for alternate layout techniques due to optimisation time and/or local optima. The use of a multi-criteria objective function also allows easy implementation of additional criteria.

2.2.3 Additional Notes on Force-directed and Search-based Layout

Many examples used to illustrate the results of a graph layout technique are symmetrical – this can be seen in Figures 15, 18, 19 and 23. The reason for using symmetrical graphs is that they provide an easy way to visually verify the effectiveness of a layout method. This is because symmetrical layouts are visually appealing, and therefore justified as being deemed "effective" layouts (effective being a graph that is quickly and accurately traversable). It is therefore much easier to say that the layout method provides good results if it manages to obtain this expected layout, rather than using a non-symmetric graph which does not have such an obvious expected output. This is not to say that these layout methods cannot also produce quality optimisations for non-symmetric graphs, as can be seen in Figures 16, 21 and 22.

The previous sections indicate that force-directed approaches are more efficient than search-based approaches, yet perhaps do not fully clarify the reason; the justification for this is as follows. Force-based approaches operate by summation of the forces applied to each node. This resulting force is equal to the negative gradient of an objective function to optimise node positions. As such, the algorithm knows exactly the direction in which to move nodes in order to quickly minimise the function – there is no wasted movement. In search-based techniques, is it not possible to directly calculate the gradient of the objective function, and so a large number of trial-and-error node movements are required

to determine the general direction for each node at each iteration. It is these necessary, yet wasted, node movements that make search-based approaches much slower than their force-directed counterparts at reaching a solution.

Similarly, this reason contributes to the difficulty of implementation of additional criteria into force-based approaches. Each additional criterion must be expressed as a force for which the gradient of it's objective function can be calculated.

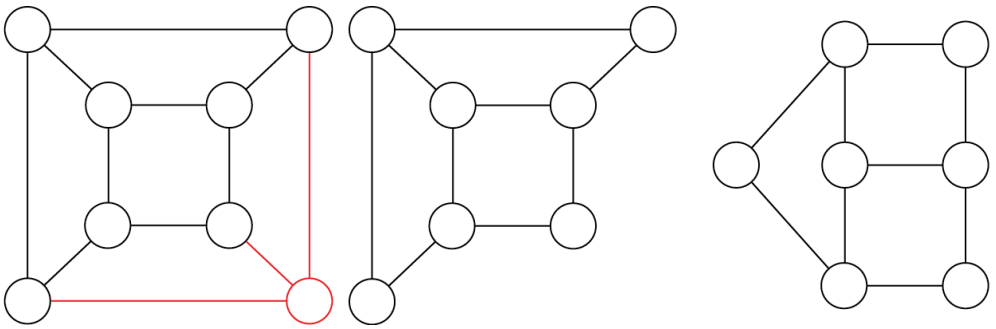
2.2.4 Dynamic Layout and Mental Map Preservation

Dynamic graph layout is concerned with the visualisation of data sets which continually change. These changes typically consist of nodes/edges being added or removed, or other transformations which reflect changes in the underlying data. When this happens, it is desirable to modify the graph to accommodate the changes whilst still adhering to the layout criteria used. Many such changes can be reflected in the graph with simple modifications, but some require considerable changes to node positions in order to produce a high quality layout. A naïve approach to re-enforcing layout criteria after a modification is to perform an algorithm designed for static layout on the whole graph. However, this approach can drastically change node positions in the visualisation and thereby hinder readability, as users must re-familiarize themselves with node positions – this problem gives rise to the concept of mental map preservation (Eades et al. 1991).

From (Diehl and Görg 2002) (page 2):

> *The term mental map refers to the abstract structural information a user forms* [cognitively] *by looking at the layout of a graph. The mental map facilitates navigation in the graph or comparison of it and other graphs. In the context of dynamic graph drawing changes to this map should be minimal, in other words algorithms to draw sequences of graphs should preserve the mental map.*

The reason for this is that large node movements between graph states can confuse users as a topologically identical graph can look radically different. Consider the example graph G_n in Figure 26, after deletion of the red node and its



(a) G_n. (b) $G_{n+1}(0)$. (c) $G_{n+1}(1)$.

Figure 26: Preservation of the mental map between sequential graph states.

associated edges, the mental map will be similar to the graph $G_{n+1}(0)$. However, an optimisation algorithm not designed to preserve the mental map may output $G_{n+1}(1)$ which is isomorphic, but for a user this is not immediately obvious. Because of this, a dynamic algorithm should take into account the previous state of the schematic and aim to minimise the change in layout.

An example of a technique to preserve the mental map includes implementing an additional layout criterion into a static method that limits node movement during re-optimization, in order to retain similarity and alleviate the issue of re-familiarisation. A key paper on mental map preservation (Misue et al. 1995) states that *"Intuitively, layout adjustment should preserve proximity relations: items which are close together should stay close together"*; however, subsequent implementations of mental map preservation techniques restrict absolute node movement without regard to proximity relations. For example, Kelly Lyons in (Lyons 1992) constructs a Voronoi diagram upon the schematic and uses the resulting node-enclosing polygons to restrict absolute node positions. Another technique includes that used by Graphael (Forrester et al. 2005), where the authors construct inter-timeslice edges and vary their strength in order to limit node movements. There have been a number of studies on the effect of mental map preservation on user readability and accuracy (Archambault, Purchase and Pinaud 2011)(Archambault and Purchase 2012)(Purchase, Hoggan and Görg 2007)(Purchase and Samra 2008)(Saffrey and Purchase 2008); however, none have identified the technique to have any significant effect.

2.3 Automated Schematic Layout

Considerable effort has gone into developing automated methods for schematic layout. These methods are extensions on standard graph drawing techniques in that they must perform the same basic tasks with the added requirement of adhering to additional aesthetic criteria.

Metro maps are commonly used when developing automated schematic layout techniques. This is due to a large number of reasons: 1) Availability of predefined layouts in the form of real-world maps. 2) Variance of layouts both in size and difficulty. 3) Published, hand-drawn, metro schematics can be used as benchmarks for algorithm performance. 4) Published metro maps share a similar set of layout criteria which can be emulated. 5) Familiarity/popularity of maps to a wide audience. Early layout methods, including (Avelar and Müller 2000)(Hong, Merrick and do Nascimento 2005)(Stott and Rodgers 2004), were the first to compile a number of criteria derived from examination of published schematics, and subsequent methods often implement many of these as a baseline. These criteria, covered in the following sections, are generally accepted to improve the readability and aesthetic appearance of a schematic to create a metro map style diagram.

The following sections provide an overview of notable schematic layout techniques which have been implemented, and explain the advantages and disadvantages of each method. Martin Nöllenburg covers the background of automated metro map layout algorithms with further emphasis on computation in (Nöllenburg 2014).

2.3.1 Topologically Correct Schematic Maps

Before the use of metro maps became commonplace for developing schematic layout techniques, Avelar and Müller used road network data to present the first schematic layout technique in (Avelar and Müller 2000). They discuss the characteristics of schematic maps including straight lines, a reduced number of fixed line angles, and contrasting colours to differentiate transportation lines. With this in mind, they choose to implement the following criteria into their

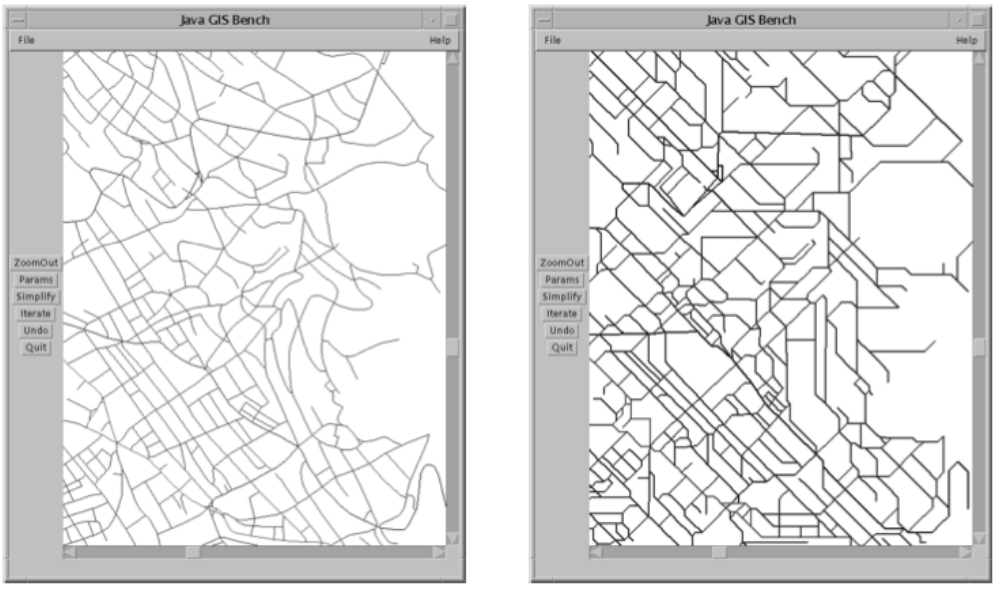


Figure 27: Example of streets in a standard map and a derived schematic map by Avelar and Müller from (Avelar and Müller 2000).

layout technique:

1. Straight lines which can be horizontal, vertical, or diagonal (angles of 45°).
2. Maximum length for straight line sections.
3. Minimum distance between lines.

An important aspect of their method is that these are all binary criteria, either met or not met, and there is no value representing fitness. Along with these criteria, the authors emphasise the importance of topological correctness in the resulting layout, and state the following properties which must be adhered to throughout the method:

1. Line crossings, where present in the input map, must not be removed.
2. Line crossings must not be introduced.
3. Cyclic order of outgoing connections around any node agrees with the ordering of connections in the input map.

This method uses the network lines to be schematised as the input to the algorithm, and the Douglas-Peucker line simplification (Douglas and Peucker 1973) preprocessing step is applied to remove line crenulations. The resulting long line sections are then cut according to the user defined maximum line length. The next step finds an improved location for each node by evaluating each in turn; if the examined node does not satisfy all criteria, then it is moved to the nearest position that does. This process is performed iteratively until the user decides the schematic reaches an acceptable appearance. Figure 27 shows an example of how the method can successfully schematise a large public road network.

2.3.2 Octilinear Force-directed Layout

The first attempt at the layout of metro maps was performed by Hong et al. in (Hong, Merrick and do Nascimento 2005). The authors devised a number of criteria for metro map layout by studying existing hand-drawn metro maps from all over the world, in particular Harry Beck's map of London. These criteria extend the observations of Avelar and Müller, and are defined as follows:

1. Each line is to be drawn as straight as possible.
2. Minimum edge crossings.
3. Minimum overlapping of labels.
4. Lines mostly drawn horizontally or vertically, with some at 45 degrees (octilinearity).
5. Each line is to be drawn with unique colour.

Hong et al. chose to use a force-directed layout technique for metro map optimisation, this is an extension of Eades' method described in Section 2.2.1.2. Due to the characteristics of force-directed layout, criteria 1, 2, and 3 are naturally enforced, and the main challenge involves enforcing criterion 4, octilinearity. In order to attempt this, Hong et al. combined the standard force-directed layout method with a magnetic spring model by Sugiyama and Misue (Sugiyama

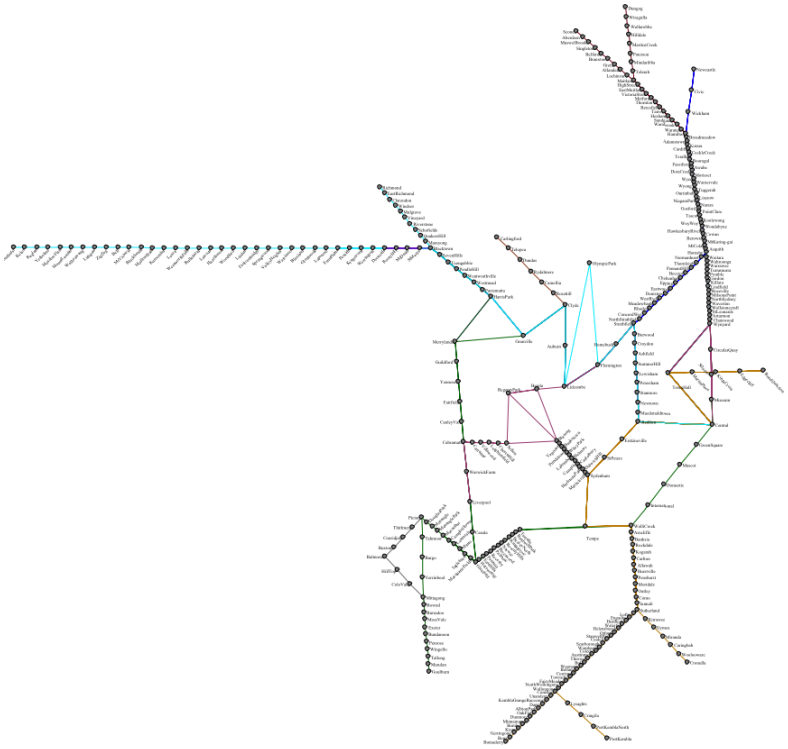


Figure 28: Sydney metro map produced by Hong et al. using their spring embedded method from (Hong, Merrick and do Nascimento 2005).

and Misue 1995) which rotates edges around their midpoint by applying additional "magnetic" forces to end nodes. Sugiyama et al. use this technique in their paper to apply an overall structure to the resulting graph, for example a tree with edges mainly in the y-axis or graphs following polar coordinate axes; they also demonstrate its use for drawing orthogonal graphs – a step towards octilinearity.

Hong et al. use this magnetic spring model in combination with the force-directed method to attempt to rotate edges to the closest multiple of 45° whilst also undergoing the standard graph layout. Figure 28 shows the result of this method applied to the Sydney metro map, and it can be seen that edges in the schematic have been roughly aligned to a multiple of 45°. However, along with a number of other characteristics including stations not being evenly distributed and sharp line bends, the slightly-off octilinearity makes the schematic less appealing and hinders its readability when compared to published maps.

An advantage of this layout method is the speed at which optimisation is performed, taking only 7.6 seconds to produce the example shown in Figure 28. Along with this, the optimisation process of force-directed methods can be viewed in real time as they produce an aesthetically pleasing animation from the initial embedding to the final layout; this is in contrast to search-based methods where nodes quickly jump around as the algorithm searches for ideal positions. This advantage should be taken into account for applications where real-time optimisation with user interaction is required, such as dynamic layout, as the animation can substitute for the wait required with alternate methods.

The main disadvantage of this method is that multiple force-types, each attempting to optimise a specific criteria, cannot be fully enforced when run simultaneously. The reason for this is that the multiple forces are combined into a single movement vector for each node during each iteration, and conflicting forces result in little or incorrect movement required to fit one specific criterion. This effect can be seen in Figure 28, and is the reason why octilinearity has not been fully applied – the edges are unable to move into the desired octilinear angles as they are held in place by the standard spring-embedder forces. This is a major problem in schematic layout, where it is often desirable to strongly enforce a number of criteria; the result is that this method is generally unable to produce schematics of a comparable quality to hand-drawn maps.

2.3.3 Multi-criteria Hill Climber

Shortly after the force-directed approach by Hong et al., an alternative multi-criteria hill climbing approach was proposed in (Stott and Rodgers 2004). This search-based layout method, covered in Section 2.2.2.1, computes a layout by iteratively repositioning nodes to optimise an objective function. As mentioned previously, a major advantage of search-based methods using an objective function is the ease at which new criteria can be added, and therefore these methods are well-suited to metro map layout. Stott et al. devised a number of criteria to implement into their optimiser based upon examination of popular published metro maps. It should also be pointed out that this method was developed simultaneously and independently from the method by Hong et al., yet both

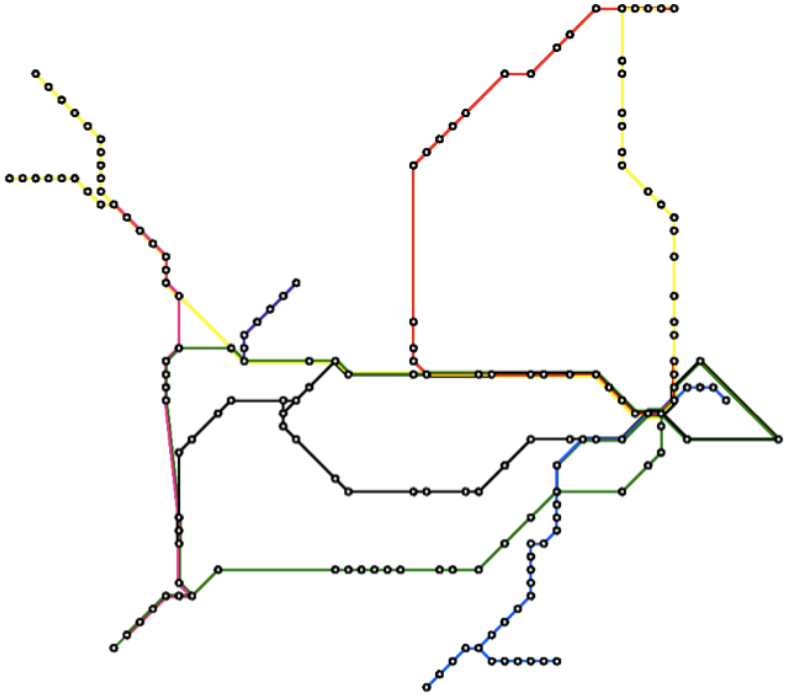


Figure 29: Sydney metro map produced by Stott et al. using their multi-criteria hill climbing method from (Stott and Rodgers 2004).

techniques share many of the same criteria gathered from examination of published maps. The criteria used in their layout algorithm are as follows:

1. Minimum edge crossings.
2. 4-gonality (edge angles should be multiples of 45°, now commonly referred to as octilinearity).
3. Equal edge lengths.
4. Angular resolution (maximise incident edge angles).
5. Line straightness.

As with the force-directed method, the major new criterion over standard graph layout is octilinearity – this being a very prominent feature in metro-maps.

Figure 29 shows the result of this method applied to the Sydney metro map. This map of Sydney differs from that used by Hong et al. as intercity lines

are excluded, shortening most periphery line sections; however, this difference should have little effect on the resulting layout. It is clear from this image that octilinearity has been enforced on almost all of the schematic, with only one edge not managing to achieve this (pink/green edge on the left side of the schematic). The method also uses a critereon for maximising the incident edge angles at nodes, and this has been effective at limiting sharp line bends such as those seen in the method by Hong et al. Clusters of nodes can be seen which give the impression of worse node distribution, but when compared to the force-directed method the distribution across the entire schematic is more uniform. Overall, this layout compares much more favourably to published maps drawn by hand.

Over a number of years, Stott et al. refined their method by adding the following layout criteria:

1. Ordering of parallel line edges should remain consistent through nodes.
2. Lengths of edges incident to a node should be kept equal.
3. Enforcement of relative position between nodes.

Along with these new criteria, node clustering methods were utilised to move multiple nodes at once in order to allow the method to escape from common situations of local optima, for example over-length edges separating two clusters of nodes. The authors also implemented a labelling algorithm into their optimiser. Figure 30 shows the same Sydney map optimised using the extended hill climbing method. This result, when compared to the previous attempt, shows improved station distribution and consistent edge ordering along with an improved overall appearance, in particular the left side of the schematic. However, the city centre (far right) has become congested and is difficult to interpret.

Along with the previously mentioned optimiser improvements, in (Stott et al. 2010) the authors present the results of a study conducted to evaluate maps drawn using their method in comparison to their official published version and an undistorted geographic map. The study used six different maps and 43 participants in order to find which map users were faster on and which they preferred to use for typical navigation tasks. They found that in most cases

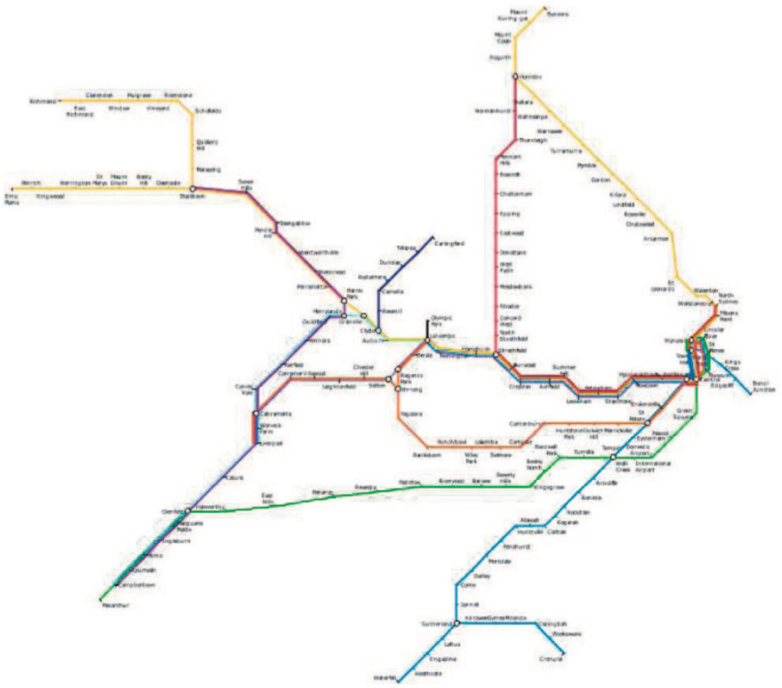


Figure 30: Sydney metro map produced by Stott et al. using their multi-criteria hill climbing method from (Stott et al. 2010).

a map drawn with their automated system was faster for finding an optimal route than both an undistorted map and the official published map. They also found that overall, users preferred the automated maps.

The advantages and disadvantages of this method are much the same as that of a multi-criteria hill climber for a standard graph layout. The method is capable of producing a good quality schematic at the expense of optimisation time and the likelihood of only finding a local optima configuration. As before, the use of an objective function allows for easy implementation of additional criteria.

2.3.4 Mixed-Integer Linear Programming

Nöllenburg and Wolff present a mixed-integer linear programming method for the optimisation of metro maps in (Nöllenburg and Wolff 2006). This method differs from standard search-based methods as it defines its constraints as one

of two types, “hard” and “soft”; hard constraints must be met, whilst soft constraints are maximised but not strongly enforced. Hard constraints are as follows:

1. Graph topology must be respected.
2. All edges must be octilinear.
3. Each edge has a minimum length.
4. Each edge has a minimum distance from each other non-incident edge.

Soft constraints are as follows:

1. Lines should have few bends.
2. Total edge length should be small.
3. Relative position of nodes must be preserved.

The MIP optimisation technique used is a derivative of linear programming – a method of determining the best outcome in a mathematical model based on a set of linear constraints (the hard constraints in this case) and a linear objective function. The linear constraints are plotted onto a graph to produce a feasible area, in which the linear objective function can then be minimised against the soft constraints.

Figure 31 shows the Sydney schematic optimised using this linear programming method. The result is of a high quality, meeting all defined hard constraints and showing signs of soft constraint optimisation. It is interesting to see just how similar this automated optimisation is to the officially published Sydney map (Figure 8), drawn by an experienced designer.

This method is advantageous in that its optimisation technique guarantees (if possible) that all hard constraints are met, whilst avoiding the problem of local optima; this allows the technique to generate schematic layouts of a very high quality. The cost of these advantages are a long-running optimisation time (the authors state Figure 31 took 22 minutes), and the possibility that it may not be able to produce a result if all hard constraints cannot be met.

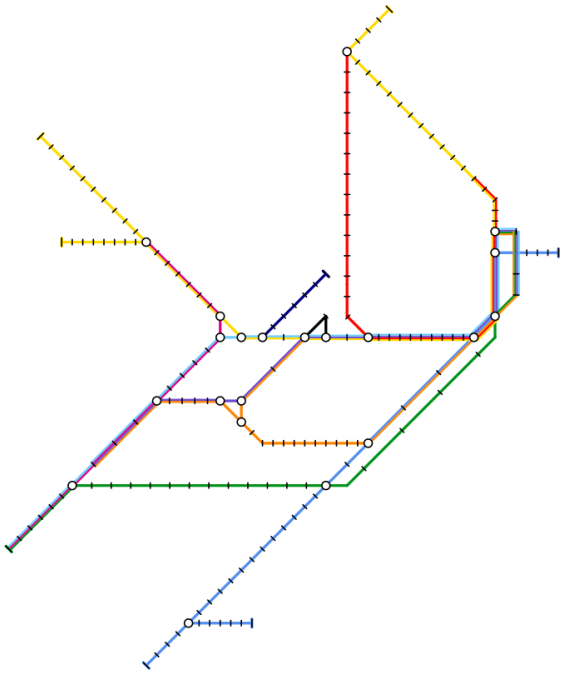
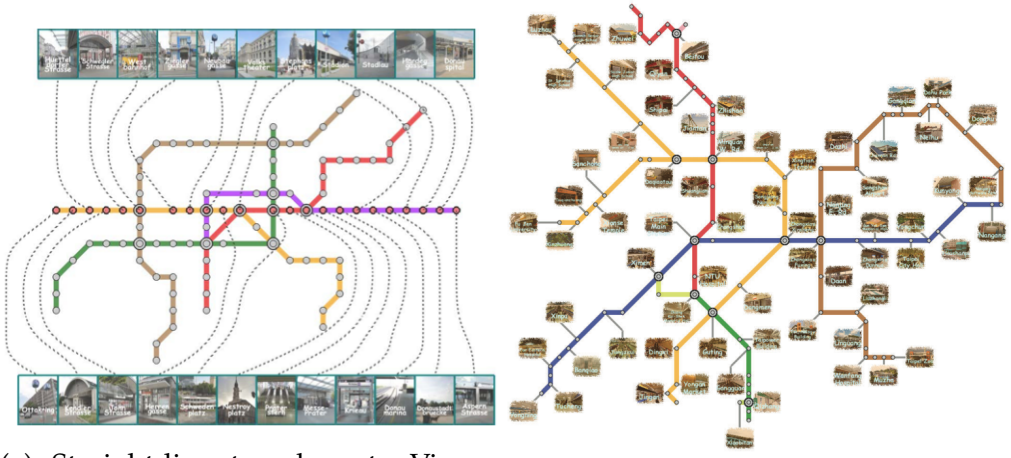


Figure 31: Sydney metro map produced by Nöllenburg et al. using their mixed-integer linear programming method from (Nöllenburg and Wolff 2006).



(a) Straight-line travel route Vienna metro map from (Wu et al. 2012).

(b) Annotated Taipei metro map from (Wu et al. 2013).

Figure 32: Two schematics produced by Wu et al. using their modified mixed-integer linear programming method for straight-line travel routes (32a) and annotations (32b).

More recently, Wu et al. have extended the mixed-integer linear programming method in two ways. Firstly, in (Wu et al. 2012), they adapt the method to form a straight, centred line based on a specified travel route with annotations around the edge of the layout (Figure 32a). Subsequently, they placed more emphasis on node annotation in (Wu et al. 2013). In this approach the authors apply additional constraints to generate schematics with enough room for station thumbnail images (Figure 32b) – such annotated maps are commonly seen in tourist guides. The technique is effective at allowing extra room for annotations whilst avoiding occlusions, however this comes at the cost of a reduction in quality of the schematic itself – many areas of Figure 32b can be seen with unnecessary line bends, in particular the red line at the top.

2.3.5 Simulated Annealing

The first implementation of a simulated annealing layout method for schematics was published in (Ware et al. 2006). The authors use this technique to produce schematic maps based upon road networks for mobile devices. The optimisation method remains the same as the approach described in Section 2.2.2.1, but the criteria used have been revised and are now defined as follows:

1. Topology should remain consistent.
2. Edges should lie in octilinear angles.
3. All edges should have a length greater than some minimum distance.
4. The angle between two incident edges should be greater than some minimum angle.
5. Edges should remain rotated as close as possible to their starting orientation.
6. The distance between disjoint nodes/edges should be greater than some minimum distance.
7. Nodes should remain close to their starting positions.

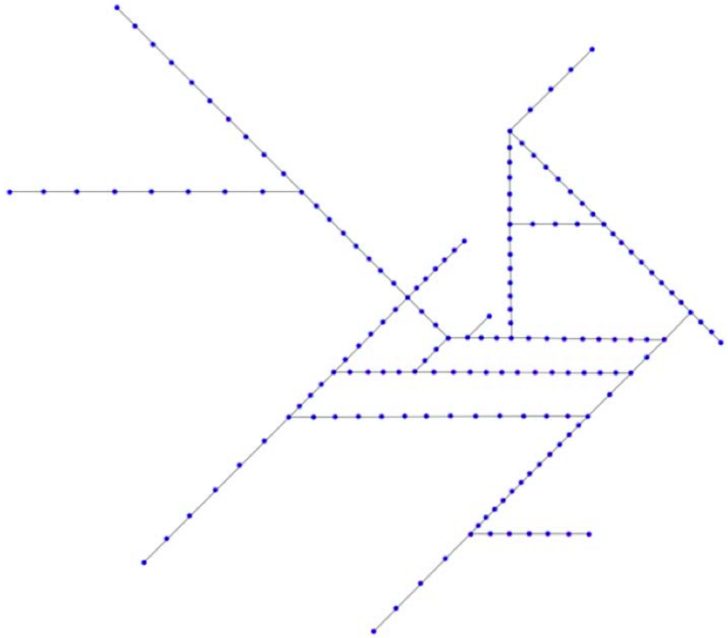
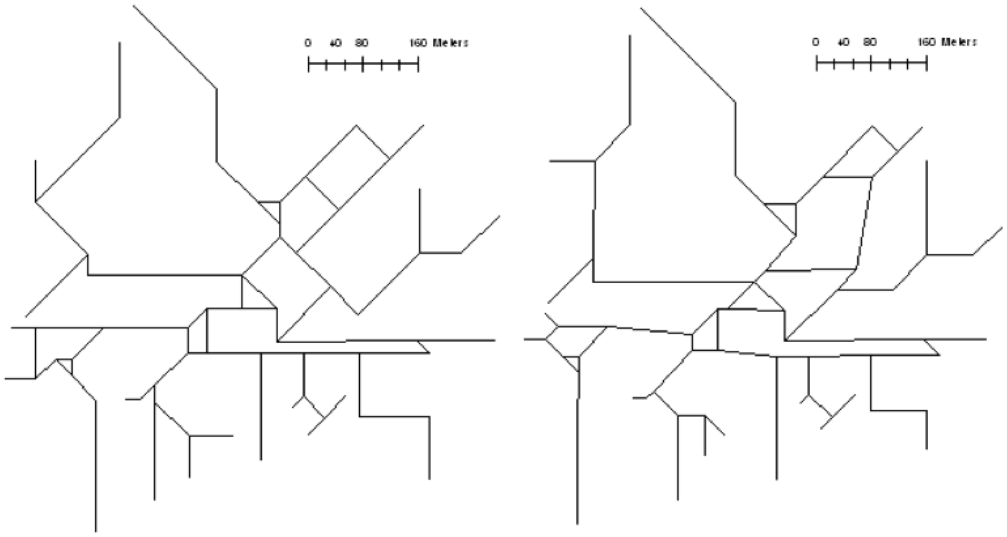


Figure 33: Sydney metro map produced by Ware et al. using their simulated annealing method in (Ware et al. 2006). Note: image from later paper (Ware and Richards 2013).

This method was later used by the authors to provide a comparison for another method in (Ware and Richards 2013). As part of the comparison, this SA approach was used to generate Figure 33, the Sydney metro. It should be pointed out, however, that the Sydney schematic used has been simplified more than that found in the alternate methods; many sections with a "triangle" of nodes have been condensed into a single junction, and the complicated city centre section of the schematic has been reduced into a set of simpler connections. However, the SA approach has produced a good layout exhibiting an evenly-distributed, fully octilinear schematic following the defined criteria. This SA approach is compared to a multi-criteria hill climber method in (Anand et al. 2007). The implemented method is a modified version of this SA approach, with negative movements disallowed. As one would expect, they found that the SA method has an advantage over a multi-criteria hill climber in its ability to escape local optima, and this is seen when comparing the results – for example SA showing a higher level of line straightness as seen in Figure 34. The



(a) Highest-cost (worst) simulated annealing result (cost=115).

(b) Lowest-cost (best) multi-criteria hill climbing result (cost=132).

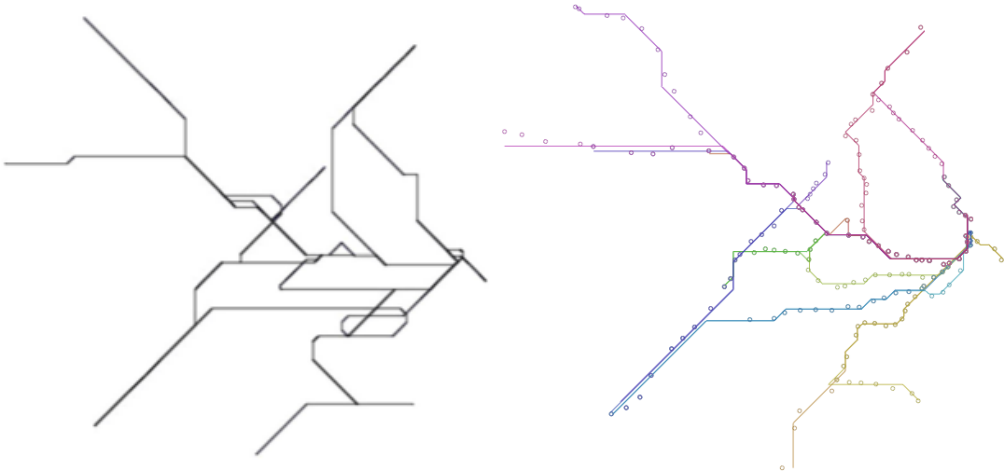
Figure 34: Comparison between the highest-cost simulated annealing (34a) and lowest-cost gradient descent (34b) approaches to show variation. From (Anand et al. 2007).

increased quality from an SA method is fairly substantial, with the worst SA result still beating the best multi-criteria hill climbing result in terms of the objective function. However, optimisation time is increased and non determinism introduced.

2.3.6 Path Simplification

Two path simplification methods for metro map layout were proposed. The first (Merrick and Gudmundsson 2007) extends previous work in path simplification (Douglas and Peucker 1973) and applies additional constraints to allow only specific edge angles. The second (Dwyer, Hurst and Merrick 2008) simplifies the first algorithm developed by Merrick et al., and improves its performance.

These methods are capable of producing simplified metro schematics as shown in Figure 35, but layout quality is heavily compromised. It can be seen that although all edges conform to the restricted angles, other criteria commonly found in alternate algorithms are not followed – this is because these



(a) Sydney metro (Merrick and Gudmundsson 2007).

(b) Sydney metro (Dwyer, Hurst and Merrick 2008).

Figure 35: Sydney metro map produced by Merrick and Gudmundsson (35a, (Merrick and Gudmundsson 2007)) and Dwyer et al. (35b, (Dwyer, Hurst and Merrick 2008)) using path simplification. Note: Figure 35b has been flipped vertically because it was upside-down.

criteria are not implemented into the algorithm, but as a result it leaves the schematic with many line bends and some dense sections.

The advantage of these path simplification methods is the speed at which they run, with stated timings of 268ms and <10ms respectively for the Sydney schematic shown in Figure 35, which in specific situations could be more beneficial than the output quality. For example it is suggested that these methods could perform part of a preprocessing step to simplify the input for another method.

2.3.7 Focus+Context Least-Squares Conjugate Gradient

Wang and Chi present an automated metro map layout technique using a least-squares conjugate gradient optimisation method in (Wang and Chi 2011). They present their method in terms of a focus+context method, although it can also be used to produce a layout for the entire system. Focus+context visualisation

(a) Standard Sydney layout.

(b) Sydney layout with focused route.

Figure 36: Sydney metro maps produced by Wang and Chi using their focus+context least-squares conjugate gradient method.

techniques magnify focal regions whilst maintaining visibility of contextual regions, for example like a fisheye lens, and are most commonly used in the exploration of large data sets on limited screen space. Their system determines the best route between two user-selected vertices using a shortest path algorithm. The edges contained within this route are then emphasised during the layout process to apply focus, whilst other edges are shortened and de-emphasised to provide context.

Their algorithm consists of two discrete steps; initially, they compute a smooth deformation in which vertices are moved into positions which result in smooth angles between edges, they then rotate edges to octilinear directions. The motive for splitting octilinearity into a separate step is that the octilinear direction is a discrete property and integrating this constraint into a continuous objective function would make global optimisation more challenging. Much like other search-based methods, the conjugate gradient method (Hestenes and Stiefel 1952) minimises an objective function defined by a number of criteria. To speed up optimisation, the objective function is solved in a least-squares sense, which means constraints are only approximated and not fully satisfied. Wang and Chi use fewer criteria than typically seen in metro map optimisation, these are:

1. Regular edge lengths (Smooth deformation).
2. Maximal angles of incident edges (Smooth deformation).
3. Positional constraints – nodes are very slightly attracted to their original positions to provide basic geographical guidance (Smooth deformation).
4. Octilinearity (Route octilinearity).

Figure 36 shows two examples of Wang and Chi's layout method – 36a shows the standard optimisation of the Sydney metro map, and 36b shows the same Sydney schematic with a focused route. Both layouts are aesthetically pleasing, and satisfy the implemented constraints. Wang and Chi quote a time of 0.816s for optimisation of this schematic, which is very fast for a search-based method, and considered fast enough that the flow of thought is uninterrupted for an interacting user (Nielsen 1993). As such this is the first metro map layout method, capable of producing high quality results, suitable for interactive applications.

2.3.8 Ant Colony System

Ware and Richards present an ant colony system algorithm for automatically schematising network data in (Ware and Richards 2013). Their ant colony system takes a similar approach to a multi-criteria hill climber, using the same criteria as found in their simulated annealing approach discussed previously (Section 2.3.5). Non-determinism is introduced into the node-positioning stage by moving nodes in a random order; as opposed to the same sequential order each iteration. During each iteration, the current configuration is modified multiple times, with each new layout being known as an ant. At the end of each iteration the best resulting layout applies pheromones to its node positions in a global grid so as to affect all ants in future iterations. The pheromone increases the likelihood of nodes moving near to it during the node-positioning stage, and gets stronger over time in node positions that consistently produce a good layout (as multiple ants will lay pheromones in the same position, creating a cumulative high pheromone value). As the pheromone gets stronger in node

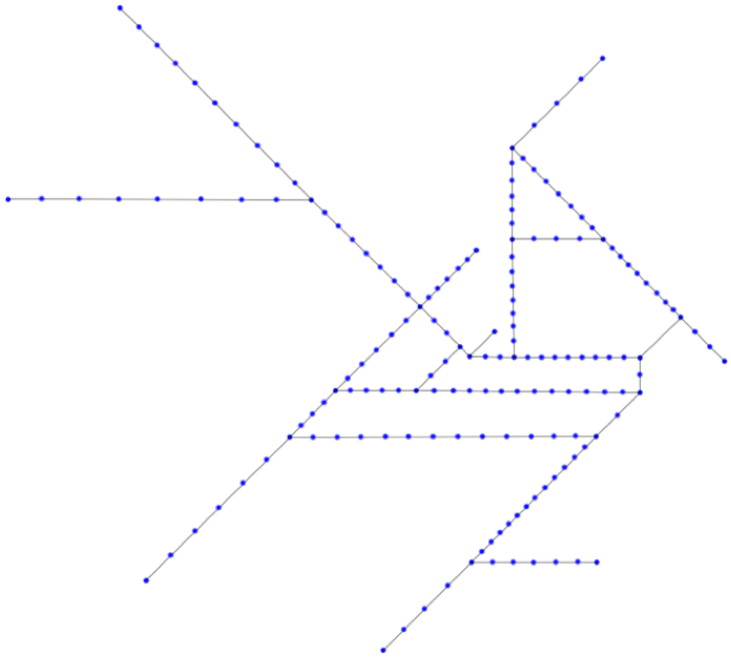


Figure 37: Sydney metro map produced by Ware and Richards using their ant colony system algorithm from (Ware and Richards 2013).

positions that produce a good layout, the algorithm converges to a single solution.

Layouts produced by this method (Figure 37 – Sydney), much like the SA method, appear cleaner than a standard hill climbing approach with the entire schematic fully enforcing octilinearity and, where possible, a consistent length between adjacent stations; this is partly due to the fact this method can avoid some local optima by the possibility of moving to a worse layout. There are still sections of this layout that show signs of unoptimal layout, including the top right line which comes off at an angle not expected from the incoming two lines. The authors conclude that *"in each case the ACS algorithm outperformed that of a previous SA algorithm in terms of cost (quality) and time"*. The running time is quoted as 150 seconds for the Sydney map shown, which is comparable to search-based methods.

2.3.9 Curved Metro Map Layout

Some of the latest work on the automated layout of metro maps has begun looking at metro layouts with alternate criteria, in particular curved metro lines are used instead of the more traditional octilinearly-aligned edges. This work builds upon the previous curvilinear graph drawing research discussed in Section 2.2.2 for standard graphs.

In (Roberts et al. 2013), the authors suggest that there are circumstances in which the conventional schematic map layout criteria fail to yield benefits, and performed a study to investigate the use of an all-curved design for the Paris metro map. The authors found that journey planning time for the all-curved map was up to 50% better than when using the official RATP octilinear map. Roberts then also worked with researchers in automated schematic layout to develop a force-directed layout method capable of producing curvilinear metro maps using Bèzier curves in (Fink et al. 2013). This paper concisely states the premise of, and problem with, octilinear metro map designs in the following paragraph (page 1):

> *Such* [octilinear] *schematic maps potentially offer usability benefits by simplifying line trajectories, and hence reducing the amount of information that is irrelevant for deciding how to travel from one station to another. However, there is often a misbelief that it is merely the use of straight lines and a restricted angle set that benefits the user, and as a consequence many human designers fail to optimize octilinear maps, converting chaotic real-life line trajectories into complex sequences of short straight-line segments and bends (Roberts 2012). In other instances, the network structure itself makes the benefits of octilinearity difficult to realize.*

For this reason the authors suggest the use of curvilinear designs for situations where octilinear layout techniques fail to produce high-quality maps, both by automation and manual design, such as in dense interconnected networks with many bends.

The authors apply the following criteria to the Bèzier curves:

1. Any pair of Bèzier curves that are consecutive on a metro line must meet in a station and must have the same tangent there.

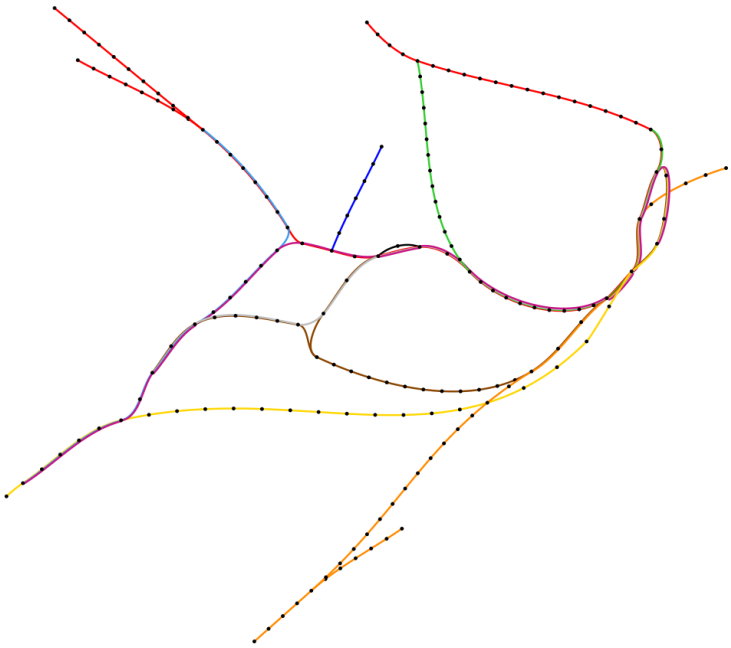


Figure 38: Curvilinear Sydney metro map produced by Fink et al. using their force-directed Bèzier curve method from (Fink et al. 2013).

2. The aim for each individual metro line is to consist of the smallest number of Bèzier curves necessary in order to maintain interchanges.

3. Points of inflection should be avoided (change in curve direction).

Previous attempts such as (Finkel and Tamassia 2005) (Section 2.2.2) at drawing curvilinear graphs with a force-directed layout method treat vertices as control points, and subject these to standard straight-edge layout forces. This new method implements additional forces which modify the curves by handling vertices and control points in different ways in order to meet the defined criteria. Vertices are influenced by standard force-directed placement forces as defined in (Fruchterman and Reingold 1991), and forces acting upon control points are applied as a rotation around the vertex towards the tangent of the colinear edge. This has the effect of straightening edges as much as possible, whilst ensuring consecutive line sections share the same tangent through vertices. Forces are also applied to line tangents at vertices to improve angular resolution.

Figure 38 shows the Sydney schematic as optimised using this curvilinear

force-directed method by Fink et al. The authors use octilinear produced optimisations from the MIP approach of (Nöllenburg and Wolff 2006) as the input schematic; geographic input configurations were also tested, but they state that superior results were obtained from octilinear inputs. This layout technique has produced an aesthetically pleasing rendition of the Sydney layout, and the constructed Bèzier curves follow all predefined criteria. There are no appearances of sharp edges in lines caused by mismatched tangents through vertices, and there are minimal points of inflection. Arguably the most difficult section of Sydney to optimise for all algorithms is the city centre (far-right), and this method struggles here with an unnecessary crossing below the circular section (the orange line crossing out of the circle is unavoidable), caused by the entire section angling upwards as opposed to the more horizontal layout seen in other methods. Periphery line section splits also remain very close (top-left red line), presumably caused by both sections attempting to align to the tangent of the section before the split whilst remaining straight.

This method is a very good adaptation of curvilinear graph drawing methods into the area of schematic layout, and the authors have outlined potential improvements including labelling mechanisms. As mentioned previously, the topology of certain schematics does not adapt well to octilinear configurations, and as such this method has useful applications. Limitations include the use of an octilinear input configuration meaning that the schematic must be optimised beforehand, increasing the total time taken to produce a schematic. This can be seen as a disadvantage depending on the application requirements; however, with layout methods becoming increasingly fast at producing good quality octilinear configurations, the issue can be mostly mitigated.

2.4 Summary

Throughout this chapter we have covered the areas of schematic mapping, graph drawing, and automated schematic layout. Existing research in each of these fields has been explored, along with a look at examples of current officially used designs and those using the metro map metaphor for non-transport data.

We have talked about current design processes, and explained how an automated layout system would provide benefits for both professional and amateur schematic designers in the hope of increasing the popularity of this type of visualisation.

Chapter 3

Improving the User Interface

Initial research into the creation of metro map style schematics revealed few applications designed to aid users during the design process. Applications which claim to do so such as (EdrawSoft 2014) (iMapBuilder 2014) turn out to be more general purpose vector based design applications and do not allow easy modification of drawn schematics. This distinction is important, as users should ideally be able to easily modify the schematic layout after the structure itself has been created. For example, moving nodes should also update adjacent edges to remain connected. Lack of such connectivity between node and edge objects makes this task laborious to perform, hindering the user during layout.

CAD/GIS packages such as (AutoCAD 2015) (ArcGIS 2015) do support schematics with connections between objects, however many important features specific to metro maps are not supported, such as the common requirement of multiple lines running parallel to each other with different colours, or schematic layout techniques tailored specifically to the metro map style.

The number of problems present in the creation of metro maps is further compounded by a lack of support for automated layout. As discussed in the previous Chapter, automated metro map layout methods have been the subject of much research, but have only been implemented in a proof-of-concept context.

Taking these two points into consideration, our initial work focused on the development of an application to facilitate the fast manual drawing of metro

Figure 39: SchemaSketch running on an ASUS Eee Pad Transformer device using Android v3.1.

map style schematics whilst recording connectivity information between objects to allow easy layout modifications once the structure has been input. Using the same connections, it is also possible to perform automated layout techniques upon the drawn schematic – this extension is covered in Chapter 4.

The timing of this work coincided with a great increase in both the availability and popularity of portable touch-sensitive devices such as mobile phones and tablets, and with them a large number of applications designed for touch interactivity. We were unable to find a gesture-based input system for the creation of schematics, and were interested in exploring their suitability for fast and accurate input of schematics into software. We therefore decided to implement our application, SchemaSketch, for Google's Android operating system found on a large number of such devices. SchemaSketch (Figure 39) has been primarily developed for v3.0 but is compatible with newer releases and will accommodate a variety of screen sizes. The application (.apk) can be downloaded at the following URL: `http://cs.kent.ac.uk/projects/schemasketch`.

(a) Junction. (b) Bend Point. (c) Edge. (d) Station.

Figure 40: Schematic objects.

3.1 Interface

This section explains how SchemaSketch is used to create and edit metro map style schematics. There are two editing states for the input and modification of schematics, these are termed the "Draw" and "Move" modes respectively and can be toggled using a context menu. Draw mode allows the use of sketch gestures for the creation and connection of various schematic objects and is explained further in Section 3.1.1. Move mode allows the user to reposition objects along with object deletion and load/save functionality, and is explained further in Section 3.1.2.

3.1.1 Draw Mode

This mode allows schematics to be created by using gestures to input objects (see Section 3.2 for details on gesture recognition). The following list describes the objects that can be constructed:

- *Junction* – Junctions are circular stations and are used when multiple edges join or diverge (Figure 40a).
- *BendPoint* – Bend points should be used when a bend is desired in a line. Similar to a junction, edges can split or converge at bend points. Ensuring bends in a line do not occur at either junctions or stations provides more aesthetically pleasing results in the schematic. SchemaSketch's optimisation process will attempt to automatically insert bend points where

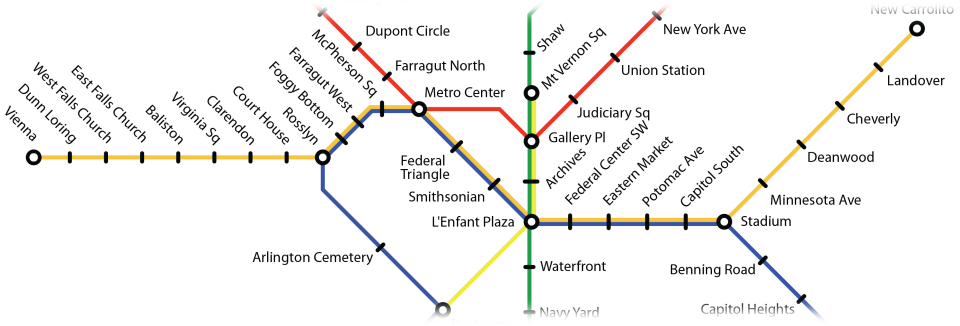


Figure 41: Section of the Washington metro map represented in SchemaSketch using all available objects.

required, however functionality for manual addition is also provided (Figure 40b).

- *Edge* – Provides a connection between a pair of junctions or bend points. SchemaSketch provides support for different coloured edges to be drawn. Multiple incident edges of the same colour will form a line, and extending an edge will add the new edge onto the existing line. Parallel edges (of different colours) can be drawn and will be automatically aligned parallel to each other (Figure 40c).

- *Station* – Stations can be added onto existing edges and will distribute themselves evenly along it. They will also align to be perpendicular to the parent edge. Stations can be added to both single and parallel edge segments (Figure 40d).

The draw mode also allows the user to add labelling to the schematic, see Section 3.4 for details. Figure 41 shows a section of the Washington metro map represented in SchemaSketch using all available objects.

3.1.2 Move Mode

This mode allows the manual modification of created schematics by enabling drag and drop functionality for all objects. Objects retain their connections during movement, and this functionality allows faster and easier manual modification of schematics than using an application which does not support this by

default, such as many vector graphics applications.

Whilst in move mode, touching a blank section of the drawing area will bring up additional operations. These include the ability to manually snap all nodes to grid intersections, show or hide labels, mark parallels and auto-align nodes. The mark parallels mode allows users to select a number of edges that they wish to be parallel in the final layout; the automated layout method then uses additional criteria to ensure these edges are kept parallel. The auto-alignment of nodes is intended as a post-processing feature and allows users to select a number of nodes, then choose to align these vertically or horizontally. Nodes are moved along the angle of their parent edge so as to not break octilinearity after optimisation. This feature is mainly intended for the alignment of periphery nodes to improve the appearance of the schematic. The move mode also allows users to manually position labels on the schematic, see Section 3.4 for details.

3.1.3 Contextual Menu

In both draw and move modes, a contextual menu allows a number of additional options. These are:

- *Colour* – Change the colour of the next edge to be drawn.
- *Eraser* – Changes the pen to an eraser pen that will remove everything drawn over.
- *Clear All* – Clears all objects from the screen (requires confirmation).
- *Optimise* – Uses a multi-criteria hill climbing optimiser to rearrange nodes into a more optimised schematic (see Chapter 4).
- *Load/Save* – Allows the loading and saving of drawn schematics to a file.
- *Mode: Move/Mode: Draw* – Switch between move and draw modes.

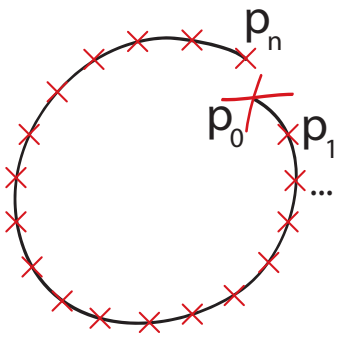


Figure 42: Gesture capture – Red crosses indicate time-sampled points along the drawn gesture.

3.2 Gestures

We have examined previous work in gesture-based input (Rubine 1991), sketch recognition and beautification of hand drawn sketches (Freeman and Plimmer 2007) (Kara and Stahovich 2004) (Gusaite 2006) in order to decide on an efficient and intuitive input mechanism for drawing schematics. For the purpose of our application, only a small fixed set of gestures need to be supported. The application does also not need to provide a method for learning new gestures. To meet these requirements, we chose to use a number of metrics to identify drawn gesture shapes rather than use a learning algorithm. Figure 43 illustrates the supported gestures to input the various objects defined in Section 3.1.1.

Although full sketch recognition can provide more advanced functionality than gestures by supporting multi-stroke symbols, it comes with a performance overhead in recognition as well as a mechanism to determine when the user has finished one symbol and moved onto the next. A common mechanism to implement this is the use of a waiting time between pen strokes (Gusaite 2006) but this hinders the input flow of the user. Simpler gesture recognition, where a single-stroke gesture corresponds to an object, can provide the required functionality whilst ensuring the user is not disturbed by workflow pauses.

Gestures are recorded as a sequence of time-stamped coordinates. As a gesture is drawn, SchemaSketch will record the current position of the touch motion ten times a second. Figure 42 shows a visual example of this sampling process upon a junction gesture. Red crosses indicate sampled coordinates, starting

from p_0 (the initial point of contact) up to the final point p_n. Although some accuracy of the gesture is lost, providing a high enough sample rate is used there is still enough information to determine the drawn gesture. Using these sample coordinates, SchemaSketch will attempt to recognise the gesture based on the metrics explained in the following subsections.

3.2.1 Minimum direct length to be classified as an edge

Direct length refers to the absolute distance between the start and end coordinates of the gesture. For a gesture to be an Edge object, this distance must be ≥ 45 pixels.

3.2.2 Minimum straightness to be classified as an edge

The straightness of the gesture G is measured using Equation 7.

$$straightness(G) = \frac{dist\,(G_{start}, G_{end})}{actualLength(G)} \tag{7}$$

where $actualLength(G)$ is calculated using Equation 8.

The straightness calculation will produce a value between 0 and 1. A value of 1 is a perfectly straight line. For a gesture to be classed as an edge, $straightness(G)$ must be ≥ 0.9. If a gesture passes the minimum direct length test and minimum straightness test, it can be classified as an edge, otherwise it is potentially either a junction, station, or bend point. Differentiating between these last three is performed by the following four features.

3.2.3 Minimum actual length to be classified as a station

Actual length refers to the length of the gesture if it was straightened out, and is calculated using Equation 8, where n is the number of points in the gesture and p_i is the i^{th} sample point along the gesture.

$$actualLength(G) = \sum_{i=0}^{n-1} dist\,(p_i, p_{i+1}) \tag{8}$$

$actualLength(G)$ must be $\geq$ 10 pixels for the gesture to be evaluated. This means any gesture shorter than 10 pixels will not be recognised and nothing will be added to the schematic. This is useful for discarding unintentional screen touches.

3.2.4 Minimum straightness to be classified as a station

The straightness is once again checked using Equation 7 and if $straightness(G) \geq 0.5$ then it will be classified as a station. Although stations and edges are both straight line gestures, edges require a higher $straightness(G)$ value because the longer a gesture, the easier it is to obtain a high $straightness(G)$ value.

3.2.5 Minimum number of sharp bends to be classified as a bend point

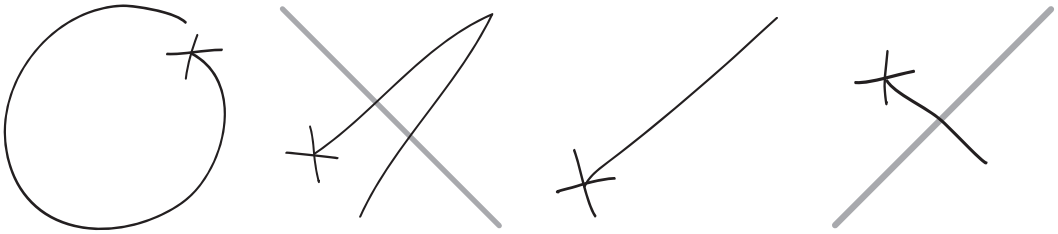
The number of sharp bends along the length of the gesture is measured using Equation 9. If $sharpBends(G) \geq 1$ then the gesture is classified as a bend point. A sharp bend is defined as a difference in edge angle that is $\geq 100^{\circ}$ between any two adjacent edges in the gesture.

$$sharpBends(G) = \sum_{i=0}^{n-2} \begin{cases} 1 & \text{if } angle(p_i, p_{i+1}, p_{i+2}) \geq 100 \\ 0 & \text{else} \end{cases} \tag{9}$$

3.2.6 Minimum average radius to be classified as a junction

If there are no sharp bends, this last check is performed to identify a junction gesture. We calculate the average radius of the shape (we know the shape is curved, as $straightness(G)$ is low). First we calculate the centre point of the gesture, by averaging x and y co-ordinates across all points. We can then calculate the average radius using Equation 10, where n is the number of points in the gesture and p_i is the i^{th} point along the gesture.

$$radius(G) = \frac{\sum_{i=0}^{n} dist\ (G_{centre}, p_i)}{n} \tag{10}$$



(a) Junction. (b) Bend. (c) Edge. (d) Station.

Figure 43: Schematic object gestures. Gestures start at the $\times$.

If $radius(G) \geq 10$ pixels, this gesture can now be classified as a junction, otherwise the gesture will not be recognised and nothing will be added to the schematic.

These rules result in junctions being drawn by a circular shape with start and end points close together (Figure 43a); bend points by a line across an edge with at least one sharp bend (Figure 43b); edges by a long, straight gesture (Figure 43c); and stations by a short, straight gesture across an edge (Figure 43d). Figure 44 illustrates the gesture interpretation process in a flow chart.

Although no studies were performed to objectively evaluate the effectiveness of our input mechanism, during extensive personal use it has proven to be an effective method for the construction of metro map schematics. Gestures are interpreted accurately and allow quick and easy schematic creation, providing a solid proof of concept for gesture-based schematic construction.

There are a number of accessibility issues associated with touch and gesture-based input systems, for example for individuals with visual or dexterity disabilities. Our current implementation system does not provide additional support for these users, however we have ensured that the gestures used for creating schematics are as simple and intuitive as possible. There are also no time limitations on gesture input, meaning the system will read gestures with identical accuracy when drawn at any speed. As mentioned later in this chapter, implementation difficulties prevented the addition of zoom and scroll functionality, however this is another feature that could greatly further aid users who may find it difficult to use the gesture input system accurately.

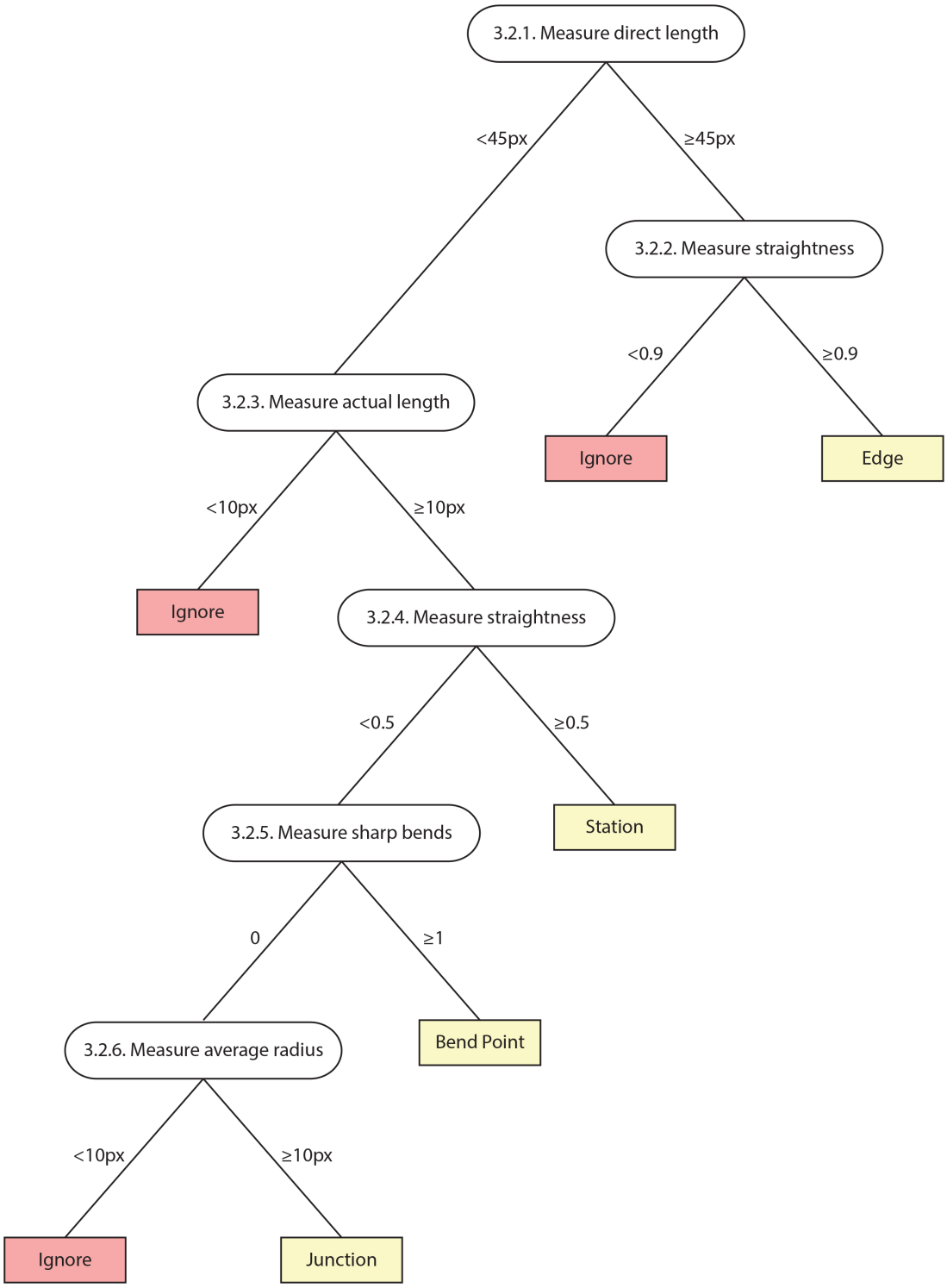


Figure 44: Gesture interpretation flow diagram.

3.3 Connections

SchemaSketch will automatically create graph connections between drawn objects based on location and how the gesture is drawn. Connections between objects in the schematic are essential to allow modification without breaking the structure. Whilst the user draws their schematic and connections are added, SchemaSketch maintains a graph structure representing the visualised schematic. This underlying graph structure allows automated layout techniques to be applied to the schematic as explained in the next chapter. Below is a explanation of how each gesture can be used to connect objects in the schematic.

- *Junction*

  ▷ Drawing a junction encompassing one or more edge ends will connect the new junction to these ends.

  ▷ Drawing a junction encompassing a bend point will replace the enclosed bend point with a new junction.

- *Bend Point*

  ▷ Drawing a bend point crossing an edge will insert it at the drawn location – this can also be used to insert a junction along an edge, as the inserted bend point can then be replaced by a junction.

- *Edge*

  ▷ An edge can extend an existing edge by starting or ending the edge gesture at another edge start or end.

  ▷ Edges can be connected to junctions and bend points by starting or ending the edge gesture close to them.

- *Station*

  ▷ Drawing a station either crossing or close to an edge will insert it along that edge. Existing stations along the edge will be redistributed equidistantly.

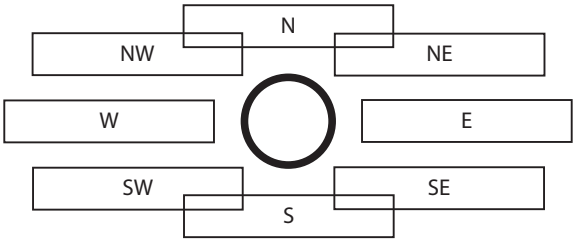


Figure 45: Possible label positions relative to parent node.

3.4 Labelling

Labels can be added to junctions and stations whilst in draw mode. Touching on a junction or station will open a text input dialog allowing the user to enter a label name. Whilst in move mode, labels can be manually relocated by dragging them around their parent. There are eight positions in which a label can be placed, these positions relative to the parent are *North*, *North-East*, *East*, *South-East*, *South*, *South-West*, *West* and *North-West*, as shown in Figure 45. A label will initially be placed in the *South* position.

Along with these positions, during the automated layout process labels are rotated to one of four angles in order to help avoid occluding other parts of the diagram. This is further explained in Section 4.6.

3.5 Summary

This chapter has covered the user interface of our metro map creation application, SchemaSketch, along with details of how we recognise gestures to construct schematic objects and connections.

Drawbacks of the application include the limited screen space available on mobile devices. Although techniques such as zoom and scroll can be used, SchemaSketch does not currently support this functionality and therefore has a limited drawing area. Another problem we contended with was the relatively low performance of these devices, with many struggling to maintain a smooth interface whilst viewing complicated schematics. This last point was also a hinderance to the implementation of features to remove drawing area restrictions

and allow larger schematics. These two drawbacks ultimately arose from a lack of processing power in early devices, but performance has vastly increased in the last few years and it may now be possible to develop software which allows much larger schematics to be drawn utilising zoom and scroll functionality.

Although there were a number of disadvantages associated with the hardware, personal use shows the gesture-based input system to be very effective and allows much faster creation of schematics when compared to a more traditional approach of selecting objects from a toolbox and placing them. It would therefore be a very useful addition to any schematic creation software, and could also be used on non-touch enabled devices with the aid of a graphics tablet and pen.

Following on from this work in creating an effective input system for constructing metro map style schematics, we chose to implement an automated layout method into our software. This layout algorithm is able to modify the drawn schematic by manipulation of the generated underlying graph structure. The following chapter details our chosen method, and illustrates how automated techniques can be used to vastly improve a number of schematic characteristics and aid designers in the creation of effective schematic layouts.

Chapter 4

Multi-criteria Hill Climbing Optimiser

Along with our gesture based input system described in the previous chapter, we implemented an automated layout algorithm into SchemaSketch. The purpose of such algorithms is to aid users in the layout of their planned schematics – typically a time consuming and difficult task. To do so, our algorithm follows a number of criteria in order to make informed node movement decisions, as described throughout this chapter.

The layout technique chosen falls under the category of a multi-criteria hill climbing optimiser, and takes inspiration from previous work in (Stott et al. 2010) (Section 2.3.3). The reasons for choosing this particular method to extend were as follows: 1) The hill climbing method is capable of producing decent results which have been evaluated and shown to be effective in user studies against geographic and published schematics. 2) There is still obvious room for aesthetic improvement and additional criteria when comparing produced layouts to published schematics. 3) The method is very extensible, allowing us to implement new experimental criteria rapidly. 4) The method operates in a more reasonable time-frame than alternate methods including SA and linear programming which can take many hours to run. 5) The result of a hill climber is often inferior to that of an SA method, as it has no way to escape local optima. However, the benefit of this is that it produces a deterministic result making it

easier to see the effects a specific new criterion and/or criterion weight has upon the final layout.

4.1 Overview

An abstraction of the operation of our layout method allows us to divide it into three main operations and their most important sub-functions, as listed below. The ordering of this list also corresponds to the order of execution of our method (note: the main layout process is iterative). References are provided to related sections which cover each of these operations in further detail.

1. Pre-processing (Section 4.2).
    1.1. Swap 2-degree junctions for stations (Section 4.2.1).
    1.2. Add bends proportional to station counts (Section 4.2.2).
    1.3. Identify periphery line sections (Section 4.2.3).
    1.4. Align nodes to grid (Section 4.2.4).
2. Main layout process (Iterative) (Section 4.3).
    2.1. Single Node Movement (Section 4.3.2).
    2.2. Node Cluster Movement (Section 4.3.3).
    2.3. Mid-processing (Section 4.3.4).
3. Label layout process (Section 4.6).

4.1.1 Extensions & Modifications

The following list identifies the main extensions and modifications that we have made to the existing method (Stott et al. 2010) upon which it is based:

1. We have introduced a new schematic object, bend points. Previously, line bends in the schematic were only possible at junctions/stations. The result of this contributed to a less aesthetically pleasing layout than published versions and could be harder to follow, for example when multiple lines

enter a junction and exit at different angles. Our new bend point objects can be inserted along edges allowing the optimiser to move line bends away from junctions. This change is combined with a new criterion (explained below) to enforce the straightness of lines through junctions, and greatly helps increase the angular resolution of nodes. As well as manual addition of bend points, our algorithm automatically introduces and removes bend points where necessary to help produce a layout which more closely resembles a published schematic.

2. The method by Stott et al. operates by moving both junctions and stations. This creates a very large number of objects for which positions must be evaluated – the main contributor of slow performance. A common technique to reduce this number and increase performance is to replace strings of 2-degree vertices with a single edge, weighted to account for the number of vertices upon it; however this does not produce satisfactory results, as it does not allow line-bending along edges between junctions. With the introduction of bend points, we can use a half-way solution capable of both faster performance and allowing bends within lines when required. Stations are now not moved by the layout method, and lines cannot bend at them; instead we move only junctions and bend points. Considering typical mid-sized metro schematics often contain upwards of 100 stations, this change considerably improves the performance of the method as it significantly reduces the number of objects requiring evaluation.

3. We have reduced the number of potential positions to check when moving nodes. Previously, all grid intersections within a certain distance of the original node position were regarded as potential positions and evaluated. Now, we only evaluate eight positions for each level of distance (in grid steps) from the node. This is a relatively small modification, but has a noticeable effect on the performance of the method by further reducing the number of position evaluations performed, without any visual detriment in layout quality. This modification also allows the distance nodes can be moved to be increased whilst only incurring linear time complexity growth; each additional grid unit of distance only requires eight more

evaluations, rather than the exponential growth resulting from evaluation of all new positions as the distance increases.

4. The method by Stott et al. includes clustering methods designed to improve specific situations in the layout of schematics that the optimiser has difficulty with when using only single node optimisation. These clustering techniques operate by grouping a number of nodes together and moving them as one, as explained later in this Chapter. We have implemented a new node clustering method – periphery clustering. This solves a common situation of suboptimal layout where kinks in periphery line sections can not be removed during layout, affecting metro maps in particular. Our new technique effectively straightens out these periphery line sections resulting in an improved aesthetic appearance of the schematic.

5. We have conceived and implemented a number of new criteria, previously not considered in any metro-map layout method, to perform the following functions:

   ▷ Enhance the straightness of lines through junctions. With the introduction of bend point objects, this criterion helps shift line bends away from junctions and onto bend points.

   ▷ Enhance the straightness of periphery line sections. Previously, all lines are subject to a line straightness criterion; we extend this by implementing an additional straightness criterion affecting only periphery line sections which have no need for bends.

   ▷ Enforced parallel lines. SchemaSketch provides functionality for the user to identify multiple edges to be aligned in parallel – a new criterion then enforces this by penalising the edges if they are misaligned. This can be used to great effect on many metro maps in which the user desires specific sections to be aligned.

   ▷ Enhance the balance of the schematic. This balance criterion aims to provide more even node distribution across the entire schematic. This criterion helps both with the expansion of dense centre sections and condensation of sparse periphery sections of schematics.

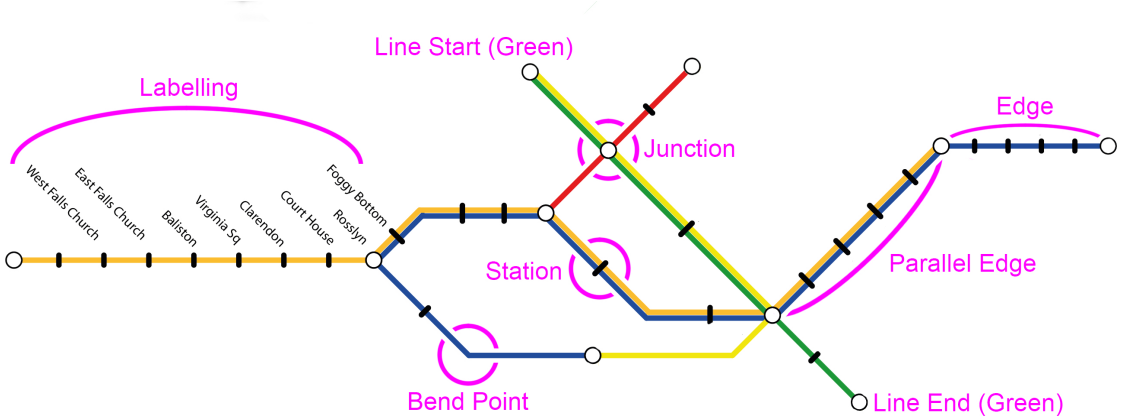


Figure 46: Section of a metro map highlighting the various objects that make up a typical schematic diagram.

6. Criteria weightings have been recalculated to account for the varying differences in the layout method, as well as the need to be balanced with new criteria.

7. The previous implementation of labelling only supports horizontal orientation of text. We have extended functionality in this area to allow four possible angles for labels to be oriented. These additional angles provide an increase from 8 to 24 positions per label, greatly increasing placement flexibility. Our label placement method also performs an extra step in which labels are grouped with neighbours and positioned simultaneously. This greatly enhances label position consistency.

4.1.2 Definitions

Many of these definitions have been previously covered and retain their meaning. Figure 46 provides a visual illustration of these objects.

- *Junction* – Circular stations which are used when multiple edges join or diverge.

- *Bend Point* – Allows a bend in the line. Similar to a junction, edges can join or diverge at bend points.

- *Node* – A parent group consisting of junctions and bend points. All edge objects start and finish at a node.
- *Edge* – Provides a straight connection between a pair of nodes. Edges are always coloured to indicate their line group.
- *Line* – Consists of multiple edge objects containing the same colour connected in sequence via nodes. It is important to make this distinction as many layout criteria are evaluated upon lines.
- *Parallel Edge* – Edges can contain multiple colours if they belong to many lines, in this case the colours are drawn in parallel. Note: a parallel edge between two nodes is represented in the graph structure as a single edge, rather than one for each colour.
- *Station* – A small perpendicular tick along an edge. Stations will distribute themselves equidistantly along their parent edge. Note: stations are **not** nodes and therefore lines cannot bend at them.
- *Label* – Allows text to be anchored to the parent object at a number of positions and orientations. Parent objects include both junctions and stations.
- *Desired Parallel* – The application allows users to designate edges to be desired parallels. The layout algorithm will then attempt to align all such edges to the same angle.

It is important to note that the layout process only operates on nodes; stations cannot be moved directly, as they are simply distributed along edges and move when either of their parents do.

4.2 Pre-processing

Pre-optimisation is performed once at the beginning of the optimisation process and is responsible for ensuring the schematic is prepared for layout. The pre-optimisation process performs a number of functions which are run in the following sequence and are covered in the listed subsections.

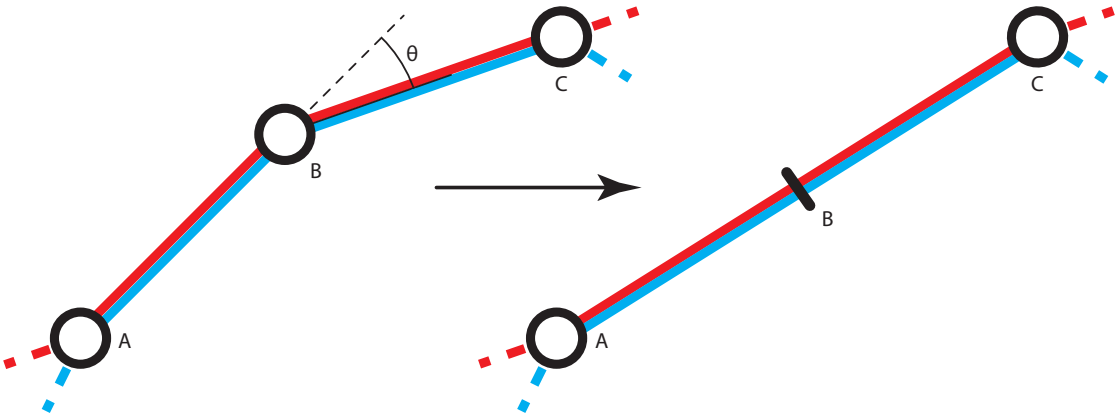


Figure 47: Junctions with a degree of 2 are replaced with stations to simplify the schematic, providing θ is less than some predetermined value (default $= 45°$).

1. Swap 2-degree junctions for stations (Section 4.2.1).
2. Add bends proportional to station counts (Section 4.2.2).
3. Identify periphery line sections (Section 4.2.3).
4. Align nodes to grid (Section 4.2.4).

4.2.1 Swap 2-degree junctions for stations

This initial step replaces all suitable 2-degree junctions with stations and is performed to simplify the schematic by reducing the number of nodes to be considered during the layout process. Stations cannot allow bending of lines and therefore it is possible that the resulting station will not be in the same position as the junction. Suitable junctions are those in which the incident edges do not form a sharp angle which could cause bad distortion when replaced. The exact angle at which this cutoff occurs is predefined in our algorithm; by default we use a value of 45°. An example of this process is illustrated in Figure 47.

4.2.2 Add bends proportional to station counts

As lines can only bend when passing through nodes rather than at stations, and as 2-degree junctions are replaced with stations during pre-processing, it is important to ensure that there are enough bend points to allow the required degree of flexibility to the lines during the layout process. We therefore automatically add a number of bend points along each edge dependent on both the length of the edge and the number of stations it accommodates.

For each three stations along an edge, one bend point is added. Additionally, for each four units of edge length, another bend point is added. The minimum number of bend points added to each edge is one, and the maximum is four. Equation 11 shows the calculation for the number of bend points based upon this specification.

$$bends = \left\lceil \left(\frac{stations}{3} \right) + \left(\frac{length}{4} \right) \right\rceil \leq 4 \tag{11}$$

We use a ceiling function when calculating the number of bend points to add – this allows the function to favour adding more bends and enforces the minimum of one bend point as $length$ is always > 0. The values we chose for this calculation were derived from personal testing and produce a suitable proportion of bend points for each edge in the schematic.

4.2.3 Identify periphery line sections

Certain criteria used during the layout process (Section 4.4) specifically target periphery line sections. We define periphery line sections as a sequence of edges starting from a termination node (1-degree), passing through any number of 2-degree nodes and ending at the first $>$ 2-degree node.

We use a simple computational process to detect these sections. Firstly, we identify all 1-degree nodes, and follow these along their parent line, recording all objects passed through, until we meet the first $>$ 2-degree node. Using these identified sequences of edges, we construct new line objects for each periphery section. These new line sections only exist within the underlying schematic structure and are used when calculating the periphery line straightness criterion

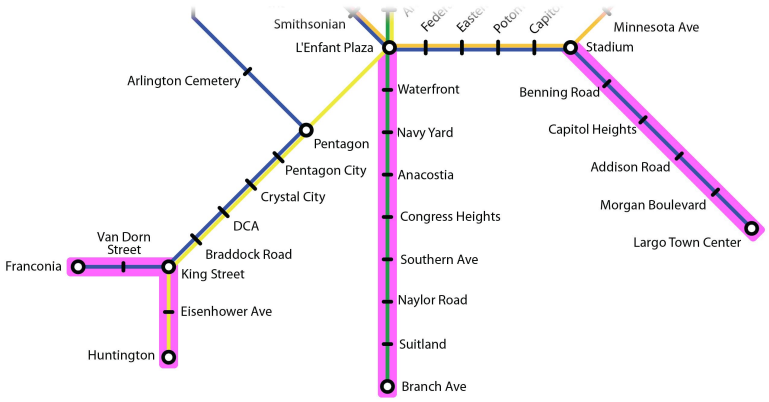


Figure 48: Periphery lines highlighted on a section of the Washington metro map.

(Section 4.4.5); they are not visible to users.

Figure 48 shows a section of the Washington metro map with periphery line sections highlighted with a pink background.

4.2.4 Align nodes to grid

Our optimisation process operates by examining the local neighbourhood of nodes in order to determine the most optimal position based upon a number of criteria. The schematic objects drawn by users are not aligned to a grid, and so the resolution of the local neighbourhood is 1 pixel. We overlay a grid with a default cell size of 10 pixels, and align all node objects onto this.

This is accomplished by examining each node in turn and moving it to the nearest grid position. If multiple nodes contest a grid position, the original position of each will be checked and the closest one moved; the remaining node will then align to the second nearest. Using such a grid has multiple advantages to simplify the optimisation process:

1. We can reduce the number of possible node positions that we are required to check, greatly speeding up the process.

2. By aligning nodes to fit to a grid, we benefit from an improved initial level of octilinearity (Section 4.4.1).

3. Providing the resolution of the grid is greater than the diameter of a node, we do not have to perform junction/junction occlusion checking, further speeding up optimisation.

4.3 Main Layout Process

Once pre-optimisation is complete, the main layout process is initiated. The layout process performs a number of iterations over the course of its operation. During each iteration a number of different methods are used to reposition nodes, these are run in the following sequence and are covered in the listed subsections:

1. Single Node Movement (Section 4.3.2).
2. Node Cluster Movement (Section 4.3.3).
3. Mid-processing (Section 4.3.4).

The layout process evaluates the quality of the schematic based upon a number of criteria. These criteria, along with metrics for calculating numerical values representing how well the current configuration adheres to each of them, are given in Section 4.4. Using these metrics to measure the fitness of each individual criteria, we can generate a single value to represent the total fitness of the schematic to all criteria. This value is known as the objective function, and is calculated by a summation of each weighted individual criteria value (see Section 4.4.11 for details on criteria weighting). Using an objective function to measure schematic quality allows us to perform incremental position adjustments to nodes, recalculating the fitness at each stage and retaining any movements which yield a lower (improved) value.

We perform this process during each iteration upon all nodes in the schematic (single node movement), along with a number of clustering techniques for moving multiple nodes at once (node cluster movement). Multiple iterations of movements are performed before arriving at the optimised schematic. The following section details the iteration process and introduces the closely related search distance parameter.

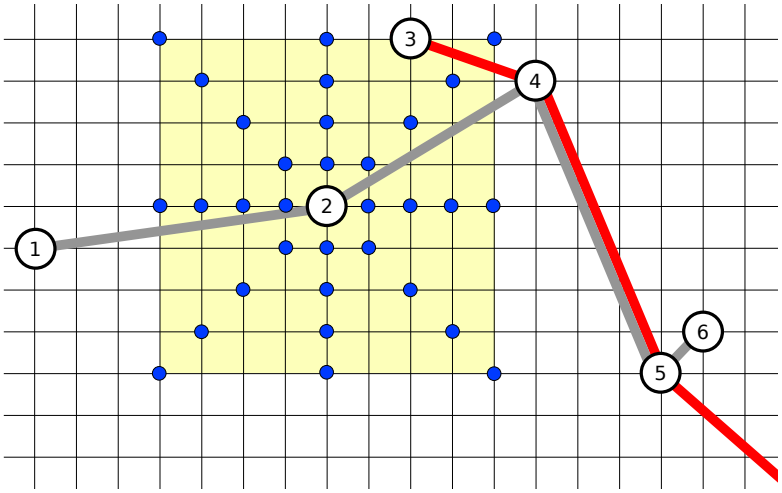


Figure 49: Possible node movement positions (blue dots) for junction 2 during an iteration with a distance of 4 units.

4.3.1 Iterations and Search Distance

The number of iterations performed and the search distance of the algorithm are configurable. By default we use values of 12 and 16 respectively, although as explored in Section 5 the optimum values vary based upon schematic characteristics. The number of iterations is a limit, as the algorithm will prematurely terminate if it cannot further lower the objective function.

Search distance refers to the maximum number of grid spaces a node can be moved away from its initial position during a single iteration. Figure 49 illustrates the possible movement positions (blue dots) for junction 2 during an iteration with a search distance of 4 units. Node repositioning is explained in further detail in the following section.

Search distance and the number of iterations are related, as the search distance is decreased linearly at each iteration until the final iteration uses a value of one unit. For the default values used, the search distance uses the following values for each iteration:

- 16, 14, 13, 11, 10, 9, 7, 6, 5, 3, 2, 1

This sequence of search distance values is calculated from the specified number of iterations and initial search distance. The distance reduction amount per

iteration (step) is calculated by $\frac{startDist}{iterations}$. The step is then subtracted from the current search distance and the result is floored to determine the distance during the next iteration. Flooring the value is the reason why, in this example, the distance reduction varies between 1 and 2 units per iteration – the step is 1.33. The actual distance value is recorded in double precision, and only floored for the iteration distance value.

Restricting the search space by applying a maximum search distance for each iteration is necessary to keep the execution time of the algorithm down, as criteria values must be recalculated for each potential position, comprising the major performance overhead.

4.3.2 Single Node Movement

During the single node movement phase, each node is examined sequentially in an attempt to improve the objective function of the schematic by finding more optimal positions. Nodes are always examined in the same order across all iterations to ensure the layout process is deterministic. The potential negative effect of using such a method is that we cannot be sure that the ordering method chosen is the most suitable. It is possible that a different ordering, such as starting with nodes that have the highest degree and working downwards, may produce improved results overall whilst still retaining determinism; however this particular area was not investigated in our work.

As mentioned previously, reducing the number of node positions to check yields large performance improvements. Therefore, as well as restricting the search distance, we restrict potential positions to lie on horizontal and diagonal lines from the node in question; this can be seen in Figure 49.

Nodes are placed in each position to be checked in turn (left→right, top→bottom), and the objective function recalculated. Fitness values are recorded for each position, and after checking all potential positions the node is moved to the best identified location. If a better position is not found, the node is left at its original location.

If a node is moved during either this phase or the upcoming clustering phases, a flag is set to indicate a change has been made to the layout. If an

Figure 50: An example situation requiring clustering methods to correct; assuming junction A is fixed in place and the desired edge length is 4 units.

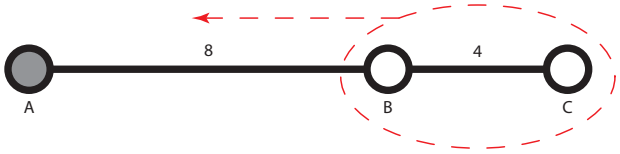
iteration completes without setting this flag, there have been no improvements found and the algorithm can terminate early.

4.3.3 Node Clustering

Although solely relying on single node movements will improve the schematic layout somewhat, there are many situations in which this technique is unable to improve the schematic. Figure 50 illustrates such an example – if we assume that junction A cannot be moved (for example if it has many connections and any movement breaks many criteria), then it is not possible to shorten edge AB. This is because it would be necessary to move first B followed by C; however, since moving B closer to A also increases the distance between B and C, the repositioning does not lower the objective function and results in no permanent movement.

In order to work around these quite common issues it is necessary to allow the optimiser to move multiple nodes simultaneously. We use a number of techniques for grouping (clustering) nodes to be moved together, outlined in the following three subsections. These clustering methods allow the layout method to escape from the most common situations that single node movements cannot solve.

Once clusters are identified, the layout method performs the same process as in single node optimisation, moving all clustered nodes together by the same offset as in single movement. At each position, the objective function is recalculated and nodes are left in the most optimal configuration. Moving multiple nodes at once frequently produces node/node occlusions – this is strictly disallowed by the algorithm which only allows one node per grid intersection. We therefore perform a check for this before any criteria are recalculated and disallow the move if any node/node occlusion is introduced.



(a) Length-based clustering used to identify and group offending junctions.

(b) Result of length-based clustering operation.

Figure 51: Length-based clustering example. Shaded junctions cannot be moved.

4.3.3.1 Length-based Clustering

Our first clustering technique groups nodes by the length of edges between them. This is used to correct issues such as previously seen in Figure 50.

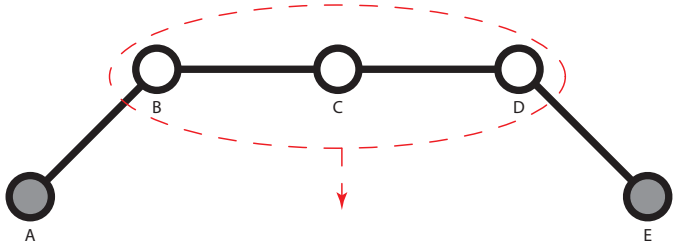
We first compile a list of all over or under length edges in the schematic. For each of these edges we then create clusters by performing a breadth first search starting from each connected node. Our breadth first search will only traverse edges of the correct length, adding each node it passes through into the cluster. If a starting node is already included in a cluster, we ignore it.

Performing the length-based clustering method upon Figure 51a results in nodes *B* and *C* being grouped; this then allows the optimiser to move both nodes towards *A* and fully satisfy the edge length criterion as seen in Figure 51b.

4.3.3.2 Angle-based Clustering

This clustering technique groups nodes by the angle at which incident edges pass through them and is used to fix issues such as that seen in Figure 52 where sections of a line are offset to their neighbours. This is a common trait caused by forcing curved lines (such as metro lines fitting to geography) into octilinear lines.

We examine each edge in turn, performing a breadth first search from each connected node. The breadth first search will only traverse edges that lie at the



(a) Angle-based clustering used to identify and group offending junctions.

(b) Result of angle-based clustering operation.

Figure 52: Angle-based clustering example. Shaded junctions cannot be moved.

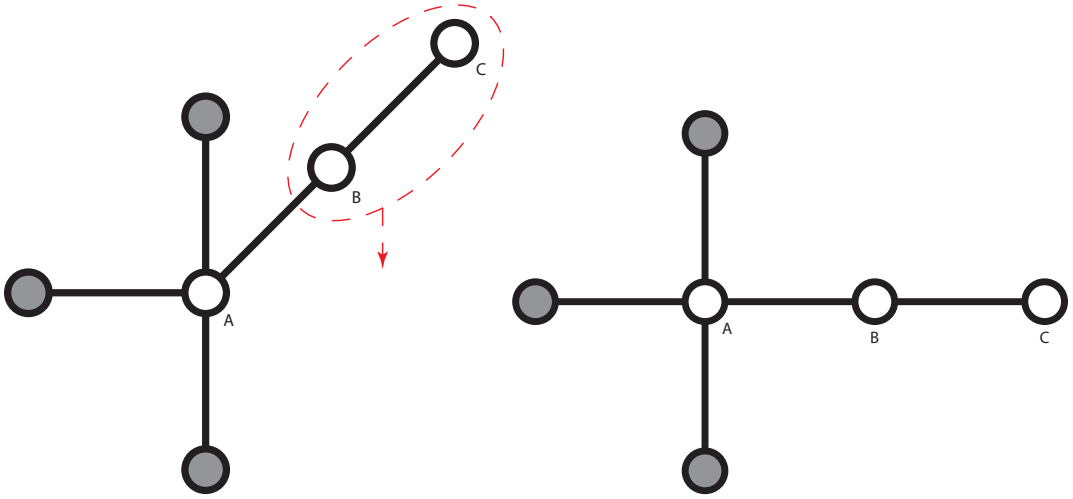
same angle to the initial edge, and are not already part of a cluster, adding each connected node to the current cluster.

Performing the length-based clustering method upon Figure 52a results in nodes *B*, *C*, and *D* being grouped; this then allows the optimiser to move them in line with *A* and *E*, fully satisfying the line straightness criterion as seen in Figure 52b.

4.3.3.3 Periphery Clustering

Periphery clustering targets an issue specific to schematics with a centralised area and protruding periphery sections, such as many metro maps. Figure 53a shows an example of the situation and Figure 53b an ideal solution. It is desirable to keep lines as straight as possible so the optimal solution is to straighten out nodes *B* and *C*. However, node *A* in Figure 53a is held in place by surrounding nodes and this makes the periphery section unable to move.

The periphery clustering method will perform a breadth first search starting from all termination nodes. It will traverse and add 2-degree nodes that lie along the same angle as the initial edge to the current cluster. For the example given in Figure 53, this will cluster nodes *B* and *C*, allowing them to move downwards and straighten over two iterations. This clustering method is capable of straightening periphery line sections even if they contain multiple bends.



(a) Periphery-based clustering used to identify and group offending junctions.

(b) Result of periphery-based clustering operation.

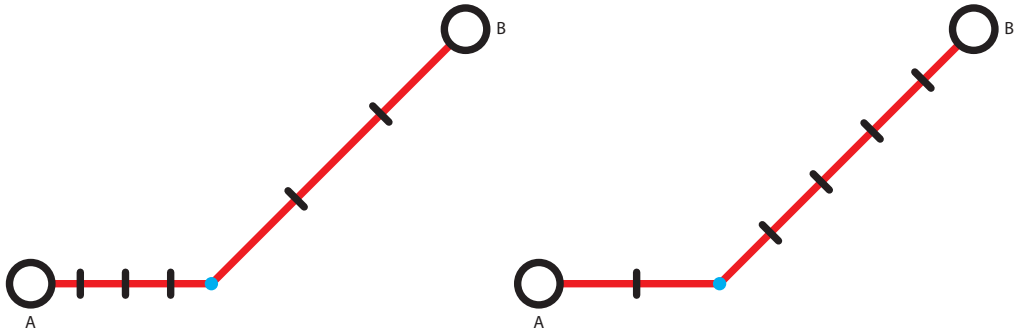
Figure 53: Periphery-based clustering example. Shaded junctions cannot be moved.

4.3.4 Mid-processing

The layout section of the iteration is now complete, but before proceeding to the next iteration we first perform a couple of mid-processing tasks.

The pre-processing section of the algorithm inserts a number of bend points into the schematic to allow greater flexibility. However, many of these are not needed and it is best to remove them for schematic simplicity as well as optimisation performance. During the layout process, each automatically-inserted bend point keeps track of the difference in the angle of the edges passing through it (automatically-inserted bend points always have a degree of 2). If this angle remains at 0° (straight) for two consecutive iterations, the bend point is removed from the schematic. It is during this mid-processing section that these checks are evaluated.

Along with removing bend points, we perform a rebalancing of stations. Although the desired length of edges is proportional to the number of stations on it, more strongly weighted criteria such as occlusions or crossings can occasionally cause bend points to position themselves in non-ideal places; causing bunched up or sparse stations (Figure 54a). As bend points are used simply to



(a) Unbalanced stations from layout movements.

(b) Stations have been rebalanced around the bend point.

Figure 54: Station rebalancing on edges separated by bend points.

allow bends along lines, we can freely move stations from one side of a bend point to another to help lower the fitness function. Figure 54 shows the result of the station balancing step applied to a line section with uneven station distribution.

4.4 Layout Criteria & Objective Function

This sections explains each of the criteria used during the layout process to determine the quality of the current configuration. The equations we use to evaluate each criteria value are provided, and commonly use the following characters:

- G = Graph (containing nodes and edges).
- N = Node.
- E = Edge.
- L = Line.

Many criteria value calculations listed are squared, this provides the advantage of a non-linear conformity progression where criteria are penalised more heavily the less they conform.

The resulting fitness values from each critereon have no upper bound, but will always be ≥ 0; a value of 0 indicating that this criterion is perfectly met. Our objective function is then calculated using Equation 12; a summation of all critereon values after they have been weighted as explained in Section 4.4.11.

$$fitness = C_0W_0 + \ldots + C_nW_n \tag{12}$$

where C_i and W_i are the fitness and weighting values for criterion i respectively.

4.4.1 Octilinearity

All edges should be at an angle that is a multiple of 45°. The octilinear layout criterion is evaluated using Equation 13.

$$octilinearity = \sum_{E \in G} \left(angle\left(E\right) - round\left(angle\left(E\right), r\right)\right)^2 \tag{13}$$

r = the resolution of accepted angles in degrees ($r = 45$ for octilinearity).
$round(\theta, r)$ rounds the argument to the nearest value that is a multiple of r.

4.4.2 Edge Length

The length of edge sections between schematic objects (including stations) should be equal to an ideal length. The edge length criterion is evaluated using Equation 14.

$$edgeLength = \sum_{E \in G} \left(t \times gridRes - \frac{length\left(E\right)}{stations(E) + 1}\right)^2 \tag{14}$$

t = Target length of edge measured in grid squares.
$gridRes$ = The resolution of the grid, used to convert a grid square measurement into a pixel distance.
The $stations(E)$ function calculates the number of stations on edge E.

The desired length of edges is defined in grid squares and multiplied by the grid resolution in order to ensure that nodes can be positioned where they will fully satisfy the edge length criterion. However, problems arise from this as

diagonal edges fitted to grid intersections are unable to achieve the exact desired length – this is because the diagonal of a square is $\sqrt{2}$ longer than the edge, so the calculated desired length will very rarely lie on a grid intersection. Rather than adjust the length calculation by multiplying the desired length by $\sqrt{2}$ and allowing the edge to be t diagonal squares, we retain the same calculation which causes nodes to position themselves on the intersection closest to the desired length, accepting that the criterion cannot be fully met for diagonal edges. This decision was made due to sequences of diagonal edges appearing very noticeably longer than their non-diagonal counterparts.

4.4.3 Line Straightness

Lines should be as straight as possible and have the minimum number of bends. The line straightness criterion is evaluated using Equation 15.

$$lineStraightness = \sum_{L \in G} \sum_{E \in L} angleDiff\ (E, E_{+1})^2 \tag{15}$$

The *angleDiff*() function calculates the smallest angle between the adjacent edges E and E_{+1}.

Figure 55 illustrates how the angles along a line are calculated. Lines that split into two edges at a node will be measured both relative to the first edge encountered. As mentioned previously this formula uses a squared exponent, providing the desirable behaviour that fewer sharper bends are penalised more than multiple smaller bends.

4.4.4 Line Straightness through Junctions (LSJ)

Lines should be as straight as possible when passing through junctions. The line straightness through junctions criterion is evaluated using Equation 16.

$$LSJ = \sum_{L \in G} \sum_{E \in L} \begin{cases} angleDiff\ (E, E_{+1})^2 & \text{if Junction} \\ 0 & \text{else} \end{cases} \tag{16}$$

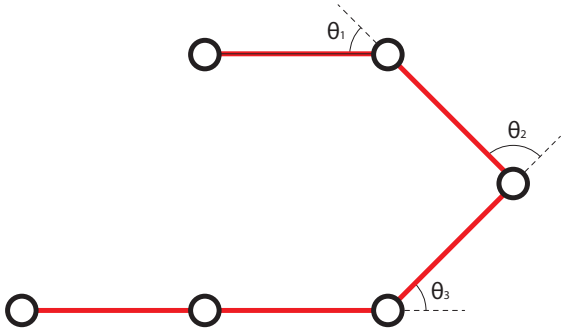


Figure 55: Calculating Line Straightness. Angles θ_1 to θ_n are squared and summed.

The *angleDiff*() function calculates the smallest angle between the adjacent edges E and E_{+1}.

This criterion is identical to the more general line straightness criterion but is only calculated for sections of the line that pass through junctions. This has the desired effect of shifting any line bends away from junctions and onto bend points, which is not penalised as heavily.

4.4.5 Line Straightness along Peripheries (LSP)

Periphery line sections should be as straight as possible and have the minimum number of bends. The line straightness along peripheries criterion is evaluated using Equation 17.

$$LSP = \sum_{L \in G_{periph}} \sum_{E \in L} angleDiff\ (E, E_{+1})^2 \tag{17}$$

The *angleDiff*() function calculates the smallest angle between the adjacent edges E and E_{+1}.

This criterion is identical to the more general line straightness criterion but is only calculated for periphery line sections (G_{periph}). Periphery line sections are identified during the pre-processing stage of optimisation (Section 4.2.3). This has the effect of applying additional emphasis on keeping periphery line sections straight.

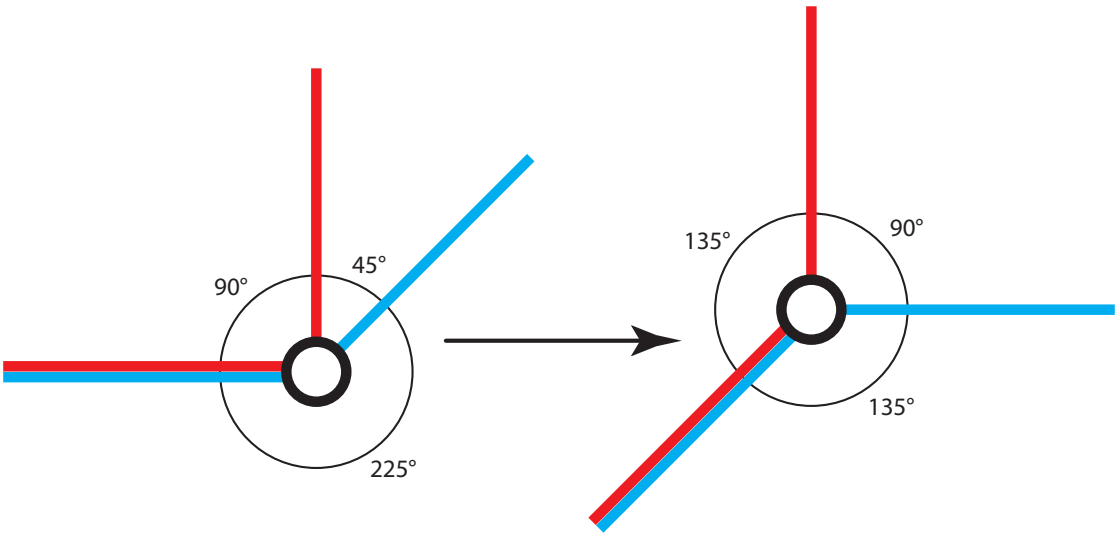


Figure 56: Improving the angular resolution whilst maintaining topological accuracy. It is often not possible to attain a perfect angular resolution whilst maintaining octilinearity as it requires a 4 or 8 degree node; although as shown here, even an imperfect result can greatly improve the aesthetic appearance.

4.4.6 Angular Resolution

The incident edges of a node with a degree of ≥ 3 should separated by an equal angle. The desired angle is first calculated and is equal to $\frac{360}{degree}$, this is then compared to the actual angles in Equation 18.

The angular resolution criterion is evaluated using Equation 18.

$$angularRes = \sum_{N \in G} \sum_{E \in N} \left(desiredAngle - angleDiff\left(E, E_{+1}\right)\right)^2 \tag{18}$$

$desiredAngle$ = The desired angle between incident edges.
The *angleDiff*() function calculates the smallest angle between the adjacent edges E and E_{+1}.

Figure 56 shows an example of how the angular resolution criterion affects a junction with 3 incident edges. We do not apply this criterion to 2-degree junctions as this situation covered by numerous line straightness criteria.

4.4.7 Parallels

SchemaSketch allows the user to identify edges as "desired parallels" along with a desired angle (Section 3.1.2). This criterion then penalises desired parallels for not being at the desired angle, causing them to align in parallel. The parallel criterion is evaluated using Equation 19.

$$parallel = \sum_{E \in G_{parallel}} (desiredAngle - angle\,(E))^2 \tag{19}$$

$desiredAngle$ = The angle to which desired parallel edges attempt to align to.

4.4.8 Occlusions & Crossings

Objects should be positioned so that they do not obscure any other part of the schematic. There are two situations in which this can occur; junction/edge occlusion and edge/edge crossings. Junction/junction occlusion is not possible providing the grid resolution is greater than the diameter of a junction.

Junction/edge occlusion is calculated by checking for occlusion between each junction and each edge in turn. Edge/edge crossing checking is done for each possible pair of edges. The criteria value for both crossings and edges is simply the number of occurrences multiplied by a weighting without a squared exponent. However, the weightings used for these criteria are relatively large resulting in any occlusions or crossings being heavily penalised.

4.4.9 Balance

The resulting schematic should be aesthetically balanced. Nodes should be evenly distributed, and dense/sparse areas should be avoided. Balance will be measured by placing a second grid, independent from the grid used to position nodes, over the schematic and measuring the density difference of the schematic between comparison squares. The balance grid is shown in Figure 57; pairs of cells that share the same character will be compared (represented mathematically in eqauation 20).

Q, S	U, W	Y, 1	Y, 2	U, X	Q, T
3, 5	E, F	I, K	I, L	E, G	4, 5
7, 9	M, O	A, C	B, A	N, O	8, 9
7, 10	M, P	C, D	D, B	N, P	8, 10
3, 6	F, H	J, K	J, L	G, H	4, 6
R, S	V, W	Z, 1	Z, 2	V, X	R, T

Figure 57: The overlay grid used to measure balance. Pairs of cells that share the same character will be compared; each cell has two comparison cells.

The grid is always 6×6, and is initialised to the size of the schematic bounding box as opposed to the entire canvas. This is because placing the grid over the entire canvas produces incorrect results if the drawn schematic is not centred. A value for each cell is then calculated by summation of the number of junctions, bend points, and stations within it, based upon their centre points. The grid is updated or recreated each time the schematic is modified, depending on the modification. If any part of the schematic is moved outside the balance grid, then the entire grid must be recreated; however, if a node simply moves between cells then the grid values need only be updated.

The grid is represented by a 2D integer array, the value of each cell given by the number of schematic objects within it. The pairs of cells shown in Figure 57 are then examined and the difference between their values is squared to create the error for that pair. The error for all pairs is summed to create the final fitness value for this criterion. This is evaluated using Equation 20.

$$balance = \sum_{x=0}^{\frac{n}{2}} \sum_{y=0}^{\frac{n}{2}} \left(cell_{x,y} - cell_{n-x,y}\right)^2 + \left(cell_{x,y} - cell_{x,n-y}\right)^2 \tag{20}$$

n = The number of cells on one edge minus 1 (for the 6×6 grid used, $n = 5$).

4.4.10 Topology

It is important that the topology of certain schematics, such as geographic transport networks, is not broken during layout. To ensure this is the case, the order of edges around junctions is strictly enforced and any movement resulting in change is disallowed. This is more a node movement rule than a criterion, as it uses no objective function and weighting, but is always strongly enforced. On abstract schematics where network topology is unimportant, this rule can be disabled.

4.4.11 Criteria Weighting

Fitness calculations produce values which vary greatly by criterion (up to many orders of magnitude) due to being measured on different scales. As using these unweighted values would put more emphasis on the criteria that are naturally larger, it is necessary to weight the values so that they are comparable before summation into the objective function.

Basic weighting involves multiplying the unweighted value by 1 over the maximum possible value; this constrains the value to between 0 and 1. However, it would be incorrect/not possible to scale all criteria in this way as they may either never reach the maximum in practice, or may not have a maximum. Therefore, to calculate weightings, we created a series of example graph layouts and recorded all unweighted criteria values. We averaged the value of each criterion across all tests and used the inverse of the average as a weighting. Occlusions and crossings are an exception to the standard weighting calculations, and simply use a value of 1, as these are to be very heavily penalised. Table 1 lists these weightings for each criterion.

Although these values appear to be very specific this is simply due to the method by which they were obtained. In actuality, these values are not that significant and, in order to visually see an effect, must typically be varied by orders of magnitude.

Criterion	Value
Octilinearity	4.322×10^{-3}
Edge Length	6.150×10^{-6}
Line Straightness	1.882×10^{-5}
Line Straightness through Junctions	1.882×10^{-5}
Line Straightness along Peripheries	1.900×10^{-4}
Angular Resolution	3.032×10^{-5}
Parallels	1.000
Occlusions	1.000
Crossings	4.762×10^{-1}
Balance	1.105×10^{-3}

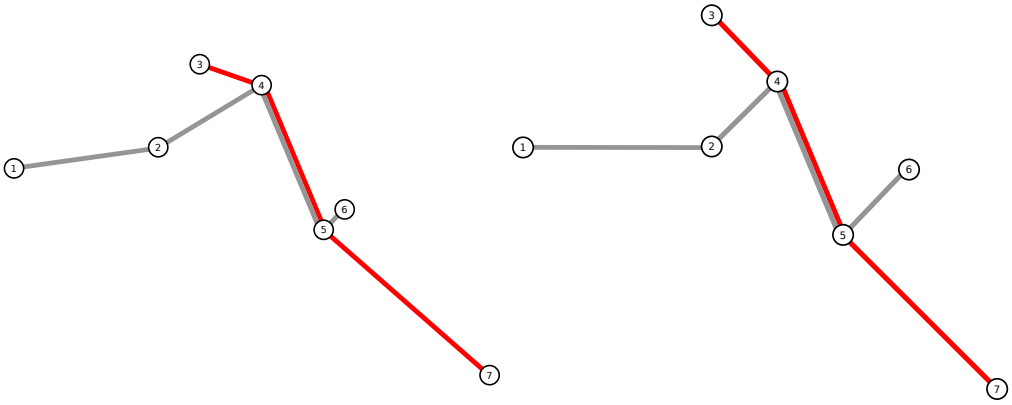
Table 1: Default criteria weightings used.

4.5 Minimum Working Example

Figure 58 provides a minimum working example to illustrate the effects of the layout method. An unoptimised schematic (Figure 58a) is used as the starting point. As explained, the algorithm moves each node in turn (numbers in the diagram indicate the order in which they are moved) attempting to lower the objective function, calculated by summing a set of weighted criteria metrics. Figure 49 shows an example of the node positions evaluated during an iteration for node 2 during an iteration with a distance of 4 units.

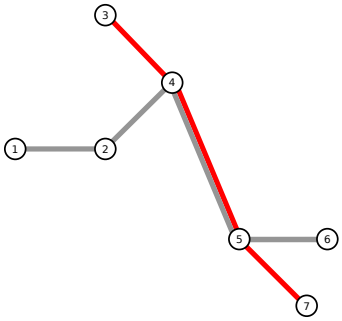
Figure 58d shows the resulting output, and it is clear to see how the algorithm has adjusted node positions to adhere to each criterion: 1) Each edge is now at a multiple of 45°, satisfying octilinearity. 2) Edge lengths have been equalised. 3) Lines have been straightened as much as possible (without introducing occlusion) both in total, as they pass through junctions, and at peripheries. 4) Angular resolution has been improved. 5) Line crossings have been avoided. 6) Occlusions have been avoided. 7) Balance has been improved. These changes result in the fitness value dropping from 7.63 to 0.74 as almost all criteria have now been fully applied. The resulting fitness value is due to unresolvable criteria conflicts – it is not possible to straighten both grey and red lines without occlusion, and two 45° line bends are required.

Figures 58b and 58c have been included to show exactly how the schematic

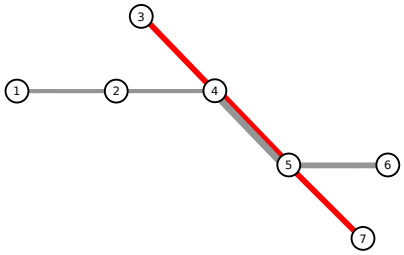


(a) Unoptimised Input (Fitness=7.63)

(b) First iteration (Fitness=4.22)



(c) Second iteration (Fitness=2.65)



(d) Optimised output after cluster movements (Fitness=0.74)

Figure 58: Minimum working example of the layout algorithm.

changes during each iteration of the algorithm. During the first iteration (58a to 58b), many nodes are moved into positions which satisfy octilinearity. This is due to octilinearity being weighted strongly as it is regarded as one of the most important criterion to schematic layout. It can be seen that there is one edge (from node 4 to 5) that does not manage to achieve this, and this is because moving either of the two ends to make the section octilinear would result in a negative effect on the other neighbouring nodes. Node 6 shows how the algorithm also attempts to standardise the edge lengths. This first iteration results in a drop in the fitness value from 7.63 to 4.22, as many octilinear issues are addressed. During the second iteration (58b to 58c), given that most nodes now already lie in octilinear directions from their neighbours, some over length edges are shortened (from node 1 to 2, and from node 5 to 7). Node 6 has also

been moved down, improving the total line straightness. These two changes further lower the fitness from 4.22 to 2.65. The transition between the last two schematics (58c to 58d) is a result of the clustering techniques used during the previous iteration. Nodes 1 and 2 are clustered by angle, and both moved upwards; nodes 5, 6 and 7 are clustered by length, and are also moved upwards to make the edge between nodes 4 and 5 octilinear. These final clustering techniques further reduce the fitness from 2.65 to 0.74.

4.6 Automated Label Layout

After the schematic layout method has found optimum positions for each node, the label placement algorithm is run. This process involves the repositioning of labels into one of a number of preset locations and orientations based upon two criteria. These criteria ordered by priority are:

1. Occlusions.

2. Position/rotation consistency.

Each of which explained further in the following subsection. Examples of the automated label layout can be seen in Section 4.7.

There are many positions a label can occupy. Firstly, the label can be placed in 8 locations: *North*, *North-East*, *East*, *South-East*, *South*, *South-West*, *West*, and *North-West*. Each location also supports 4 angles of rotation: *Horizontal*, *Positive Gradient*, *Negative Gradient* and *Vertical*. This combination of location and rotation offers a total of 32 possible label positions per node, as illustrated in Figure 59.

4.6.1 Placement Fitness

The following two criteria are used to position labels. Rather than weighting each and attempting to lower a summed objective function, we prioritise occlusions over consistency; this means that occlusions are minimised at any cost before consistency is examined.

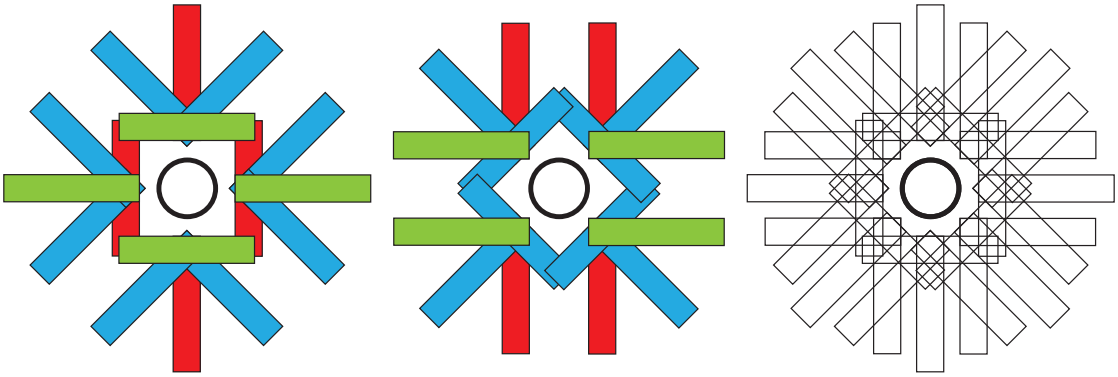


Figure 59: 32 potential label positions for a single label object. Drawn first in two images to aid comprehension (colour), then a combined version to illustrate label flexibility (no colour).

4.6.1.1 Occlusion – Priority 1

Each label must be checked against each other label, edge, and node. For label/label occlusion, each pair of labels need only be checked once and a variable incremented for each occlusion. For label/edge and label/node occlusion, each label must be checked against each other edge/node.

4.6.1.2 Consistency – Priority 2

Desirable label consistency is to have labels in the same position and rotation as their neighbours. Each label is compared to its neighbours and a value is incremented once if their position is not the same, and incremented again if their rotation is different.

4.6.2 Label Placement

The label placement itself is a two-stage process. Firstly, we group 2-degree node strings of labels together and position the entire group as one. During the second stage we examine labels individually. These stages of label placement are explained in the following two subsections.

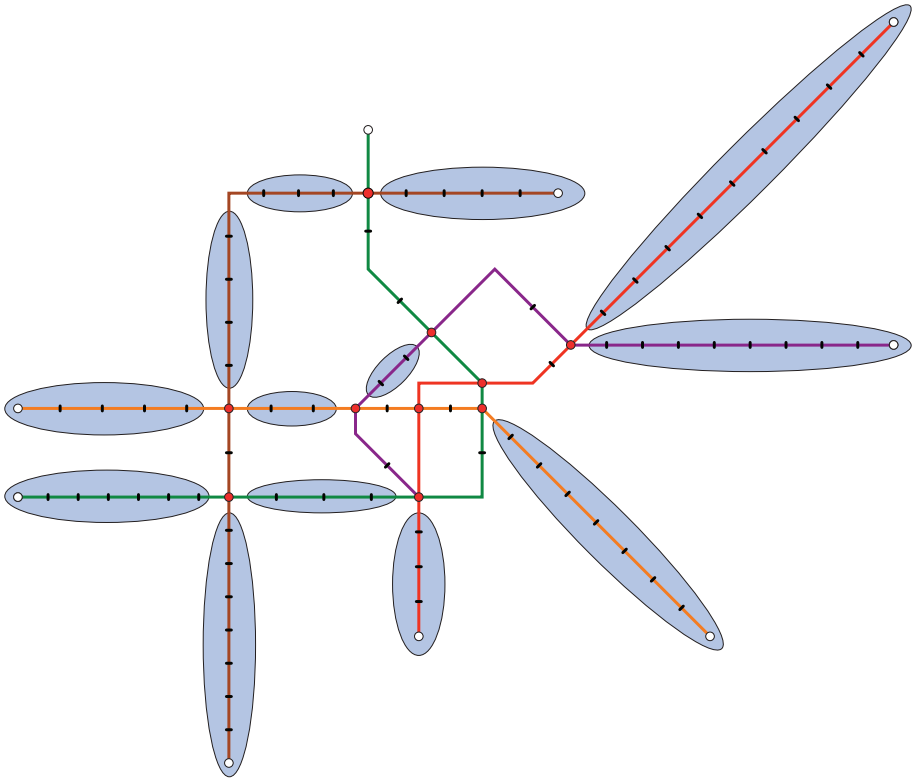


Figure 60: Example label groups are shown here. Take note of the top-left brown line, in which the label group is split by a line bend. Junctions filled with red are those with degree > 2 and initially discarded.

4.6.2.1 Label Group Placement

During the initial layout stage, we create groups of label objects based upon the degree of their parent node. In order to create these groupings, we first create a clone of the schematic and remove all nodes with a degree > 2. This leaves a number of strings of 2-degree nodes, which we use as groups after discarding any single nodes that have become disconnected. We further split up groups if the string of nodes changes angle by $\geq 30°$. All ungrouped nodes (those with degree > 2, or that have no 2-degree neighbours) are referenced in a separate array. We then give these nodes priority label placement in the following stage. A visual example of the groups generated from a schematic is shown in Figure 60.

Once the groupings have been created, each is examined in turn and all labels are moved simultaneously into each position and rotation to determine the best overall placement; because all labels in each group are moved together, position consistency is enforced. At this stage, it is very likely that some labels will occlude other parts of the schematic, we therefore focus on placing the group of labels in the position yielding the lowest number of occlusions. This process of moving groups of labels together greatly improves the position consistency of labels at the end of the label positioning process.

4.6.2.2 Individual Label Placement

Starting with labels that were not added to groups for the previous stage, each is moved into all possible positions and rotations to determine the best placement in terms of occlusions and consistency. Once all positions have been checked, we discard any options which contain occlusions and position the label in the best location for consistency. If no better location is found, we do not move the label.

For both stages of label layout, the order in which positions are evaluated is predetermined. This fixed ordering provides the effect of prioritisation for positions checked first. For example, given two positions A and B who produce the same fitness value and are evaluated in that order, position A will be chosen. The order, and therefore position priority, in which positions are evaluated is as follows: *North*, *South*, *East*, *West*, *North-East*, *South-East*, *South-West*, *North-West*. The horizontal orientation for each position is checked first, followed by the diagonals and finally the vertical orientation.

4.6.3 Summary

When compared to the previous label layout technique used by Stott et al., our method is relatively simple. The previous method operates as part of their standard multi-criteria hill climbing optimiser and uses five criteria as follows:

1. Occlusion.
2. Position priority.

3. Position consistency.

4. Station proximity (penalises labels that come into close proximity to unrelated stations).

5. Perpendicular tick (Encourages the tick for a particular station to be perpendicular to the line).

In contrast, our label placement technique only uses criteria 1 and 3. Criterion 2 is enforced by the order in which positions are evaluated, our method does not take into account criterion 4, and criterion 5 is implementation specific – as labels are placed relative to the end of station ticks, whereas we position relative to the tick centre.

Our labelling method operates independently and after the main layout process, and it could be argued that this is inferior to a method which operates during the layout process. The potential advantage of positioning labels within the main layout process is that the optimiser is able to move graph nodes to allow space for labelling; however, in practice we saw no negative effect from post-optimisation label layout using our method.

The major advantages of our technique come from the addition of a two-stage approach and the ability to rotate labels. The group-positioning stage provides very large benefits in label position consistency, and multiple options for label orientation go a long way in minimising occlusions whilst maintaining this.

In summary, our method uses fewer criteria with a two-stage approach combined with larger selection of label positions, and we believe that the results we have achieved are an improvement over the previous results in (Stott et al. 2010).

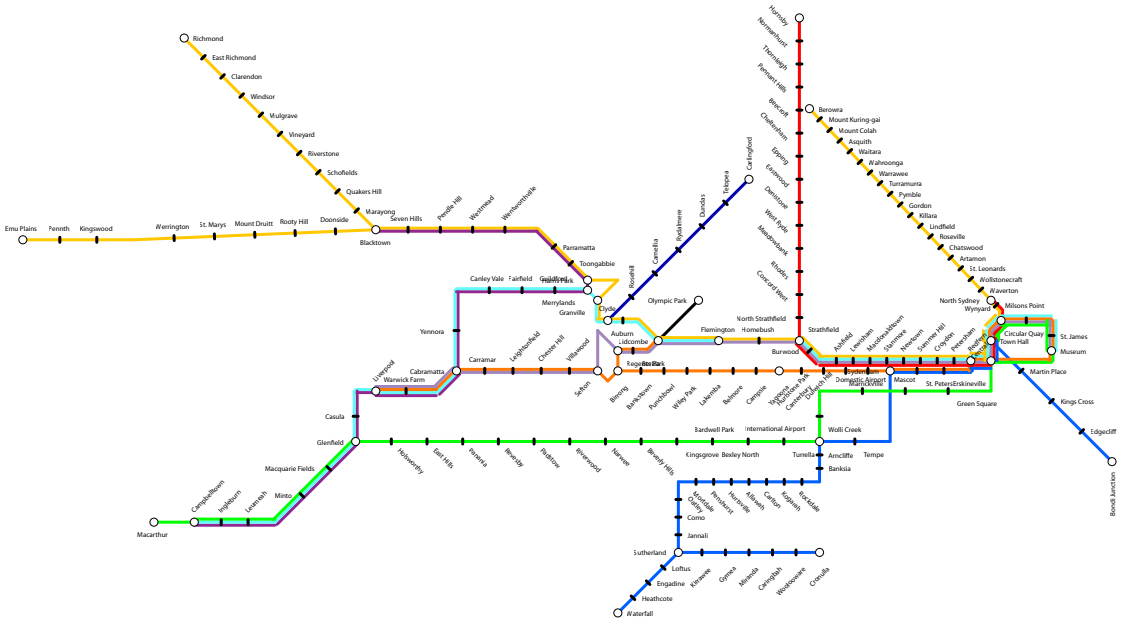


Figure 61: Sydney metro map as optimised by our multi-criteria hill climber.

4.7 Produced Layout Example & Comparison

Figure 61 shows the Sydney schematic as laid out using our method. This can be compared to Figure 62, produced by the previous multi-criteria hill climbing method by Stott et al. Our new result illustrates how our additional modifications have increased the line straightness and octilinearity in many areas, for example the central green line has now been kept horizontal and periphery sections contain fewer bends thanks to the new clustering technique. The city centre area to the right side of the schematic, typically very dense, has also been expanded slightly due to the new balance criterion. In addition to these points, the effect of bend points can be seen as lines now change direction more often along an edge rather than at junctions.

Along with significant layout improvements, an improvement can be seen in the positioning of labels. Our implementation that allows rotation of labels has allowed for fewer occlusions and an improved level of label position consistency overall.

An interesting aspect of our layout is that it appears to be more compacted vertically. This is possibly attributable to the fixed screen size with which we

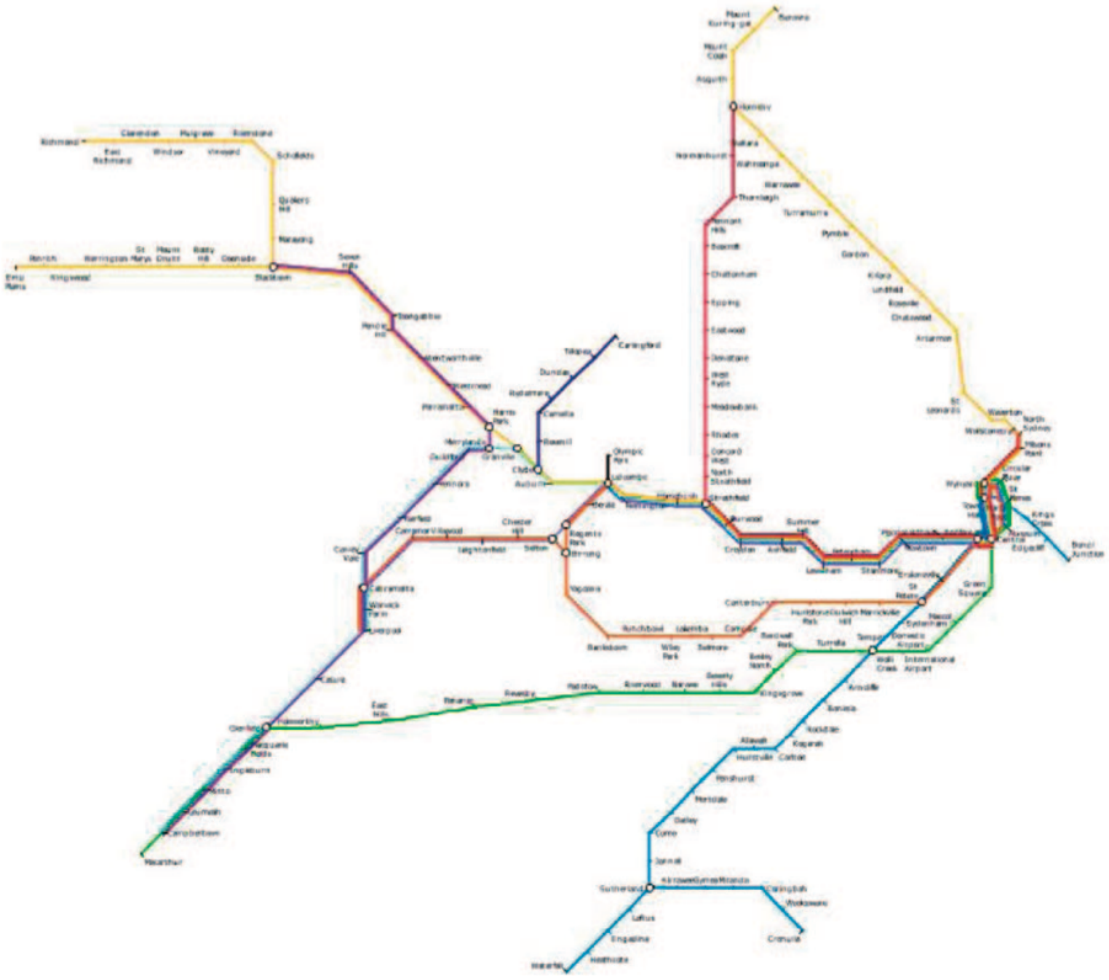


Figure 62: Sydney metro map as optimised by the previous multi-criteria hill climber by Stott at al.

worked not letting the schematic expand to its desired size, and has caused the horizontal section with many parallel lines (to the right) to be positioned in very close proximity to nearby separate lines. Along with this drawback, there are also a couple of strangely sharp bends (centre – yellow, purple), caused by bend points attempting to meet the octilinear criterion in a confined area.

4.8 Summary

This chapter has detailed our implementation of a multi-criteria hill climbing technique for the automated layout of metro map style schematics.

We have explained the modifications made to the original optimiser in (Stott et al. 2010), including 1) New schematic objects – bend points, in order to move line bends away from junctions/stations. 2) A change from operating on stations to bend points to reduce the number of objects for which to evaluate positions. 3) Reduced the number of potential positions for objects to move to without hindering layout in order to improve optimisation speed. 4) A new clustering method in order to ensure periphery line sections are straightened. 5) A number of new criteria: line straightness through junctions, line straightness at peripheries, enforced parallel lines and schematic balance. 6) Recalculated criteria weightings. 7) An improved labelling technique allowing much greater flexibility by rotation of labels. We believe the modifications we have made allow the method to produce layouts of a higher quality as explained in the previous section.

Although we employ techniques such as node clustering in order to allow our method to escape common occurrences of suboptimal layout; as pointed out in our examples, our layout technique still suffers from a number of occurrences of this. Sections of suboptimal layout are a common feature of search based optimisations, caused by the necessity to restrict the search space due to performance limitations.

We noticed that modifying the optimiser parameters had a large effect on the resulting output, and often resulted in different sections of the schematic not achieving an optimal configuration. We were therefore interested in exploring if a methodological approach to modifying parameters would produce any pattern of suboptimal layout sections in the output for specific schematics. If this was the case, we hoped that we could then identify optimal parameter settings based upon characteristics of the input schematic. The following chapter covers our work exploring how optimiser parameter manipulation affects situations of suboptimal layout in the final output.

Chapter 5

Exploring the Effects of Parameter Manipulation

In search-based graph drawing methods there exist a number of parameters which control the operation of the search algorithm. These parameters do not affect the fitness function, but nevertheless have an impact on the final layout. This chapter covers our work exploring how varying three such parameters (grid spacing, search distance and cooling schedule) affects the fitness value of the resultant diagram in our multi-criteria hill climbing optimiser.

By doing this, we hoped to identify patterns in the resulting layouts which were attributable to specific changes in parameter values, and to correlate these with characteristics of the input schematic. If such a relationship exists it would then be possible to automatically set optimiser parameters based on the input schematic, helping to produce a more optimal result than would otherwise be achieved using default values, without broadening the search space.

5.1 Testing Procedure

In order to perform the required testing, we implemented a testing rig into SchemaSketch which is capable of batch-optimising a large number of schematics whilst varying the parameter settings for each. Table 2 lists the schematics

Table 2: Schematics used

Schematic	Junctions	Stations	Edges
Washington	9	77	53
Vienna	10	80	63
Mexico City	24	123	120
Sydney	24	151	103

Table 3: Parameters and values used

Parameters	Values
Grid Spacing	5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
Search Distance	5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
Cooling Schedule	None, Linear, Exponential

on which the testing was performed, along with the number of junctions, stations and edges in each. These schematics were chosen as being representative of reasonably sized schematics demonstrating different characteristics and for which we could easily access the data. The table is listed by ascending order of the total number of junctions and stations.

We examined the following three key parameters of the method. Table 3 shows the values we chose to test for each.

1. **Grid Spacing**: A grid is placed over the canvas, and each node must be positioned onto a grid intersection. This parameter defines the grid resolution in pixels. When altered, this parameter affects the starting layout as the nodes are moved slightly when snapped to the grid. It will also alter the number of potential sites that nodes can be positioned in when they move, and the absolute distance by which they can move.

2. **Search Distance**: This parameter defines the initial (and maximum) distance that nodes can be moved, in terms of grid positions, each iteration. Increasing this parameter allows greater movement flexibility for nodes during layout. The distance by which nodes can be moved decreases over time as defined by the cooling schedule.

3. **Cooling Schedule**: The cooling schedule affects the speed and pattern of

the decrease in the node movement distance and can be set to one of the following three styles (listed in ascending order of speed of reduction):

None: The distance is not reduced and stays at the initial start distance value.

Linear: The distance is reduced by one grid space each iteration.

Exponential: The distance is reduced exponentially each iteration and is calculated using Equation 21.

$$Distance = \frac{StartDistance}{Iteration} \tag{21}$$

The cooling schedule parameter that we are testing is a modification to our layout method which was necessary to obtain more valuable test data. Previously, we used a predefined number of iterations for the layout to complete in, and the distance by which nodes could move was linearly decreased over this number of durations down to one, after which the optimisation was stopped. However, when testing the effect of parameters upon the fitness of the final layout, we did not want to use a fixed number of iterations. Instead, we wanted to let the optimisation run for as many iterations as required to achieve the best layout (down to a fitness function accuracy of 3dp), as we felt this was necessary to accurately see the effect of parameter modifications upon the final fitness. To replace the reduction in search distance based upon a fixed number of iterations, we implemented a cooling schedule to reduce the search distance over time. Our cooling schedule supports 3 different schemes, as explained previously.

Using the schematics and parameters listed in Tables 2 and 3 respectively, our testing rig optimises each schematic with every possible combination of parameter values (363 variations of each schematic). It then outputs images of each resulting layout and a file containing the fitness value and number of iterations required.

5.2 Optimiser Performance

The initial implementation of our layout method was not optimised for performance, making the type of experimentation here infeasible. In order to speed up the optimisation process, we implemented an alternate method for recalculating the objective function each time a node is moved. Rather than recalculate all criteria upon all objects in the schematic, which is highly inefficient, we generate a subgraph which references all affected objects. This then allows us to recalculate the criteria values for a smaller schematic, the modified subgraph, and sum the result with that of the previous configuration minus the previous fitness of the subgraph.

A requirement of this technique is that all criteria values must be cached, to allow retrieval for addition to the updated subgraph values at each change. We chose to store these criteria values within schematic objects themselves, for example nodes store their angular resolution (Section 4.4.6) and edges store their octilinearity (Section 4.4.1). There are criteria which are globally calculated, such as balance (Section 4.4.9), and these are stored separately.

In certain cases, in particular for edge crossings and occlusions, detecting which objects have been affected by a node movement is not trivial – a change in the position of a single edge can also affect the edges or nodes along its length. In order to avoid having to re-evaluate the moved edge with every other edge and node, we place an additional grid over the entire schematic. At the start of layout, each edge is examined and all edges passing through grid cells are identified. This edge location grid is updated each time an edge is moved. Using such a grid to monitor the location of edges greatly speeds up checking for edge crossings and occlusions, as the method can identify a subset of all nodes and edges as potential occlusion or crossing candidates by the grid squares in which changes have been made.

Table 4 shows the overall time performance increases gained by the algorithm improvements for each map. There is a large performance increase from the implemented changes, averaging at 7.5 times speed improvement across the four schematics, each optimised three times. This improvement in run time

Table 4: Overall time improvement across all tested parameter sets (minutes)

Diagram	Avg. Before	Avg. After	Speedup (times faster)
Washington	36.872	4.630	8.0
Vienna	51.929	10.581	4.9
Mexico	234.982	27.475	8.6
Sydney	518.813	61.340	8.5

made our testing process feasible by allowing us to carry out the required experiments in a reasonable time frame. The timings were performed on an ASUS Eee Pad Transformer TF101 running the Android operating system, version 3.2.1. The device uses a 1GHz NVIDIA Tegra 2 and has 1GB of RAM.

5.3 Results – Fitness

In this section we present the results from our tests relating to the final fitness value. Specific examples from the Vienna schematic are shown, as they provide a good level of variation between different runs. Data and layouts generated from the other examples can been found in the Appendix.

Table 5 shows an abridged table of the Vienna schematic results from the testing rig along with the parameter settings for each (full table can be found in the Appendix). It shows the ten best schematics by fitness value, the median schematic, and the worst. Some immediate trends can be seen when looking at the frequency of certain parameter values in the top ten schematics. For Vienna, search distance has a mode of 15, grid spacing 10 and no cooling schedule. Other maps show similar, albeit with different values, patterns of frequently occurring parameter settings in the top ten schematics, indicating possible map-specific trends.

From Table 5 we can also see that there is a continuing improvement in fitness at the top of the list. This pattern, where the best fitness is found for only one set of parameters, is also shown in two other maps (Mexico, Sydney). This may be an indication that the system is not converging on the optimum solution, and so an even wider examination of the search space may be required

Table 5: Vienna results (abridged)

Rank	Search Distance	Grid Spacing	Cooling	Iterations	Fitness
1	15	10	None	9	0.850
2	13	10	None	9	0.972
3	14	10	None	9	0.986
4	15	10	Linear	9	1.037
5	14	5	None	15	1.153
6	15	5	None	13	1.184
7	15	7	None	14	1.212
8	11	10	None	7	1.233
9	9	10	None	17	1.276
10	12	9	None	8	1.284
...					
181	10	9	Linear	7	2.270
...					
363	5	6	Exponential	3	6.694

to achieve the absolute best possible fitness. Washington shows the same fitness value in the two best-ranked schematics, and we believe this indicates that something near the best layout has been achieved.

Figure 63 shows the original geographic layout of Vienna, used as the starting point by the algorithm. Figures 64, 65 and 66 show the best, median and worst produced schematics respectively. Various cases of suboptimal layout are visible in the median Vienna diagram, for example it has generally worse line straightness than the best diagram in nearly all lines. Even more cases can be seen in the worst diagram, which along with much poorer line straightness, has multiple edges not meeting the octilinearity criterion, unequal edge lengths and occlusions. Similar examples can also be seen in the produced diagrams of the other maps.

Figure 67 shows the normalised cumulative mean fitness value against grid spacing for all maps. It clearly shows how grid spacing values at both extremities of the test set produce a worse final fitness value. Conversely, the best fitness values are produced near the middle of our test set, approximately using values varying between 8 and 11. When examining this data for individual maps a similar, but map-specific, trend is seen. We present these more detailed

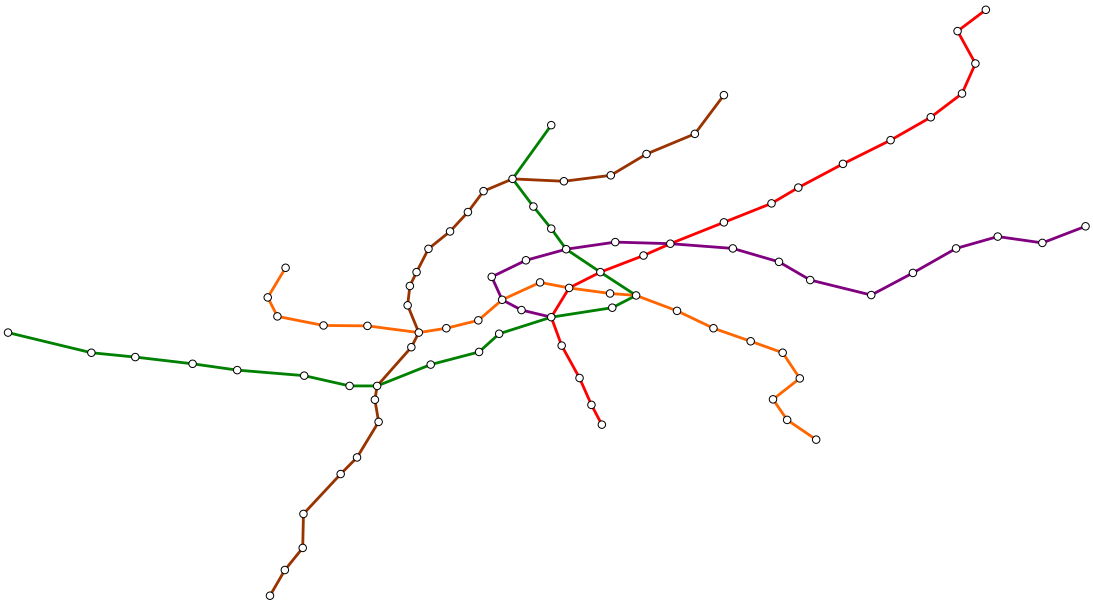


Figure 63: Vienna – Geographic.

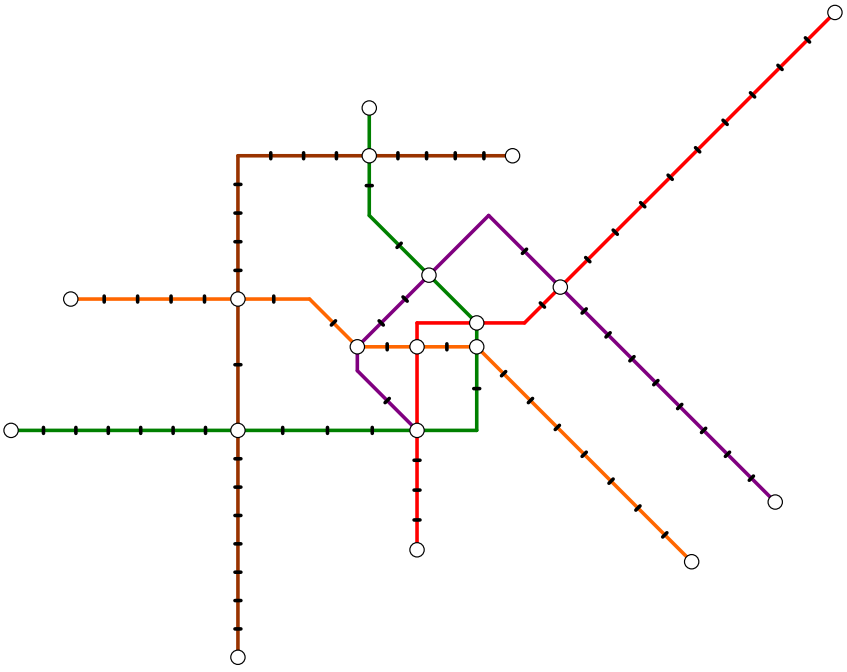


Figure 64: Vienna – Rank 1 (Fitness = 0.850).

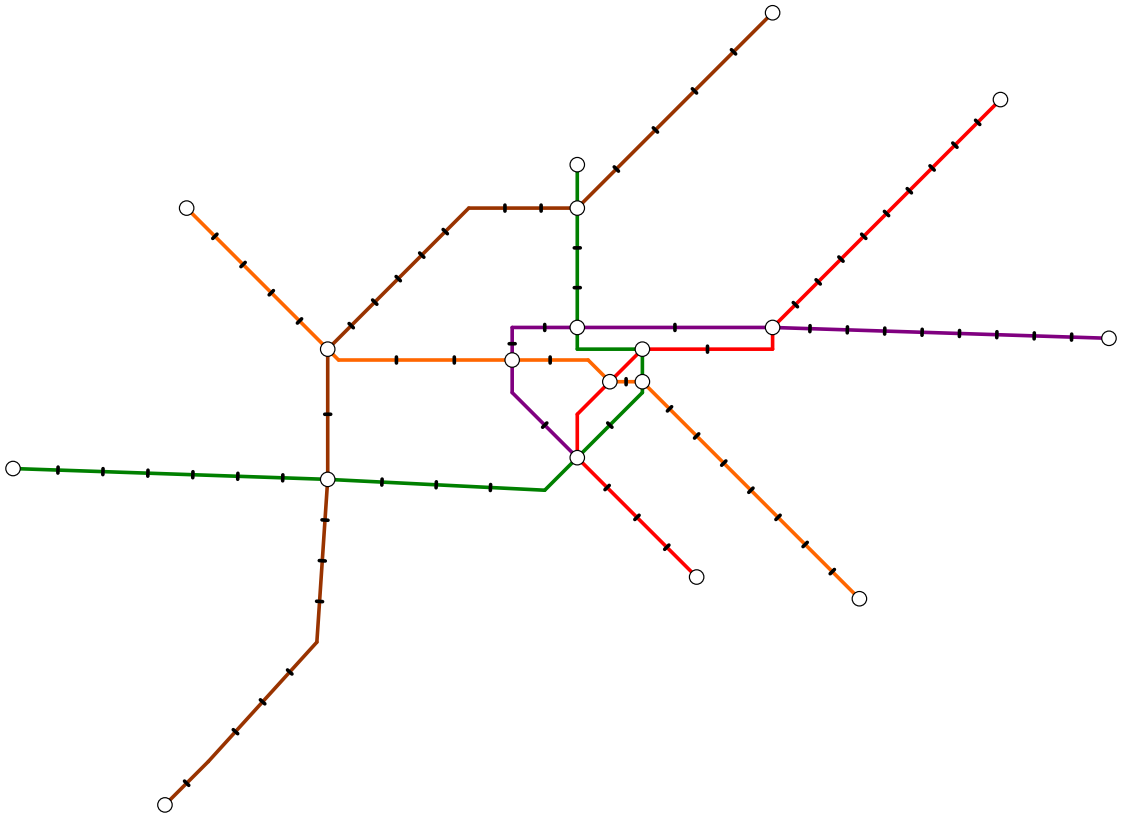


Figure 65: Vienna – Rank 181 (Fitness = 2.270).

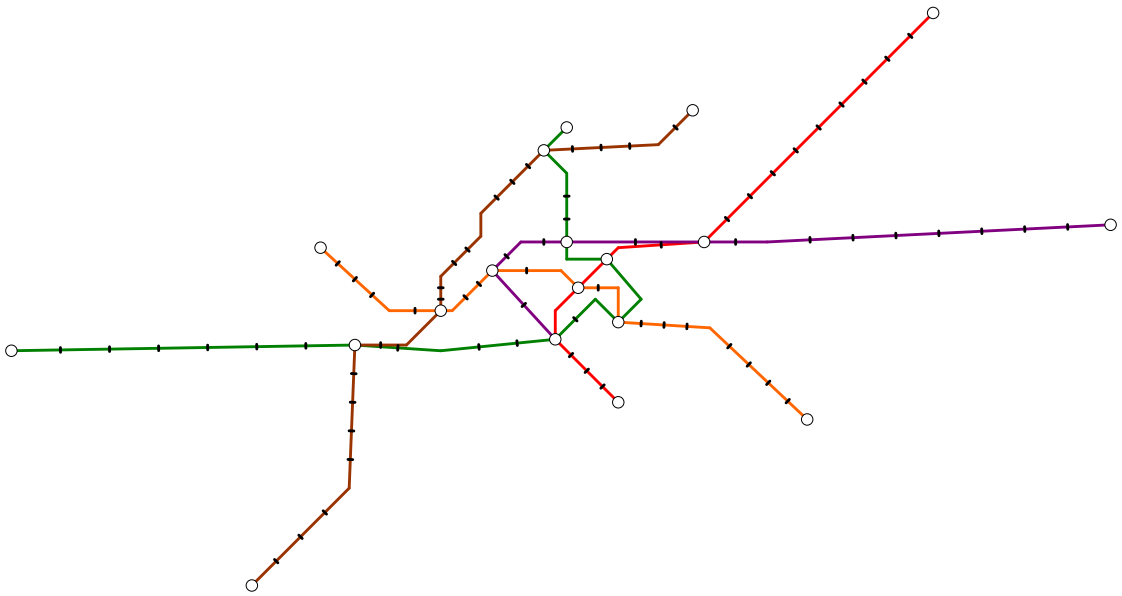


Figure 66: Vienna – Rank 363 (Fitness = 6.694).

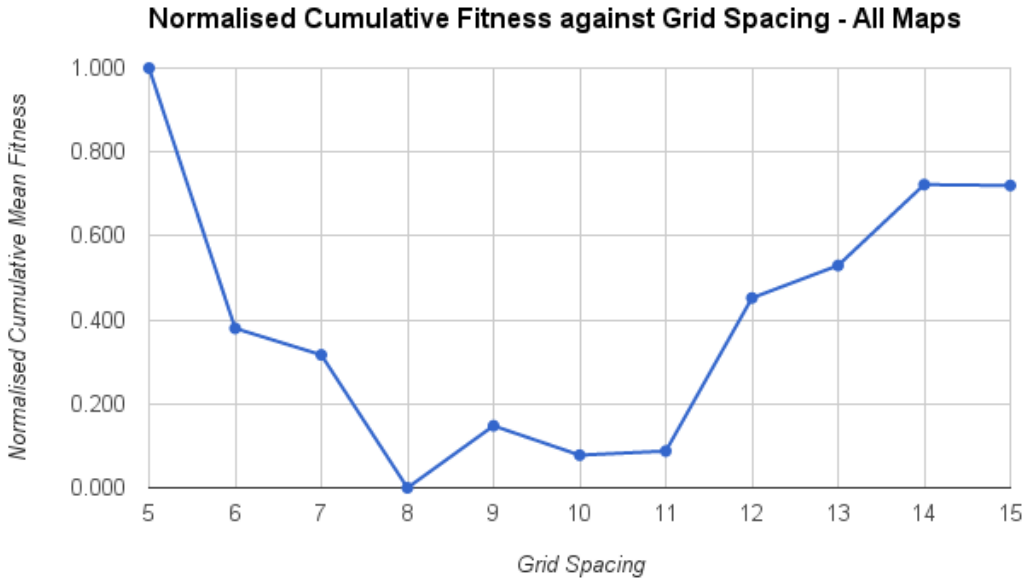


Figure 67: Normalised cumulative mean fitness value against grid spacing – all maps.

results in Section 5.3.1.

Figure 68 shows the normalised cumulative mean fitness value against search distance for all maps. This graph clearly shows how increasing the optimiser search distance reduces the resulting fitness value. This result is logical, as increasing the search distance of the optimiser directly increases the search space, allowing the optimiser to (mostly) find a better configuration. The same trend is seen for each individual map for which graphs are provided in the Appendix.

Figure 69 shows the normalised cumulative mean fitness value against cooling schedule for all maps. Using no cooling schedule produces the lowest fitness values, followed by a linear cooling schedule and then an exponential schedule. Similar to search distance, the cooling schedule directly affects the search space of the algorithm, thus hindering the fitness value as the search space is restricted. The same trend is seen for each individual map for which graphs are provided in the Appendix.

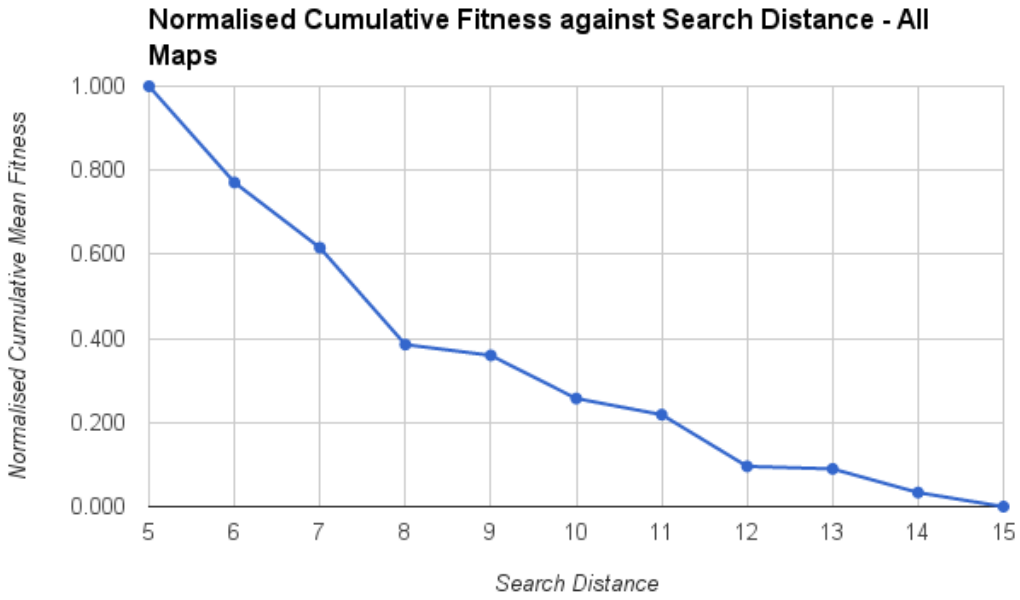


Figure 68: Normalised cumulative mean fitness value against search distance – all maps.

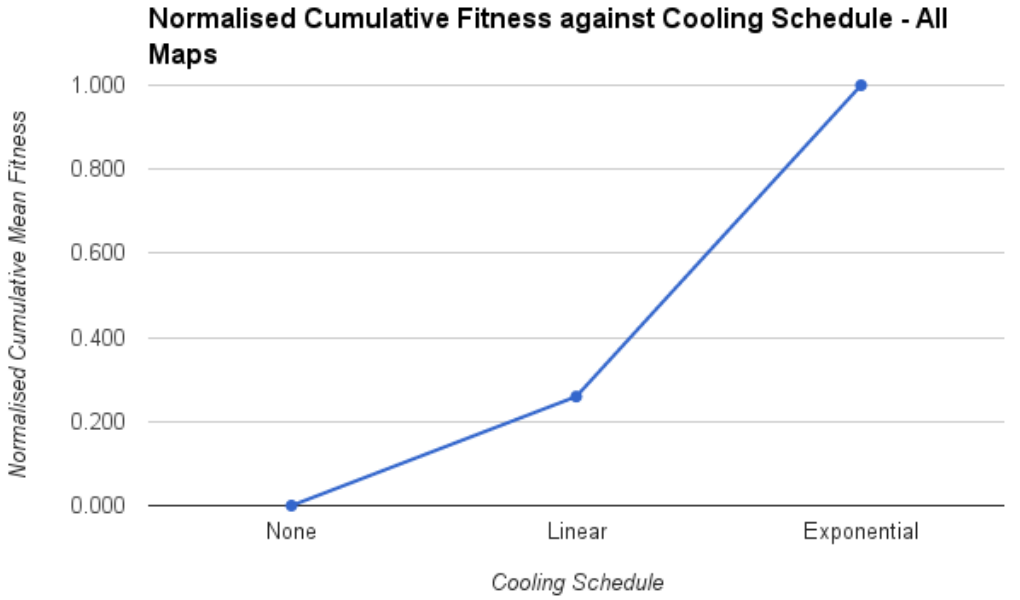


Figure 69: Normalised cumulative mean fitness value against cooling schedule – all maps.

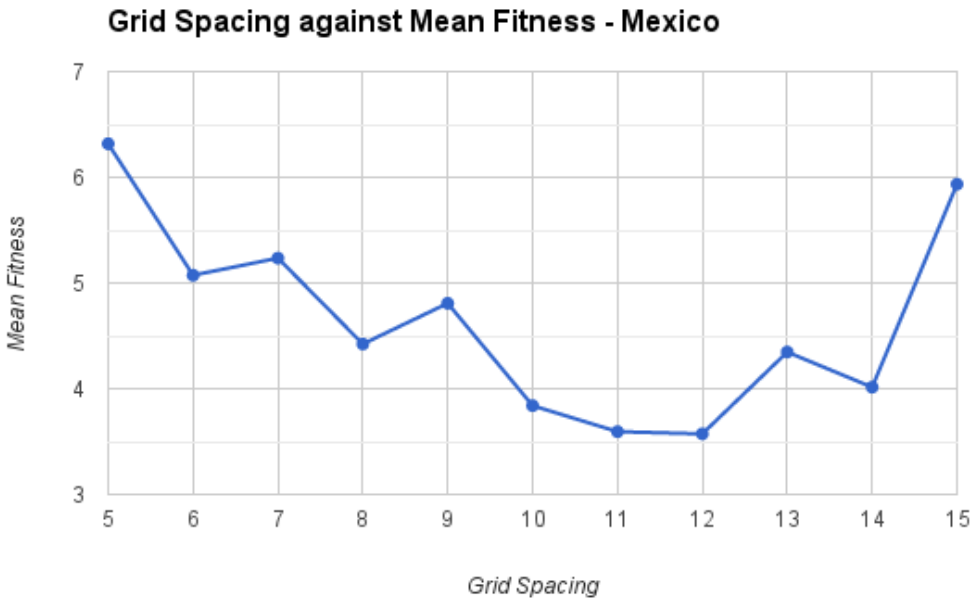


Figure 70: Grid spacing against mean fitness value – Mexico.

5.3.1 Grid Spacing

From examination of the fitness trends associated with grid spacing, search distance and cooling schedule; it is clear that the most interesting of the three is grid spacing. Unlike the other two which directly vary the search space and produce results that reflect this, this is not the case. Therefore, this section presents grid spacing fitness graphs for each individual map.

Figures 70, 71, 72 and 73 show graphs of how grid spacing affects layout fitness for Mexico, Sydney, Vienna and Washington respectively. These graphs provide more valuable information than Figure 67 which combined all maps, as they clearly show how each schematic has specific grid spacing values at which the best schematics are most commonly produced; these values are: Mexico: 12, Sydney 8, Vienna 10 and Washington 11.

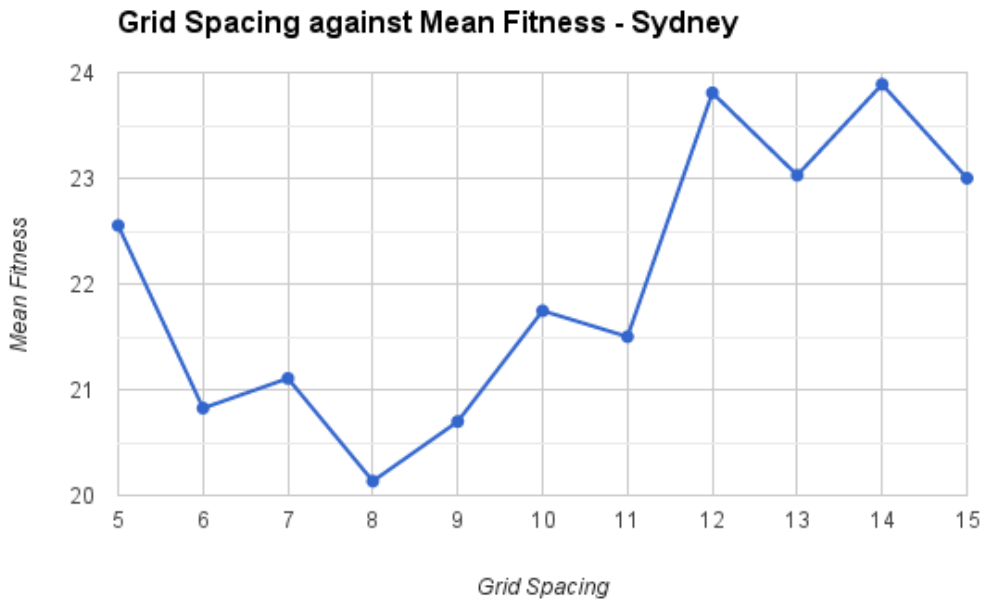


Figure 71: Grid spacing against mean fitness value – Sydney.

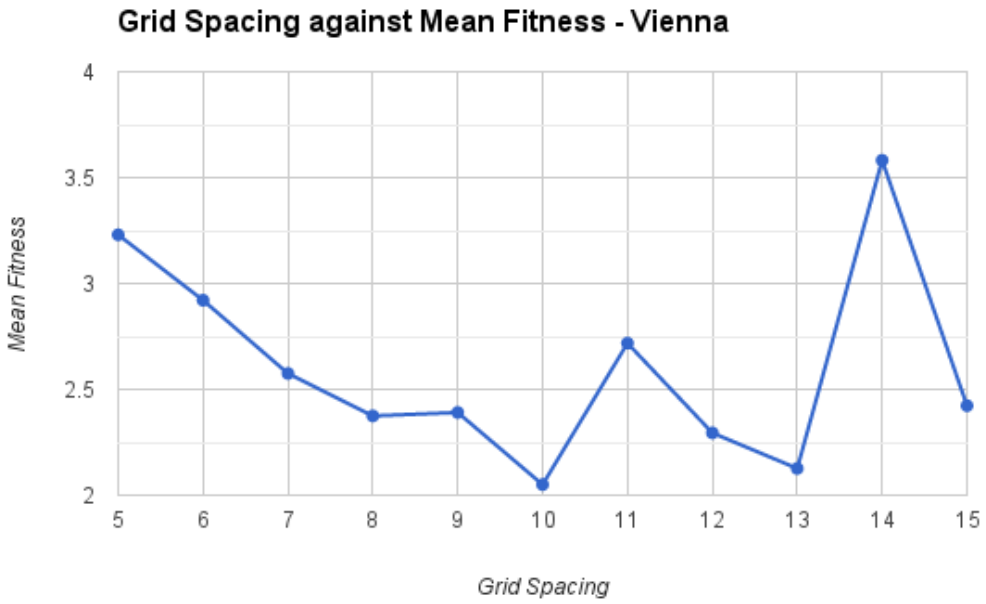


Figure 72: Grid spacing against mean fitness value – Vienna.

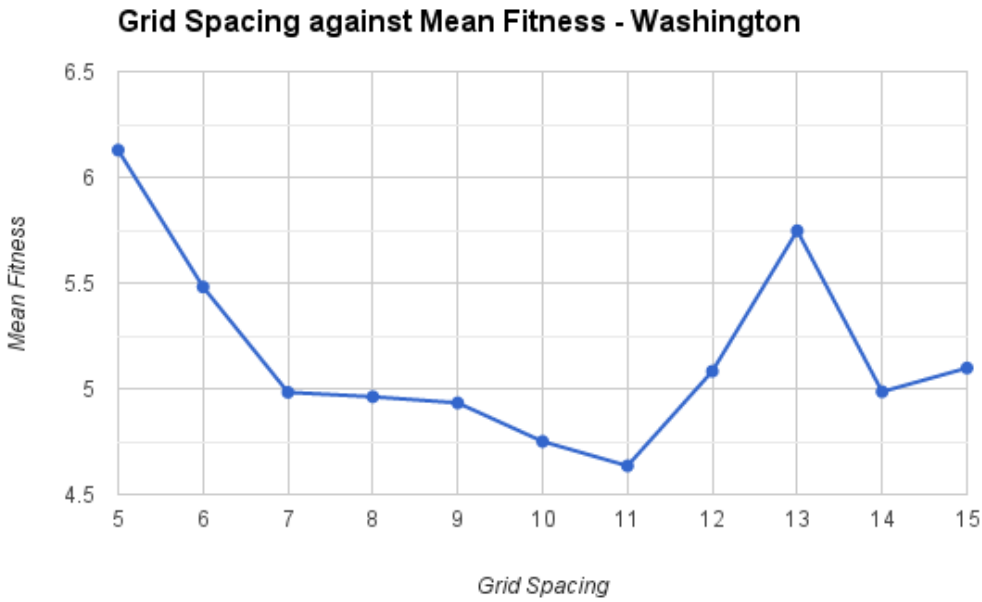


Figure 73: Grid spacing against mean fitness value – Washington.

5.4 Results – Iterations and Optimisation Time

Besides the final fitness value, it is interesting to examine how our parameters affect the number of iterations taken as well as the optimisation time. In this section we present the results from our tests relating to the number of iterations and optimisation time taken to achieve the final layout. The following graphs have been normalised due to the combination of multiple maps with different result scales, so the mean range must be taken into account when evaluating any apparent trends.

Figure 74 shows the normalised number of iterations and optimisation time against grid spacing for all maps. This graph shows a clear trend for both iterations and optimisation time with higher grid spacing values resulting in a lower average number of iterations and optimisation times. The number of iterations has a range of 5.38, which is fairly large considering the highest average number of iterations was 12.26 when using a grid spacing of 5. Although there is

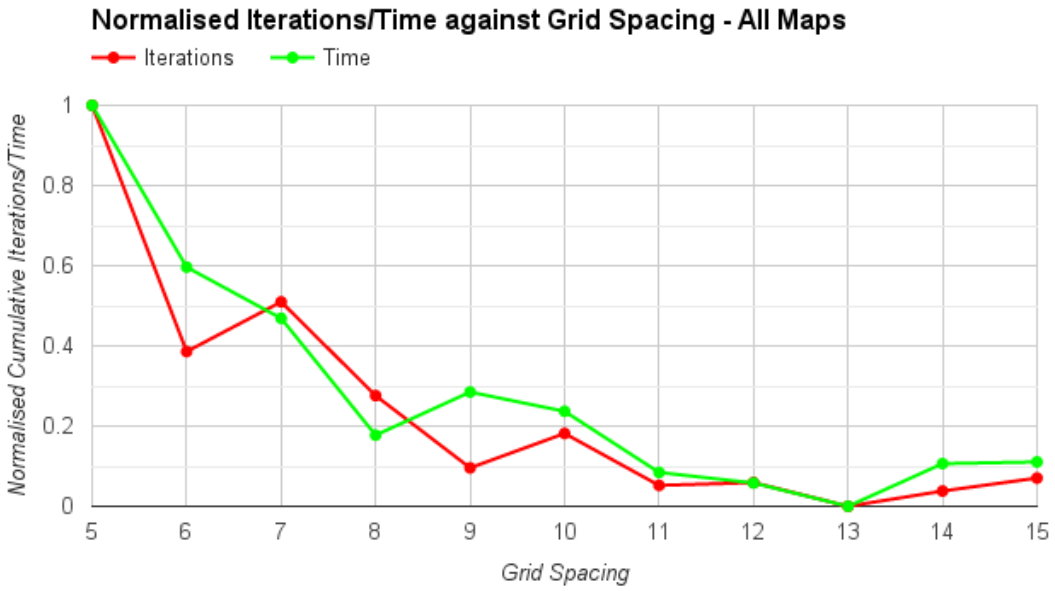


Figure 74: Normalised iterations and optimisation time against grid spacing – all maps. Range: Iterations (5.38), Time (12.58 minutes).

some fluctuation in data for specific grid spacing values when viewed for individual maps, the trend in general remains the same and unlike the final fitness value, there do not appear to be specific values which work well with particular maps. From this graph we can conclude that the smallest grid spacing values we used required significantly more iterations to achieve their final layout, and as grid spacing increases, the number of iterations required reduces and levels off. In terms of optimisation time the graph shows a similar effect, with a range of 12.58 minutes which is a 79% decrease in optimisation time from an average optimisation with a grid spacing of 5 to a grid spacing of 15.

Figure 75 shows the normalised number of iterations and optimisation time against search distance for all maps. It can be seen that there is a decrease in the number of iterations required as search distance is increased, but with a range of 1.26 iterations, the difference is not significant. This same trend is visible on individual map graphs, with each showing very little variance in the number of iterations. Although there is no significant change in the number of iterations, the optimisation time shows a very linear increase as search distance is

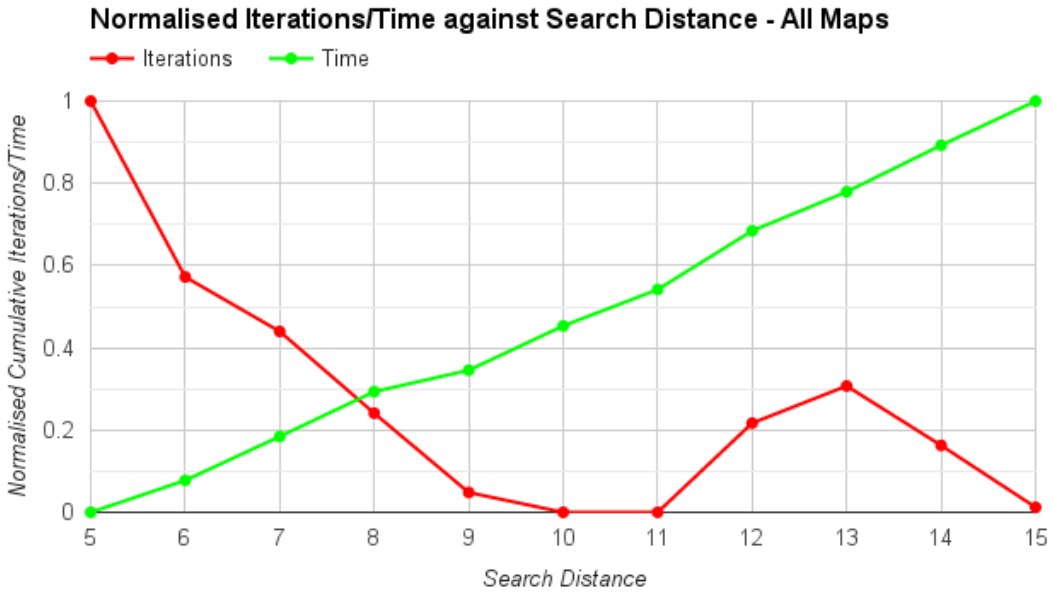


Figure 75: Normalised iterations and optimisation time against search distance – all maps. Range: Iterations (1.26), Time (20.23 minutes).

increased. With a range of 20.23 minutes, which is a significant 204% increase over the time of an average optimisation with a search distance of 5 to a search distance of 15, it clearly shows how increasing the search space via search distance directly affects the optimisation time.

Figure 76 shows the normalised number of iterations and optimisation time against cooling schedule for all maps. Although from this normalised graph it appears as though a linear cooling schedule leads to fewer iterations, the range of the data is 0.67 – less than a single iteration. This range is insignificant, and so we can deduce that the cooling schedule has no significant effect upon the number of iterations. There is a very clear decrease in optimisation time as the search space is restricted (None $\rightarrow$ Linear $\rightarrow$ Exponential). This decrease has a range of 17.32 minutes, which is a 156% decrease in optimisation time from an average optimisation with no cooling to an optimisation with exponential cooling. This is a significant decrease, and again shows how increasing the search space directly affects the optimisation time.

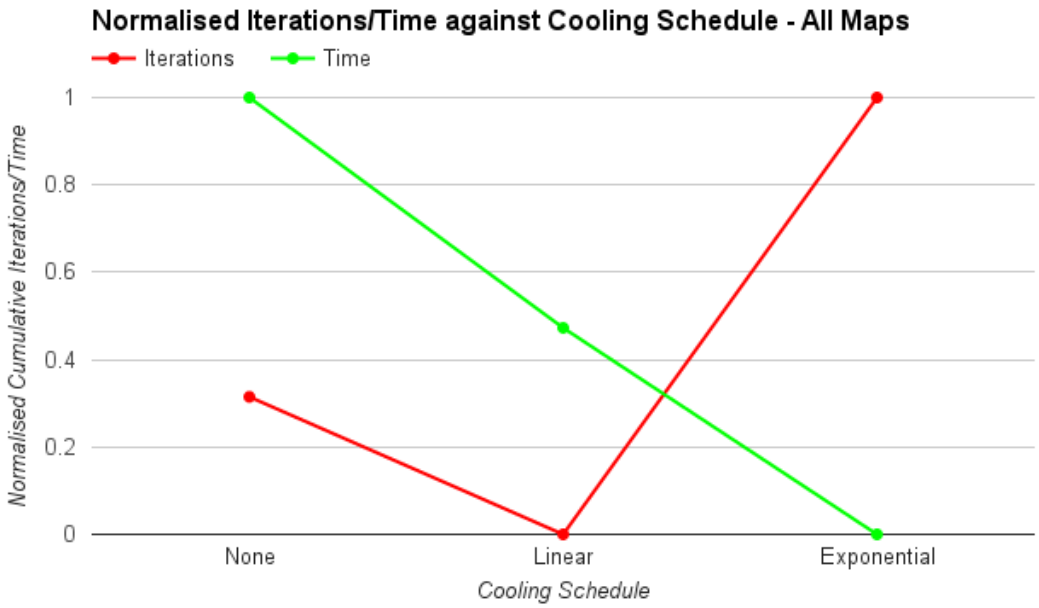


Figure 76: Normalised iterations and optimisation time against cooling schedule – all maps. Range: Iterations (0.67), Time (17.32 minutes).

5.5 Discussion

Our tests have indicated that for all four maps there is a very clear trend towards better layouts being produced by a greater search distance and slower (or no) cooling schedule. This behaviour is somewhat expected as increasing these parameters allows the system to evaluate more node positions at the cost of increased run time as seen in Section 5.4, however when beginning these tests we believed there was a possibility that specific distances or schedules may work better with different maps. Unlike start distance and iterations, grid spacing did not display any obvious trend for improving fitness across all maps; but rather showed that each schematic has an optimum grid spacing value at which lower fitness layouts are more commonly produced, as can be seen in Figures 70, 71, 72 and 73. A connection between optimal grid spacing and map density could be proposed, as the order in which maps increase in density inversely correlates to their best grid spacing value (ordered by perceived increasing density, best grid spacing in brackets): Mexico (12) $\rightarrow$ Washington (11) $\rightarrow$ Vienna (10) $\rightarrow$

Table 6: Percentage improvement of best schematic over median.

Map	Best	Median	% Improvement
Mexico	2.653	4.379	39.415%
Sydney	19.359	21.944	11.779%
Vienna	0.850	2.270	62.555%
Washington	3.844	4.976	22.749%
Mean			34.125%

Sydney (8). This is perhaps intuitive, as a denser schematic would benefit from finer control of node positioning. However, this relationship needs to be verified with a larger number of schematics and a metric for objectively quantifying density.

When parameters are set in an ad-hoc fashion and a wider search is not performed, as in the case in most layout methods, the fitness of the expected output is equal to the median fitness value – this allows us to approximate how much our best diagram improves upon an average untuned optimiser result. Table 6 shows the percentage improvement of the best schematic over the median for each map, and shows that the produced schematic will have on average a 34.125% lower fitness value than when using ad-hoc parameter settings. It has been mentioned that a small change in algorithm parameters can have a large effect on the resultant layout. An example of this is the best Vienna layout which has a fitness of 0.850 with parameters: search distance 15, grid spacing 10 and no cooling. A change from 10 to 11 grid spacing (one step) results in a drop to rank 94 with over double the final fitness (2.006). Both of these points illustrate the large effect optimal parameter settings can have upon the resulting layout, and highlight the importance of finding any connection between schematic characteristics and optimiser settings.

Interestingly, although trends have been identified in all parameters (grid spacing on an individual map basis), it can be seen in the results tables that the best layouts are not when all the parameters are at their best settings as indicated by the individual trends; but that there is still considerable variation.

5.6 Summary

When implementing any multi-criteria search method, many parameters are used to configure the algorithm, for example those studied in this chapter. We have performed a number of tests to examine how the variation of three such parameters affects the fitness and performance of the resulting layout, and made the following observations:

1. Increasing the search distance of the optimiser reduces the resulting fitness at the expense of optimisation time and there do not appear to be specific values that are suited to particular maps.

2. The optimal grid spacing value is specific to each map, and appears to be dependent on node density. Grid spacing also has an effect on optimisation time, with larger values requiring fewer iterations and less time to complete.

3. Slower cooling schedules (optimally no cooling) reduce the resulting fitness at the expense of optimisation time and there do not appear to be specific schedules that are suited to particular maps.

From these observations we conclude that for search distance and cooling schedule, each should be set to ensure the largest search space is used that can be evaluated in the target timescale. We also propose that grid spacing could be automatically tuned to adapt to the density of the map. However, from our testing we cannot draw any solid conclusions. It may be that due to the very large number of variables involved with these methods, in order to see conclusive trends in optimiser values a currently unfeasibly large data set must be evaluated.

Besides identifying trends, our results clearly highlight the importance of using optimum parameter values, which leads to an overall mean fitness improvement of 34.125% over using ad-hoc values. When algorithms for schematic layout are designed for the highest possible output quality with little regard to performance, any further improvement is desired; and our test provides evidence that optimising parameter values has a large positive effect on the fitness of the final result.

Although search-based approaches such as the multi-criteria hill climber tested in this chapter are capable of producing layouts of a high quality, there are a number of fixed limitations on these types of optimisers which limit their suitability for specific applications such as dynamic layout. These limitations are mostly performance-related and include: 1) Speed – search-based optimisers evaluate a very large data set, and therefore frequently take considerable time to complete. 2) Scalability – typical exponential time-complexities of these algorithms further heavily hinder their performance on larger data sets. 3) Computational process animation – the optimisation process moves nodes around to evaluate potential positions and is chaotic to watch, as a result most methods hide this process and simply produce a final output when finished. For these reasons, we decided to switch to exploration of force-directed optimisers which have received much less research in the specific area of metro-map layout, but which we believed would solve many of the issues of search-based optimisers.

Chapter 6

Force-directed Octilinear Layout

This chapter describes the steps taken in the development of a method to enforce edge octilinearity in a force-directed layout. To implement such a method, we first developed a new piece of software capable of creating node-link diagrams, and implemented a basic force-directed layout method on which to develop our modified algorithm. The motivation for an octilinear force-directed method is explained in Section 6.1, an overview of the software is then given in Section 6.2, and Section 6.3 details the steps taken during the development of our octilinear force-directed layout method.

6.1 Motivation

As explained in Chapter 2, Hong et al. first defined a number of criteria for metro map layout and attempted to simulate them in their force-directed metro map layout in (Hong, Merrick and do Nascimento 2005) and (Hong, Merrick and do Nascimento 2006). However, the layouts produced are not of a high quality due to weakly enforced octilinearity and unsatisfactory station spacing. The force-directed method for metro map layout has since been superseded by other methods which allow easier definition of layout criteria such as hill-climbing (Stott et al. 2010), simulated annealing (Anand et al. 2007) and linear programming (Nöllenburg and Wolff 2006), all covered previously. Although these methods can already compute layouts of a reasonable quality, a

force-directed method that could produce comparable results would provide a number of benefits. Firstly, it is a mature layout method which has been studied in great detail from a number of perspectives, so it is possible to piggy-back on considerable related work in, for example, performance improvements and scalability (Frick, Ludwig and Mehldau 1995) (Bartel et al. 2011) (Hu 2005). Secondly, the flexibility of force-directed layout also presents other potential advantages, for example use with a dynamically changing data set where the algorithmic computation can be viewed in real time as a transition animation from one graph state to another (Archambault and Purchase 2012). Thirdly, the performance of force-directed layout methods is far superior to that of search-based methods such as multi-criteria hill climbing and linear programming.

6.2 Graphs – FDOL

We first developed a piece of software, called Graphs – FDOL (Force Directed Octilinear Layout), which allows the creation and modification of node-link diagrams (Figure 77). It is possible to load in existing schematics, and we used a collection of real world metro maps for testing purposes. The software is written in Java, has been tested on Windows and Macintosh, and is freely available for download at: `http://www.cs.kent.ac.uk/projects/fdol/`.

6.2.1 Graph Creation and Modification

Our software allows easy creation of undirected node-link diagrams, such as those shown in Figures 77 and 78. Users can add nodes by double-clicking in the graph view where they wish to place a new node, and create edges by performing a right-click drag between any pair of placed nodes. Both nodes and edges can be right-clicked to bring up a contextual menu allowing access to the object properties panel, along with a delete option. Nodes also have the option to be "Pinned", which will prevent all non-manual movement (e.g. during automated layout). Graph nodes and edges can be moved by a click-drag movement. Moving an edge will also move the two connected nodes. Likewise, moving a node will update all connected edges.

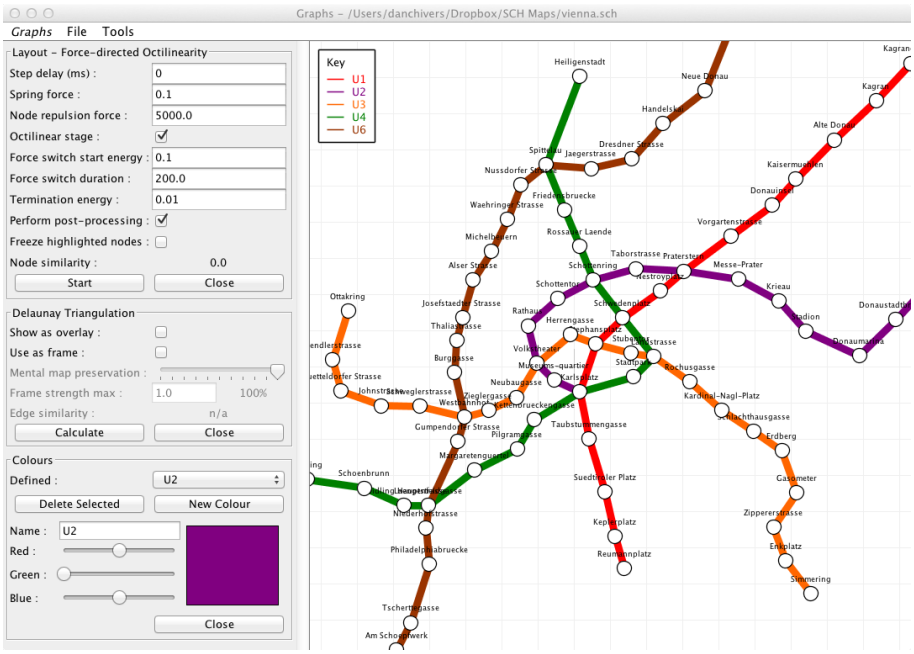


Figure 77: Graph visualization development software showing a geographic Vienna metro map with default labelling. To the left can be seen settings for (from top to bottom) force-directed octilinear layout (Section 6.3), Delaunay triangulation mental map preservation (Section 7), and line colour tools.

Each edge is associated with a line colour. New line colours are created in the "Colours" panel (bottom-left of Figure 77) by selecting "New Colour" and setting the name and colour. Edges will be created with the currently selected colour, and this allows parallel edges to be created between a pair of nodes by using multiple colours. Right-clicking on a blank section of the graph view will display a radial colour selection popup, allowing easy switching between all defined colours when creating graphs.

6.2.2 Layout

Our software supports both standard force-directed and octilinear force-directed schematic layout methods. Octilinear force-directed layout is our extension of the standard model and is explained in detail in Section 6.3. The following subsections will provide an overview of the implemented standard model.

The standard spring embedder has two types of force which act upon the

nodes to produce a layout – a localised repulsive force between all pairs of nodes and an attractive force between nodes directly connected by an edge. These are explained below and formulas given for calculation.

6.2.2.1 Node repulsion forces

Each node in the graph produces a repulsive force upon all other nodes. These forces cause the graph to expand and prevent nodes from being positioned too close together. The strength of the repulsive force between a pair of nodes is calculated using Equation 22.

$$f_r = \begin{cases} \frac{R}{d^2} & \text{if } d \leq C \\ 0 & \text{else} \end{cases} \tag{22}$$

where R is the repulsion strength constant, d is the distance between the two nodes and C is the force cut-off distance.

The force cut-off distance, C, is used to localise node repulsion. This prevents the diagram from overly expanding in the centre, and helps maintain even edge lengths in periphery line sections. It also contributes to the system reaching a lower energy state more quickly.

6.2.2.2 Edge attraction forces

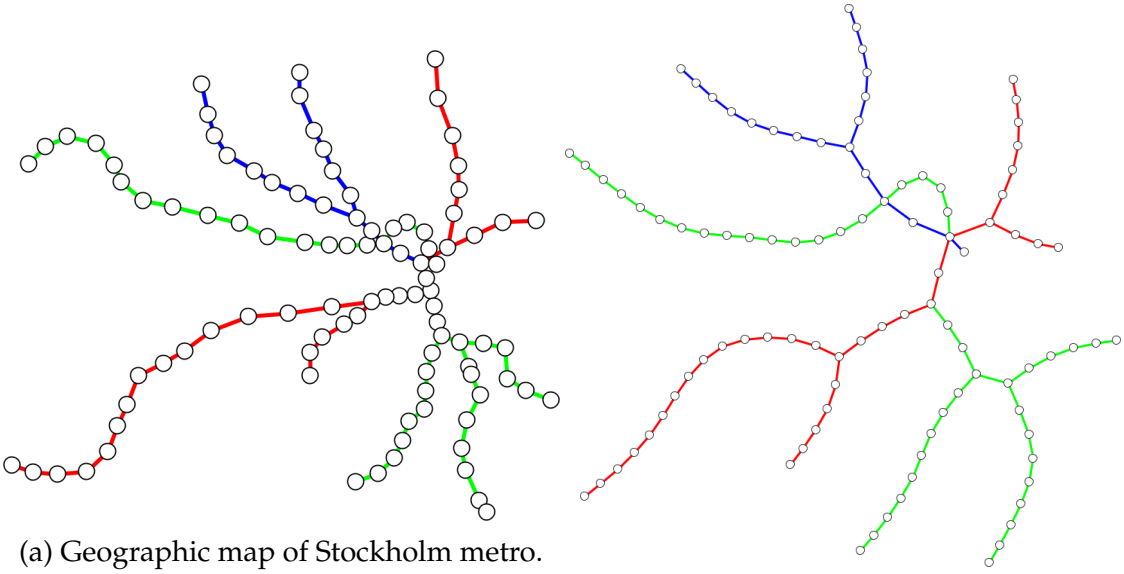
To hold together the outwards expanding graph, attraction forces are used for all pairs of nodes that are connected by edges. The strength of the attractive force between two connected nodes is calculated using Equation 23.

$$f_a = Kd \tag{23}$$

where K is the spring force constant (attraction force) and d is the distance between the two nodes.

6.2.2.3 Summary

The force-directed layout process is iterative, and during each iteration the repulsion and attraction forces are summed for each node. Once all forces have



(a) Geographic map of Stockholm metro.

(b) Force-directed layout applied.

Figure 78: Example of the standard force-directed layout algorithm.

been calculated, each node is moved simultaneously based upon the force acting upon it. This iterative process continues until the energy of the schematic falls below some predefined threshold. The energy of the schematic can be calculated in a number of different ways; our energy is calculated by averaging the force applied to all nodes. In order to prevent nodes with very high velocities from moving large distances in a single step, the maximum distance that a node can move is capped.

Figure 78 shows an example of the implemented standard force-directed layout algorithm. It shows how the forces explained above produce an aesthetically pleasing graph layout, containing a number of desirable characteristics for schematic layout: equal edge lengths, minimal line crossings and minimal node occlusion. However, these forces do nothing to ensure that edges are aligned to octilinear angles. Figure 78 and, unless otherwise stated, all further extensions of the method in the subsequent sections use the following values for force-directed calculations: node repulsion force $R = 7500$, spring strength $K = 0.1$, force cutoff distance $C = 250$, termination energy $v = 0.02$. These values were derived from personal experimentation and consistently produce

appealing layouts for a wide range of schematics.

6.2.3 Automated Label Placement

The produced schematics are required for user testing, and therefore require labels in order for users to perform a number of tasks, including node finding. Labels cannot be arbitrarily placed as this can add unwanted occlusions, making the schematic harder to read. Label position automation provides unbiased label placement between the multiple schematics being used for testing.

The method used for label positioning is adapted from the method previously implemented in SchemaSketch, detailed in Section 4.6.

6.3 Implementation of Octilinearity

This section explains the steps taken to develop our octilinear force-directed layout method. The method is built upon the standard force-directed layout method previously implemented into our graphing application. The Stockholm metro map will be used as an example schematic throughout this section.

6.3.1 Grid Snapping

The initial idea to attempt to enforce octilinearity was to create a grid to cover the schematic, and keep nodes aligned to grid intersections. It is important to note that the force-directed layout algorithm still calculates forces based on the actual position of the nodes, and *not* the visible "snapped" positions. It was known that this method would not solve the problem, but may provide a good starting point upon which to apply further techniques. The result from this step is shown in Figure 79, which clearly shows that this technique greatly increases the octilinearity of the graph.

However, because simple alignment of nodes to grid intersections does not take into account the angle of any edges, there are many edges which are left in non-octilinear orientations. This is particularly a problem with edges whose length is greater than 1.5 grid squares, because if they start off at an angle not

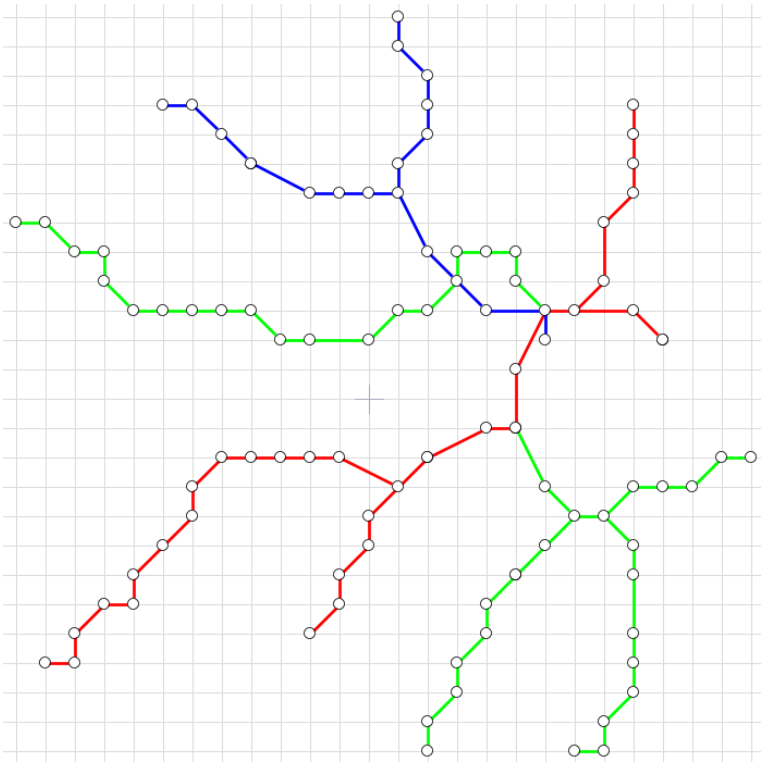


Figure 79: Nodes are aligned to grid intersections.

close to 45°, the connected nodes must move further to result in an octilinear edge. An example of this from Figure 79 has been enlarged and is shown in Figure 80 where edges a, b and c are aligned to grid intersections but are not octilinear.

Another clearly visible issue with this method is that periphery line sections seem to contain many bends. This is caused by the force-directed method producing smooth curves for these sections, as can be seen in Figure 78b, and then attempting to align these to the grid. These periphery line sections indicate that there is clearly a need for additional forces on the graph nodes in order to straighten them out to avoid this effect.

6.3.2 Edge Rotation

In order to solve the issue highlighted in Figure 80, it was clear that additional forces must be added to nodes to enforce the desired edge angles. The method

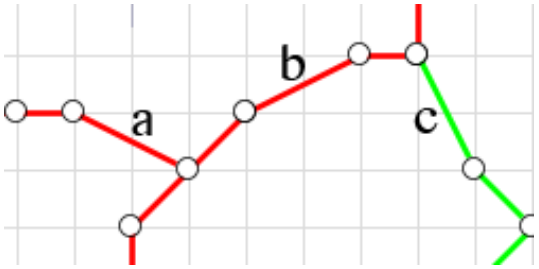


Figure 80: Example of non-octilinearity arising from aligning nodes to grid intersections.

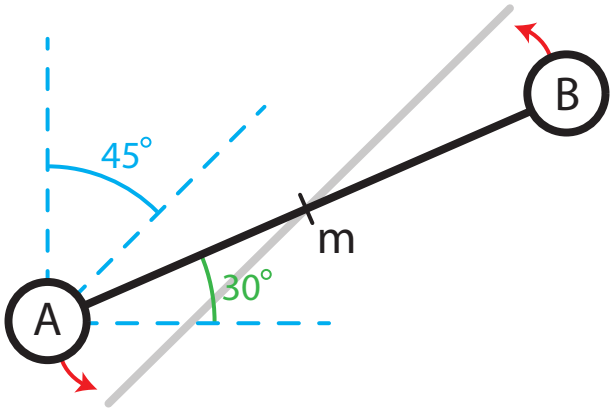


Figure 81: Example edge AB to be rotated to the indicated angle of 45°. This is an anti-clockwise rotation of 15° around m.

implemented to attempt to solve this operates by examining each edge in turn and applying forces to the end nodes in order to rotate the edge to an octilinear angle. This edge rotation technique is based upon the magnetic spring model by Sugiyama and Misue in (Sugiyama and Misue 1995), used to apply an overall structure to a graph – for example a tree with edges mainly in the y-axis. A diagrammatic example of this is shown in Figure 81, and the algorithm is detailed in the following subsections.

6.3.2.1 Pseudocode

For each edge in the graph:

1. `Calculate edge angle` a `using` $a\text{tan2}\,(y, x)$.

2. `Determine closest angular resolution multiple and calculate the rotation angle` θ `required to achieve this` (Section 6.3.2.2).
3. `Calculate edge midpoint` m.
4. `Find target point` A_t `by rotation of` θ° `around` m (Section 6.3.2.3).
5. `Calculate attractive force` f `between points` A, A_t (Section 6.3.2.4).
6. `Apply this force to nodes` A, B (Section 6.3.2.5).

6.3.2.2 Rotation Angle Calculation

Given the current edge angle a, where: $(-180 < a \leq 180)$, and a desired angular resolution r (45° for octilinearity). We find c; the closest multiple of r to a. We can then calculate the required angle of rotation $\theta = c - a$.

6.3.2.3 Target Point Calculation

Given the edge midpoint m and the rotation angle θ. The transformation matrix to rotate a point $p_{(x,y)}$ by θ° around the origin is combined with two translations (to origin, from origin) and simplified to derive equations for the target point coordinated, x' and y' (Equation 24).

$$\begin{aligned} x' &= \big((x - m_x)\, cos\theta - \big(y - m_y\big)\, sin\theta\big) + m_x \\ y' &= \big((x - m_x)\, sin\theta + \big(y - m_y\big)\, cos\theta\big) + m_y \end{aligned} \tag{24}$$

We can now calculate the rotation target point of A, A_t (Equation 25).

$$A_t = \Big(A_{x'}, A_{y'}\Big) \tag{25}$$

As trigonometric functions are relatively slow to compute, performing hundreds of these calculations per second can hinder performance. A simple technique to speed up these calculations is to use $sin\theta$ and $cos\theta$ lookup tables. The range of θ values required is zero to $floor(r/2)$ with increments of one degree. r = angular resolution.

6.3.2.4 Force Calculation

Given points A and A_t, we can calculate an attractive force f between them (Equation 26). This calculation is the same as that used by our standard spring embedder for the attraction between connected nodes and uses the same value for K.

$$f_a = Kd \tag{26}$$

where K is the attraction force constant, and d is the distance between the two points.

6.3.2.5 Applying Forces

In order to calculate the movement from the force, we must first calculate the unit vector v for A, A_t using the following method. This method requires the values for the two points A, A_t, found previously.

```
/**
 * Calculates the unit vector from a to b.
 */
private static Point2f getUnitVector(Point2f a, Point2f b) {
    float dx = b.x - a.x;
    float dy = b.y - a.y;

    // zero case.
    if (dx == 0 && dy == 0)
        return new Point2f(0, 0);

    float div = Math.abs(dx) > Math.abs(dy) ? Math.abs(dx) : Math.abs(dy);
    return new Point2f(dx/div, dy/div);
}
```

We now calculate the additional movement V in both x and y dimensions for node A (Equation 27) and add this to its existing movement. We can also move node B, as its movement is equal and opposite to that of node A.

$$\begin{aligned} V_x &= fv_x \\ V_y &= fv_y \end{aligned} \tag{27}$$

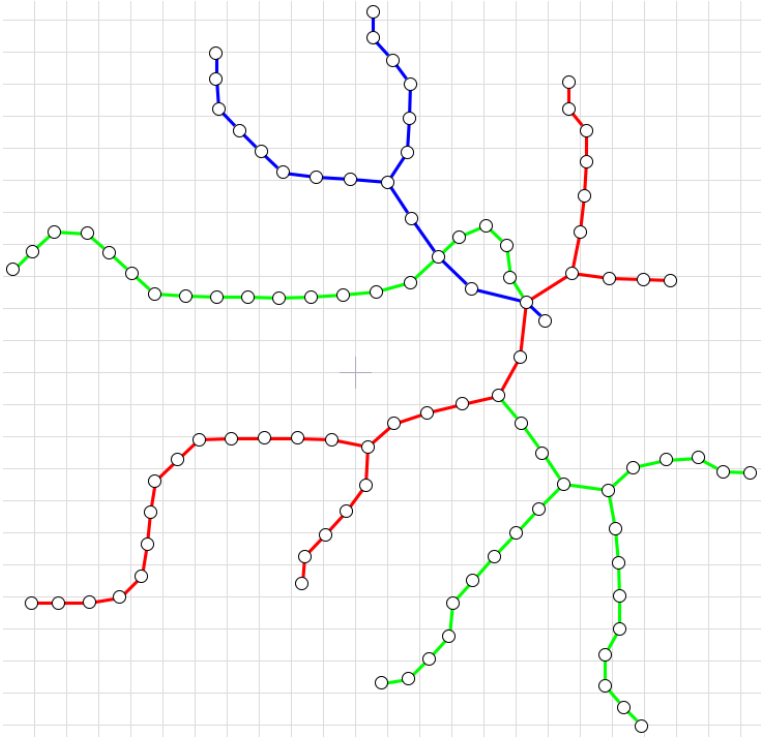


Figure 82: Simultaneous force-directed and edge-rotation forces (1).

6.3.3 Simultaneous Force-Directed and Edge-Rotation Forces (1)

Figure 82 shows the example schematic laid out using both the edge rotational and standard force-directed forces applied simultaneously. We also removed the function that snapped nodes to grid intersections, as we felt that a good level of octilinearity should be achievable by the clever application of forces, rather than superficially repositioning nodes. This method uses the same techniques as, and operates in the same way as the first force-directed octilinear layout method by Hong et al. The level of octilinearity produced by this method is not as strong as desired, and we were unable to increase the strength of the rotational forces without causing the nodes to oscillate back and forth violently – a behaviour that never stabilised.

Various other problems are apparent from examining Figure 82. Long strings of off-angle 2-degree vertices are hard to straighten, as the rotation of one edge to the correct angle puts it at an offset to the next edge; this can be seen in

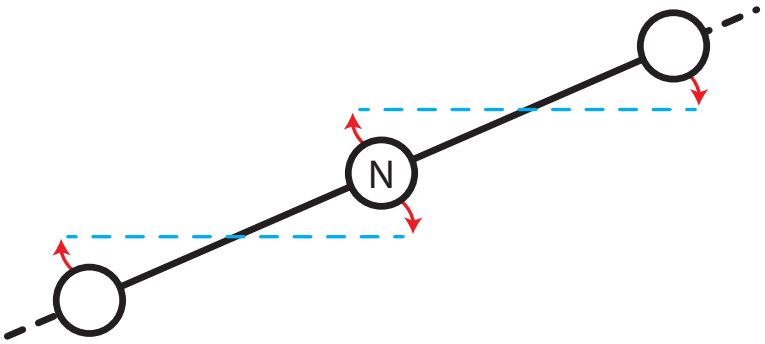


Figure 83: Example line section which is hard to align to octilinearity with edge rotation.

Figure 83 where the blue dashed lines indicate the ideal angle for each edge. This in turn causes the middle node N to oscillate, as it is being pulled in both directions by opposing forces.

6.3.4 Simultaneous Force-Directed and Edge-Rotation Forces (2)

One of the main issues with the method described in the previous subsection is that strings of 2-degree nodes have a hard time aligning themselves due to each edges' desired alignment position being offset from the next (Figure 83). An idea to solve this was to use a common simplification technique of removing 2-degree nodes, and calculating/applying rotational forces only to the remaining nodes.

Figure 84 shows a test of this technique. The red lines indicate sections in which 2-degree nodes have been removed, and therefore the edges to which only rotational forces have been applied (red edges do not count as a connection between nodes, and therefore do not contribute to the attraction forces acting on their connected nodes). All existing nodes are only under the influence of the standard force-directed layout. The method showed promise, as the graph composed of red rotational sections has much stronger octilinearity. One possible direction from this would be to apply attraction forces to the nodes that pull them towards their associated rotational edge. However, problems arose when we attempting this as the standard layout forces conflicted with any other forces applied to the nodes. This caused the nodes to settle at a position halfway

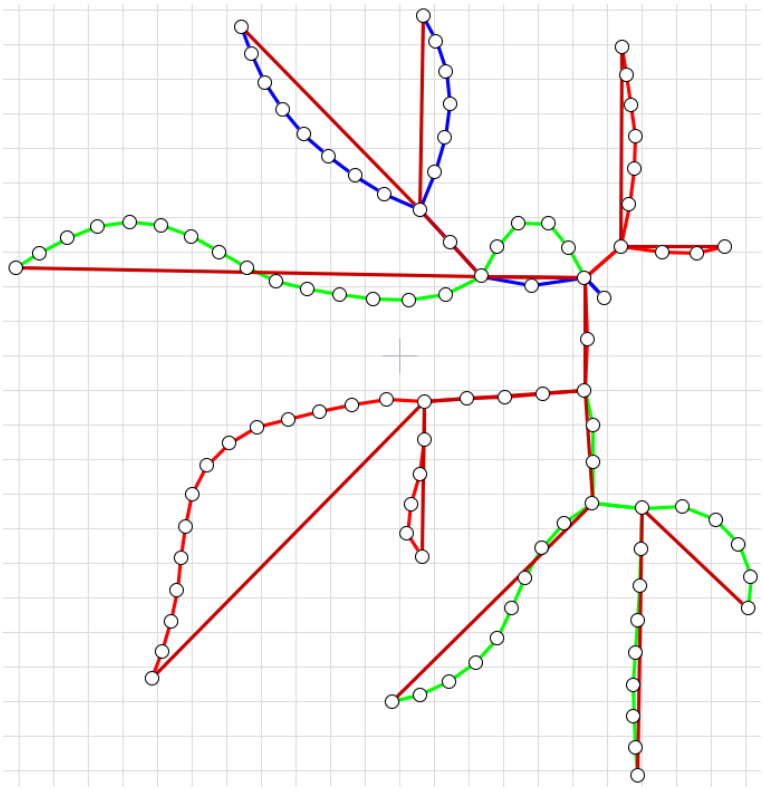


Figure 84: Simultaneous force-directed and edge-rotation forces (2).

between that shown in Figure 84 and the nearest red line – the position at which the opposing forces cancel out.

6.3.5 Sequential Force-Directed and Edge-Rotation Forces

From our two previous attempts it became apparent that applying both types of force simultaneously was causing a number of problems due to the fact that the different forces were conflicting. The different force types would often attempt to move nodes in opposite directions, leading to the resultant force moving the node into an undesirable position which satisfies neither criterion. We therefore decided to experiment splitting apart the two types of force, and applying them sequentially. It was known that octilinearity forces must be applied last, as this criteria must be fully met – even at the expense of standard force-directed layout forces. A similar two stage approach is employed in (Wang and Chi 2011) using

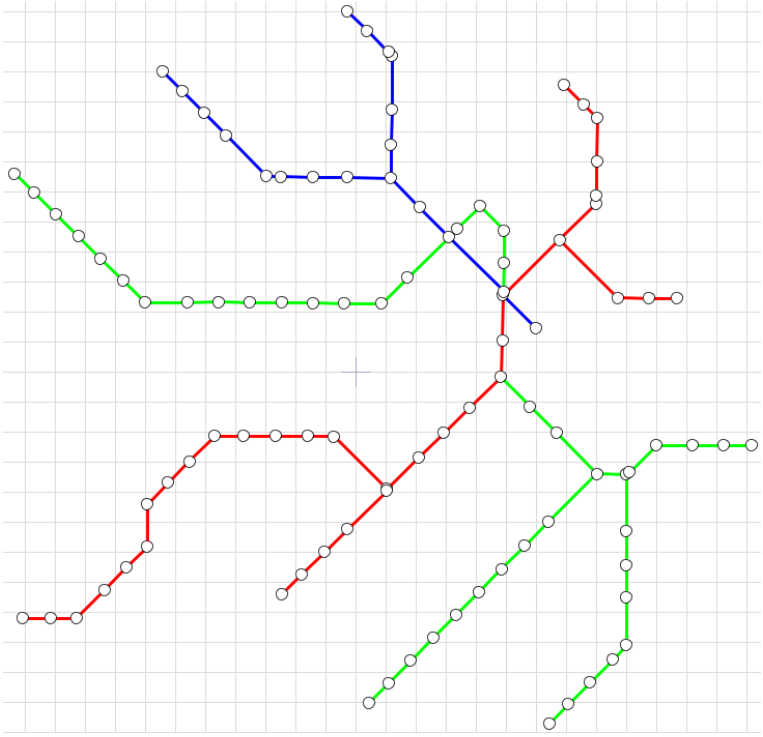


Figure 85: Sequential force-directed and edge-rotation forces.

a linear programming method. Figure 85 shows the result of applying first the standard force-directed layout method until stable and then applying rotational forces to each edge.

Although this method produces octilinear graphs, new problems arise from the two stage process. Because the original forces (node repulsion, edge node attraction) are no longer applied in the second (rotational) stage, there are no forces to stop nodes unevenly distributing themselves or overlapping, as can be seen in numerous cases in Figure 85. Aside from these issues, this method produces the most promising results in terms of force-directed octilinear graph layout.

6.3.6 Semi-Simultaneous Force-Directed and Edge-Rotation Forces

As shown in the previous section it is not desirable to use two discreet stages in the layout process, this is because the node movements performed during the

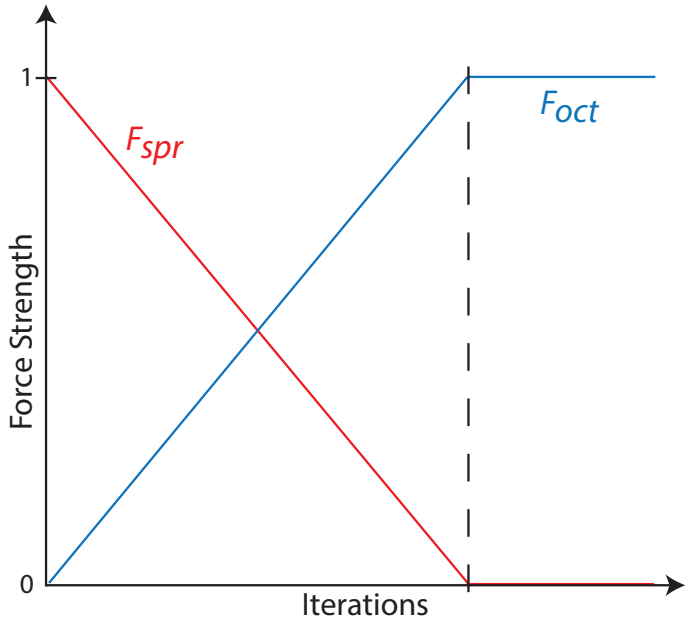


Figure 86: Coefficients F_{spr} and F_{oct} decrease/increase to smoothly transition between the two force types. Octilinear forces continue for a short time after spring forces stop to ensure octilinearity is enforced.

initial stage are undone by many iterations of applying the latter stage forces. However, as shown previously in Section 6.3.3, a solution to this is not as simple as applying both methods simultaneously. This section details a potential method to combine the two sets of forces whilst alleviating previous issues.

Both force generating methods produce a force value (F_{spr} for spring and F_{oct} for octilinear forces) for each node which is used to calculate its movement vector. The strength of these forces can therefore be controlled using coefficients which we name C_{spr} and C_{oct} respectively. Using coefficients on both force generating methods it is possible to reduce the effect of one over time whilst increasing the effect of the other. This allows a smooth transition between the two types of forces as shown in Figure 86.

A complication with this method is that in order to linearly interpolate the value of the coefficients over the duration of the layout process, it is necessary to estimate the iteration at which the energy will fall below a pre-defined value (indicated by the vertical dashed line in Figure 86). In order to estimate this, we must calculate a function based on the graph energy over time; and use this

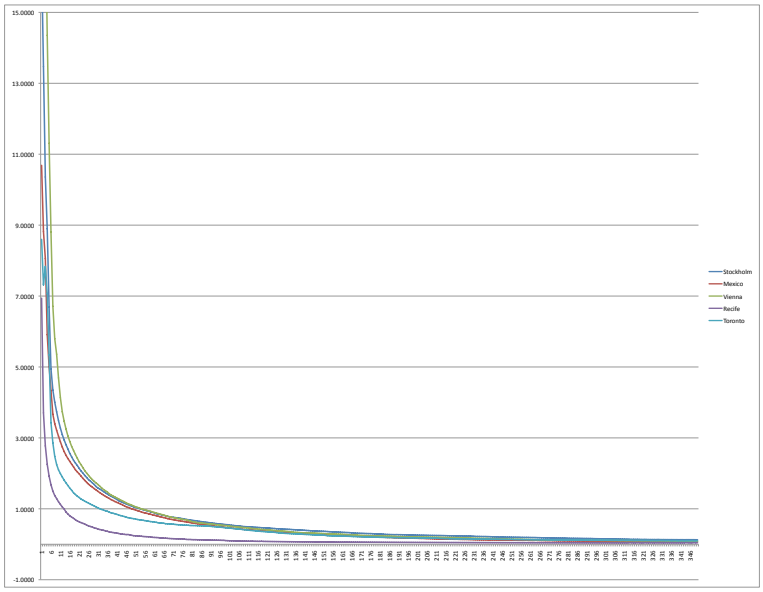


Figure 87: Chart of how average energy (y-axis) varies over the iterations (x-axis) for five example graphs of varying sizes; Stockholm, Mexico, Vienna, Recife and Toronto.

function to predict the iteration at which the system energy will fall below a defined value.

Figure 87 shows that the average energy E_{avg} of a graph over the duration of layout approximates the line $y = \frac{1}{x}$. Accounting for E_{avg} in the equation, we deduce a function to estimate the layout iteration at which the energy level will fall below a pre-defined value v (Equation 28).

$$Iteration = \frac{10E_{10}}{v} \tag{28}$$

Where E_{10} is the average energy on the *tenth* layout step.

The average energy of the tenth layout step E_{10} is used in this calculation. This is because the initial E_{avg} is highly dependent on the initial configuration of the loaded schematic (for metro maps the geographic layout). This can have an adverse effect as some maps (e.g. Vienna) start with a few nodes of very high energy. Allowing the layout an initial ten steps helps to reduce these erroneous values and produce a more accurate estimation function. Before the estimation

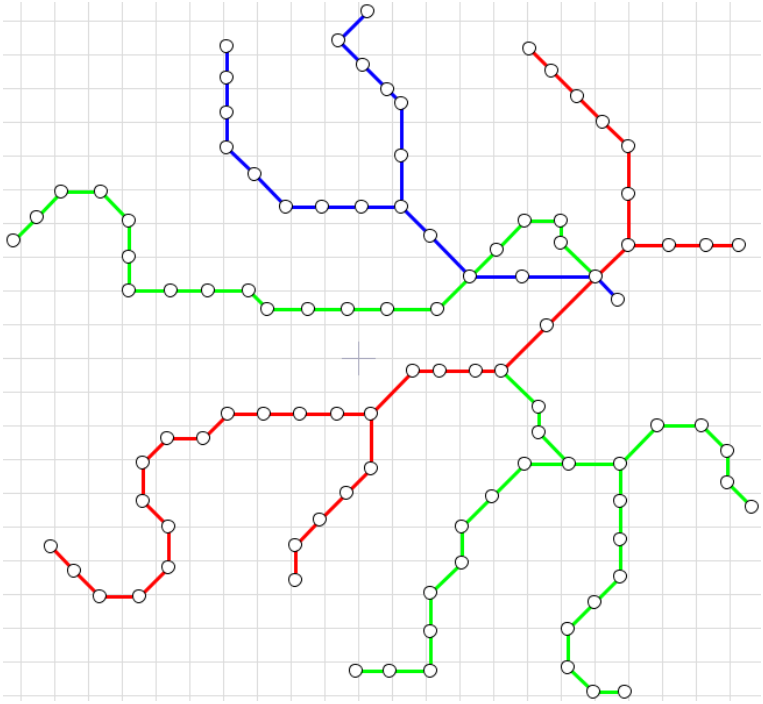


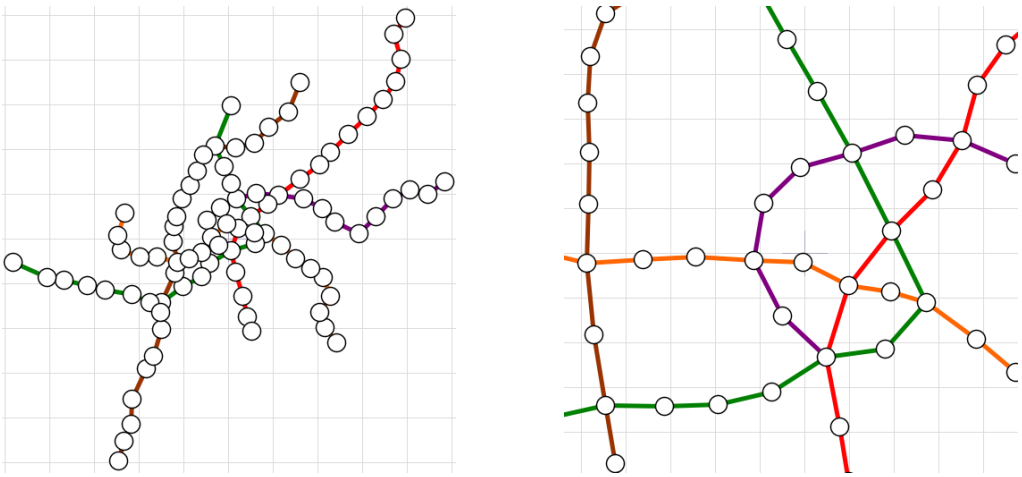
Figure 88: Semi-simultaneous force-directed and edge-rotational forces.

function is calculated we cannot modify the coefficients, and therefore their values during the first ten iterations will remain constant.

With Equation 28 we can calculate an estimated final iteration value, typically between 500 to 2500, and using this value we can linearly interpolate C_{spr}. Calculating C_{oct} is then trivial as it is the inverse of C_{spr} (Equation 29).

$$C_{oct} = 1 - C_{spr} \tag{29}$$

Figure 88 shows the result of our semi-simultaneous method. An improvement in the consistency of edge lengths can be seen and there are no longer nodes pushed on top of each other. One negative aspect of this method is that lines, especially at peripheries, appear less straight than before. This is partly caused because this method does not apply the basic force-directed method forces for as long as before, but could also be a side effect of the initial layout and behaviour of force-directed methods.



(a) Initial graph size.

(b) Size after layout (section).

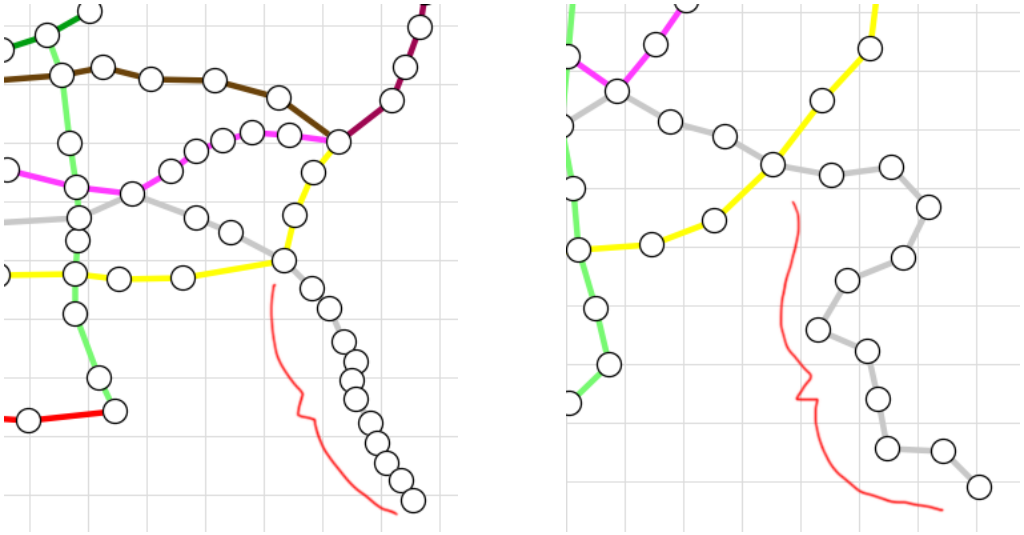
Figure 89: Comparison of the size difference between an initially loaded graph and the post-layout graph. Because nodes are so close in Figure 90a, initial strong repulsive forces cause the graph to 'explode'. Note: the grid resolution remains unchanged.

6.3.7 Schematic Resizing

From our tests, we knew that the initial layout of a schematic has a large impact on its final layout when using force-directed methods. Taking this into account, we noticed that when the maps are initially loaded they were very small compared to the size post-layout (Figure 89). This caused the graph to "explode" outwards in the initial iterations as it resized itself to adjust to the magnitude of the layout forces.

Metro maps, which commonly feature long periphery lines of 2-degree nodes, are particularly susceptible to this initial expansion. Nodes closer to the centre are affected by more forces and therefore move outwards faster than those along the periphery lines, this results in the periphery lines becoming "bunched up" before the octilinearity fitting forces are applied (Figure 90).

A potential solution to this problem is to scale the loaded graphs to a similar size as the final layout. This is done by calculating a scale factor S from the average edge length of the loaded schematic $avgLen\_map$ and the desired average



(a) Initial close nodes. (b) Node bunching caused by expansion.

Figure 90: Node bunching in the grey periphery line caused by rapid graph expansion during the initial layout iterations.

length $avgLen$ (Equation 30).

$$S = \frac{avgLen}{avgLen\_map} \tag{30}$$

All nodes are then scaled by S around the origin (Equation 31).

$$(x', y') = (Sx, Sy) \tag{31}$$

Figure 91 shows the result of the layout starting from a re-scaled graph. The graph was scaled to have an average edge length $avgLen$ of 59.65px – a value derived from the average edge length at the end of layout for four existing maps run without scaling (Table 7). As seen in Figure 91, and quantified in Table 8, performing this schematic resizing before layout considerably improves the overall line straightness when compared to the previous method (Figure 88).

Map	Mean edge length
Stockholm	56.69
Vienna	62.26
Mexico	61.13
Sydney	58.51
Mean	59.65

Table 7: Average edge lengths for maps laid out without scaling.

Line	No-resizing	Resizing
Red	810°	495°
Blue	180°	360°
Green	1620°	900°
All	2610 °	1755°

Table 8: Sum of line bending with and without schematic resizing.

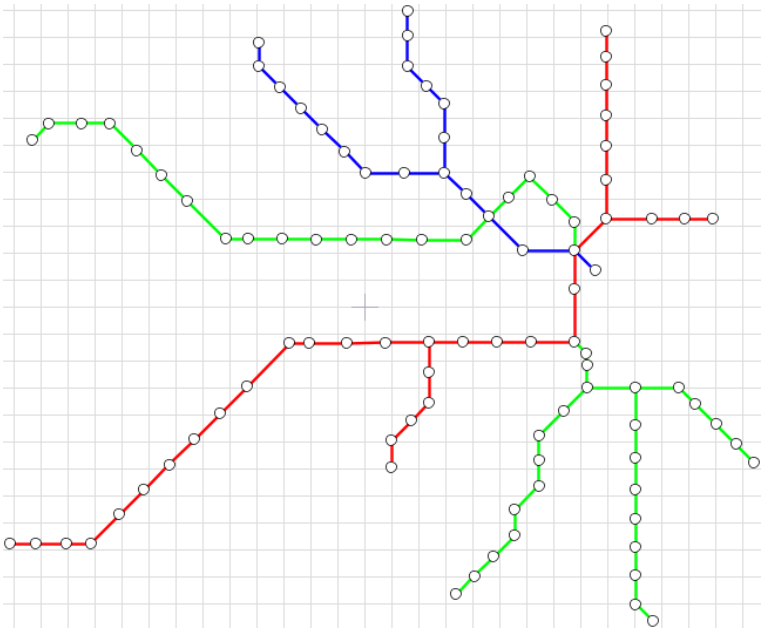


Figure 91: Auto-scaled semi-simultaneous force-directed and edge-rotational forces.

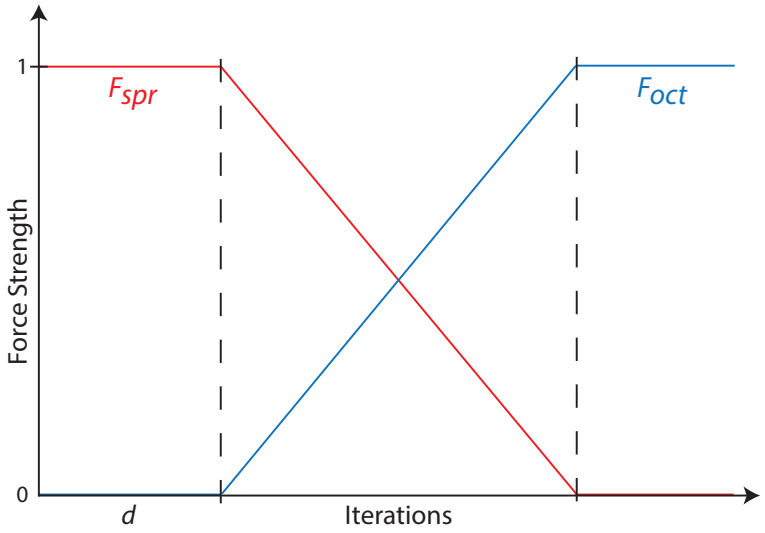


Figure 92: F_{spr} remains stable for d steps before the force transition begins.

6.3.8 Auto-scaled Postponed Semi-Simultaneous Layout

Although auto-scaling the initial graphs has gone some way to helping reduce line bends, it may be beneficial to allow the standard force-directed part of the algorithm to run for longer before starting the changeover to octilinear forces. This may give the lines more chance to straighten out whilst the forces that do this are still in full effect. This modification involves changing the force switchover process shown in Figure 86 to that of Figure 92, where spring forces are applied for a fixed number of iterations (d) before the force transition begins.

If starting the layout algorithm on a graph with a very low energy, it is possible that the last iteration of d (the start of the force transition stage) will be greater than the iteration the interpolator estimates to be the end of the force transition stage. This will result in the layout algorithm never switching to the octilinearity stage and maintaining $C_{spr} = 1, C_{oct} = 0$. To avoid this possibility, when calculating $Iteration$ (Equation 28), we use a modified function which adds the additional constraint $Iteration \geq 2d$ (Equation 32).

$$Iteration = \begin{cases} \frac{10E_{10}}{v} & \text{if } \frac{10E_{10}}{v} \geq 2d \\ 2d & \text{else} \end{cases} \tag{32}$$

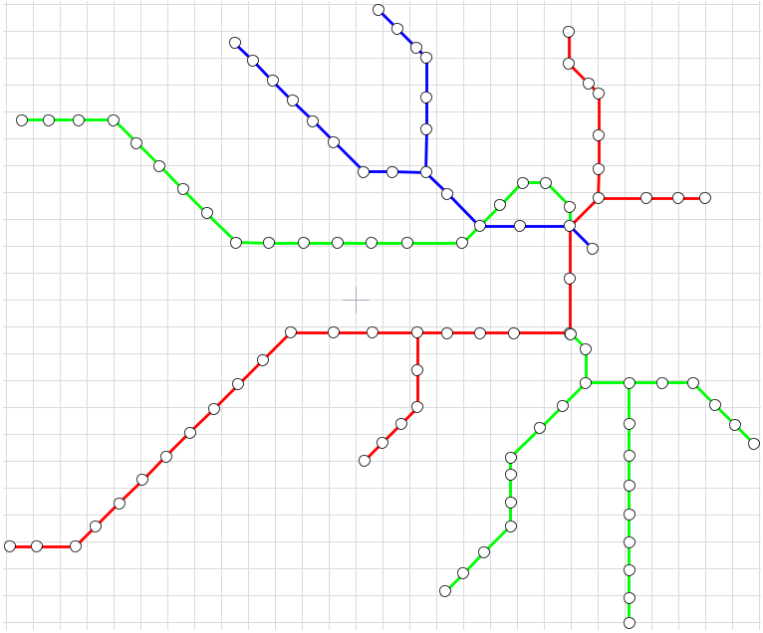


Figure 93: Auto-scaled postponed semi-simultaneous layout ($d = 50$).

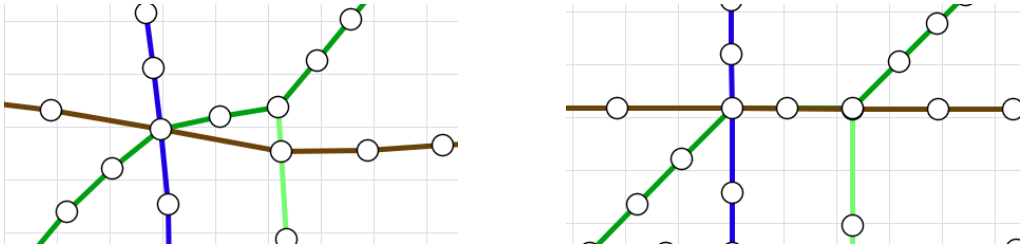
where E_{10} = the average node energy on the tenth step and v = the energy value of which to estimate the iteration of.

Figure 93 shows the result of using this method with $d = 50$. Compared to Figure 91, the method manages to further suppress line bending and produce an acceptable result. However, old problems still exist such as edge length consistency and overlapping of nodes caused by the pure octilinearity force section.

6.3.9 Small Edge Length Forces Using Hooke's law

An example of the node-overlap problem can be seen in Figure 94. This problem is caused by running edge rotational forces without the standard force-directed forces, such as the last stage in our layout method. If two edges share the same desired edge angle, it is possible that the rotational forces will move one over the other.

In order to avoid this, we apply Hooke's law to attempt to maintain a specific edge length during the octilinearity stage. Hooke's law attempts to increase or



(a) The cause of overlap. (b) The resulting overlap.

Figure 94: Example of overlap caused by rotating edges to octilinearity without simultaneously applying standard force-directed forces. Figure 94a shows an example of what causes this, as both the green and brown edges rotate to horizontal and therefore overlap (Figure 94b).

reduce the length of an edge to a specified amount, much like a spring. However, as we only wish to prevent under-length edges the force will only be applied to expand the edge if it is compressed too far. To do this, we apply the force calculated in Equation 33 to both nodes in opposing directions.

$$f = \begin{cases} C_{oct} \times -K\left(D_m - D_c\right) & \text{if } C_{oct} \times -K\left(D_m - D_c\right) \geq 0 \\ 0 & \text{else} \end{cases} \tag{33}$$

where C_{oct} = the octilinearity spring coefficient, K = the spring force constant, D_m = the desired length of the edge at rest and D_c = the current length of the edge.

Figure 95 shows the result of these included forces using $D_m = 50$. It shows that there is no longer an overlap of nodes, but as a result there is a break in octilinearity. However, because the forces only take effect when nodes are very close, this does not affect the octilinearity of the rest of the graph. The ideal situation for this problem would be to move the entire dark-green line to point north-east, but this requires more research into alternate methods to shift the entire line and quite possibly specific functions to handle individual cases rather than the current set of global forces.

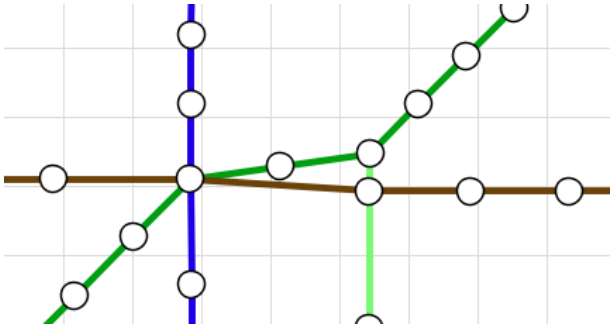


Figure 95: No overlap using small Hooke's law forces ($D_m = 50$).

6.3.10 Fine-Tuning

After personal experimentation, a number of parameters have been fine-tuned to improve the layout. The initial layout phase has been lengthened from $d = 50$ to $d = 250$ (see Figure 92). This change allows the schematic to reach a lower energy using the initial spring-embedder forces, before rotational forces are applied, which further helps straighten line sections.

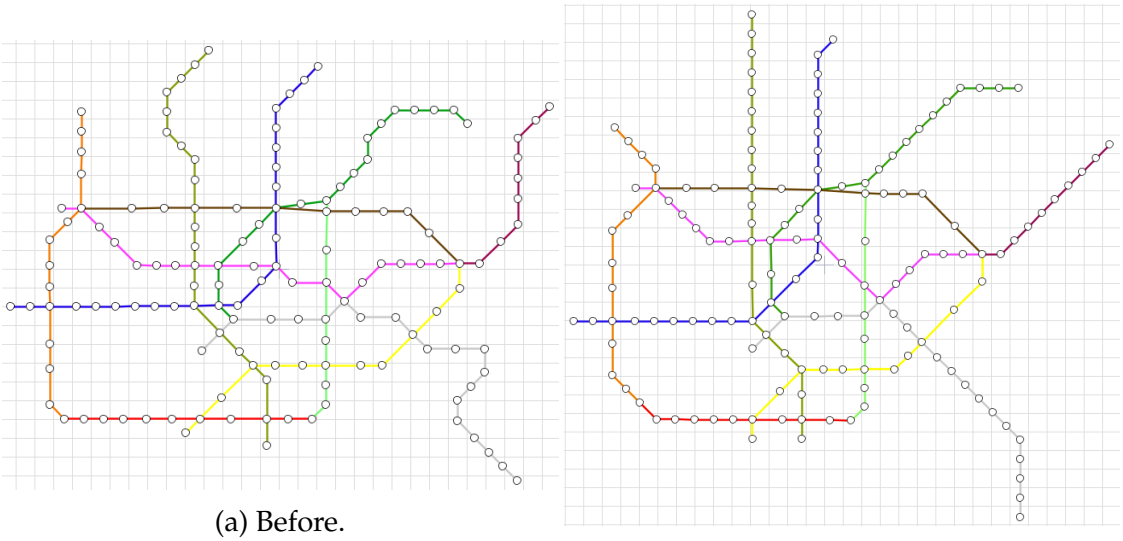
The force cutoff distance during the initial layout stage has been increased from 250 to 400. This cutoff distance is enforced on repulsion between pairs of nodes and stops far away nodes affecting each other. Increasing this value has increased the number of uneven edge lengths at periphery sections, but greatly helps straighten out these sections.

Node repulsion force has been reduced from 7500 to 5000 to compensate for the increased cutoff distance which makes schematics spread out more. The effect of these differences are shown in Figure 96.

6.3.11 Post-Processing to Further Straighten Peripheries

In order to straighten periphery line sections further, we implemented a post-processing method. This method runs after all forces have finished acting upon the schematic.

The method identifies all 1-degree nodes, and traverses the graph from these until a node with degree ≥ 2 is found. This creates a list of all periphery line sections. These sections are then ordered by descending length, as longer lines



(a) Before.

(b) After.

Figure 96: Before and after fine-tuning the Mexico metro; note the straightened periphery sections and slightly fewer bends in the centre.

should be given priority of direction.

Each line is then straightened. The preferred straightening angle is that of the edge in the line section that is connected to the node with a degree of ≥ 2. Using the unit vector of this edge and a desired edge length the projected placement for the end node can be calculated. The proposed straight line can then be checked for the introduction of edge crossings before final placement is made. Providing the proposed edge does not introduce any edge crossings, all nodes in the line section are moved into place. During this process, we also redistribute stations along the line to be perfectly equidistant.

Figure 97 shows the result of straightening periphery line sections, and quantified results can be seen in Table 9.

6.3.12 Force Switchover Changes

Due to a modification to increase the initial run in period before starting the switchover of forces, it was no longer correct to use a $1/x$ based estimation function to determine switchover duration. Also, a fixed run in period was not sensible as this started the switchover at varying energy states dependent

Line	No post-processing	Post-processing
Orange	135°	135°
Pink	360°	270°
Blue	135°	135°
Red	0°	45°
Brown	45°	45°
Yellow	135°	180°
Dark Green	315°	225°
Light Green	45°	45°
Grey	540°	225°
Maroon	135°	45°
Green Ochre	225°	90°
All	2070°	1440°

Table 9: Sum of line bends with and without post processing - Mexico.

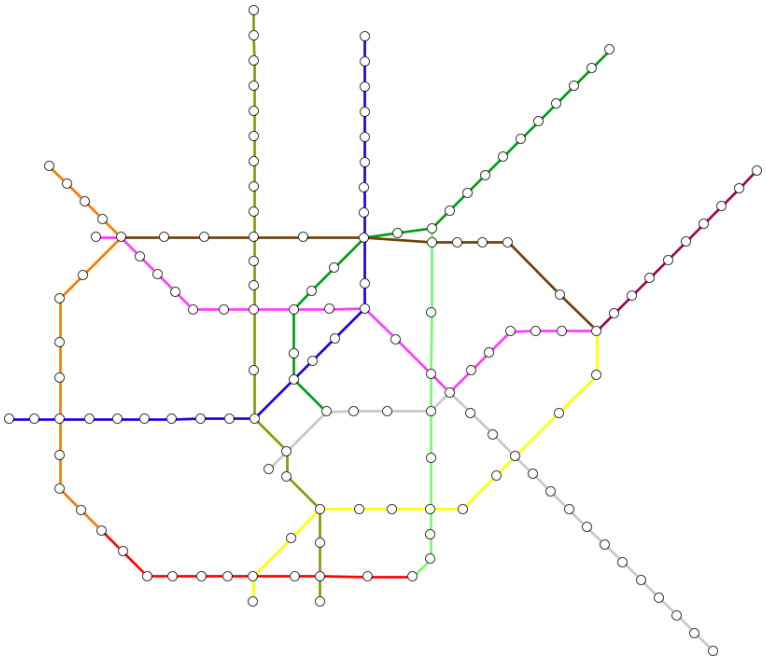


Figure 97: Mexico City schematic with straightened peripheries.

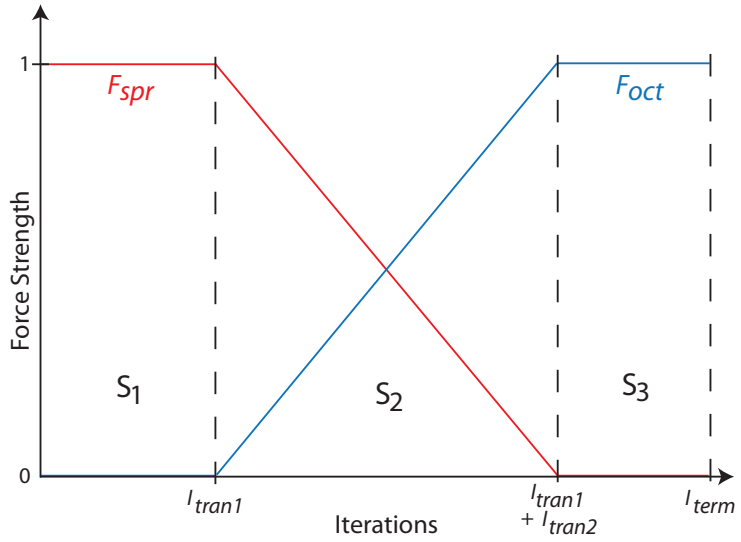


Figure 98: Force constants remain stable in phase S_1, then transition between the two forces in phase S_2. Stage S_3 then ensures octilinearity is enforced by only applying octilinear forces.

on the schematic. Figure 98 shows the updated, final, force-transition graph for the method. The run in stage S_1 has been modified to last until the energy level falls below a threshold (default $= 0.15$). At this point the second stage S_2 begins. S_2 will now run for 200 iterations, instead of being dependent on an estimation function, which produces acceptable results on all tested schematics. The final stage S_3, purely octilinear forces, will remain unchanged and run until the energy falls below the termination energy level v.

6.4 Alternate Resolution Angular Layout

Along with optimising for octilinearity, we discovered that our algorithm is also capable of aligning edges to any desired angular resolution. For example, Figure 99 shows the Stockholm metro map optimised using a desired edge angle of 60°. Using alternate angular resolutions can be effective on certain schematics, such as on Stockholm, but this is highly dependent on the schematic structure and requires experimentation.

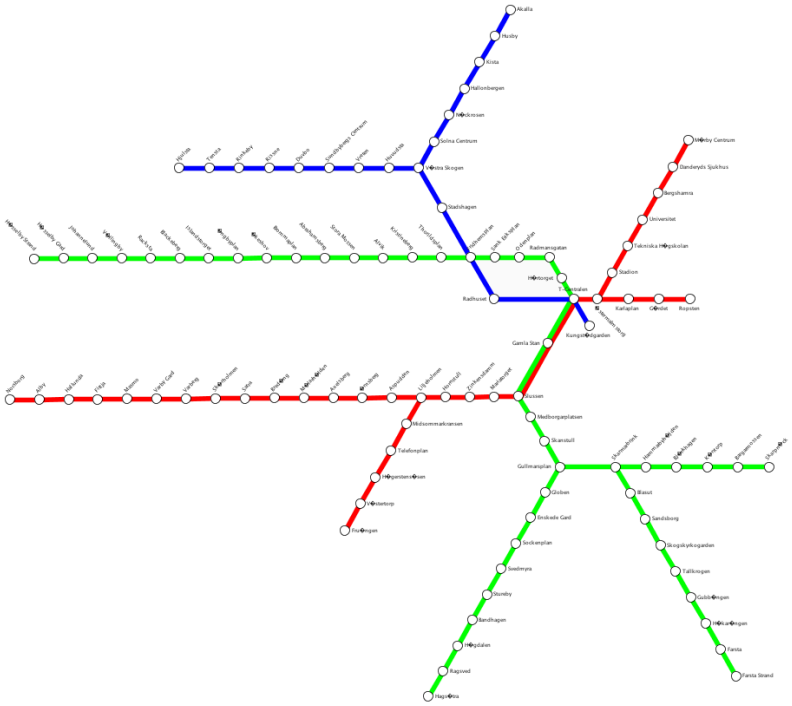


Figure 99: Stockholm metro optimised to a 60° angular resolution.

6.5 Results

Figures 100 and 101 show the Mexico City metro map as laid out by our multi-criteria and force directed methods respectively, Figures 102 and 103 show the Vienna metro map, and Figures 104 and 105 show the Sydney metro map. These layouts show that octilinearity is consistently strongly enforced, along with other criteria commonly associated with schematic layout. Due to the way in which force-directed methods function, many criteria that must be specifically implemented into a multi-criteria approach are naturally occurring. These criteria include the even distribution of nodes, minimal line bends and minimal occlusions; all of which can be seen in the example layouts.

6.6 Summary

In this chapter we have shown the application that we developed in order to implement our new schematic layout technique, as well as describing the steps taken during the development of our method to enforce octilinearity in a force-directed layout.

Our new layout technique is able to produce results of a significantly higher quality than the previous technique by Hong et al. (Figure 28), as we are able to much more strongly enforce angular resolution of edges along with equidistant station spacing. We further believe that our results are comparable to those produced by the current alternate methods such as multi-criteria hill climbing and linear programming.

Additionally, our method is able to produce these layouts much faster than typical search-based techniques. Table 10 shows timing comparisons between our final force-directed octilinear layout and our speed-optimised multi-criteria layout algorithm which we re-implemented as a Java application in order to produce comparable timings. We used a number of real world metro maps for this comparison, and it can be seen that an average speedup of 10 times was achieved. Testing for both methods was carried out on a quad-core Intel Core i5 running at 3.7GHz with 8.0GB 1066MHz DDR3 RAM and using x64 Microsoft Windows 7 Professional with x64 Oracle JRE 7. Further speed improvements are also possible using techniques explained in the introduction to this chapter. We aim to use our method to improve applications of schematic layout in which speed is crucial to a high-quality user experience.

In the next chapter we will extend the methods developed here to implement a form of mental map preservation using a Deluanay triangulation. Mental map preservation is the concept of restricting node movement during layout in order to preserve a user's familiarity with the schematic node positions. We developed this technique in order to use our octilinear force-directed technique for dynamic schematic layout. Dynamic schematic layout is used to visualise data sets which change over the duration of their lifetime, requiring layout methods to be re-applied as the data is modified and when changes are needed to ensure continued ease of readability.

Table 10: Layout timing comparison

Schematic	Nodes	Edges	Force-directed time (sec)	Multicriteria time (sec)	Speedup Multiplier (x)
Recife	28	27	0.02	0.27	13.5
Atlanta	38	37	0.03	0.26	8.67
San Francisco	43	44	0.03	0.83	27.67
Bucharest	44	45	0.04	0.44	11
Toronto	69	69	0.14	0.49	3.5
Washington	86	88	0.14	1.00	7.14
Vienna	90	96	0.37	1.25	3.38
Stockholm	100	100	0.19	0.76	4
Mexico City	147	164	0.92	2.05	2.23
Sydney	173	180	0.59	11.85	20.08
Mean	81.8	85	0.25	1.92	10.12
Standard Deviation	45.71	49.83	0.28	3.35	7.87

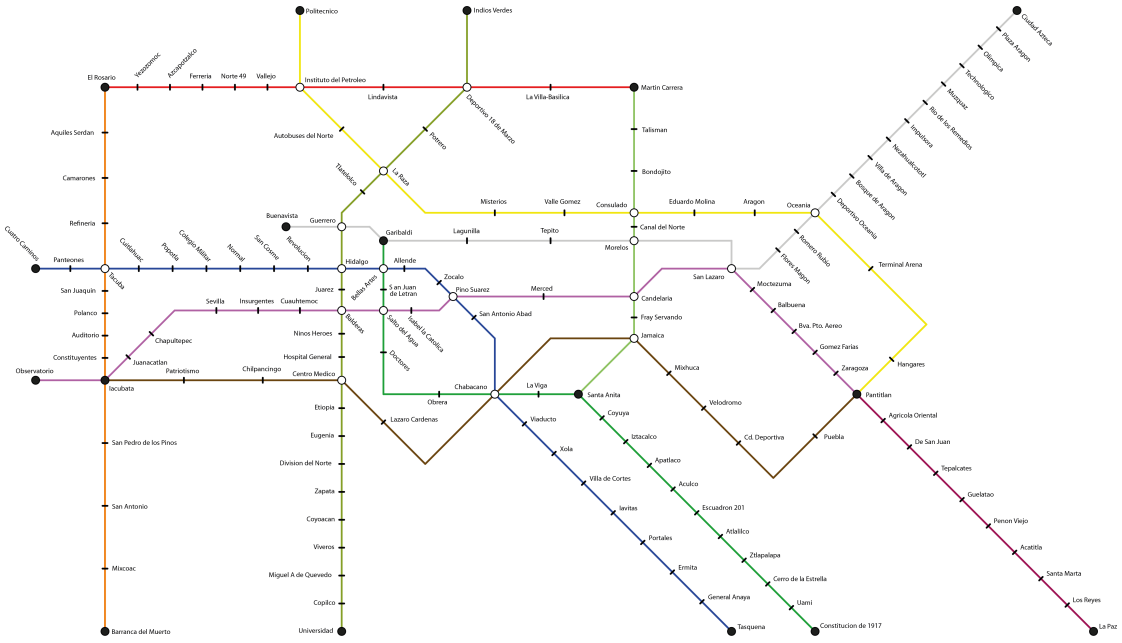


Figure 100: Optimised Mexico City – Multi-criteria hill climber.

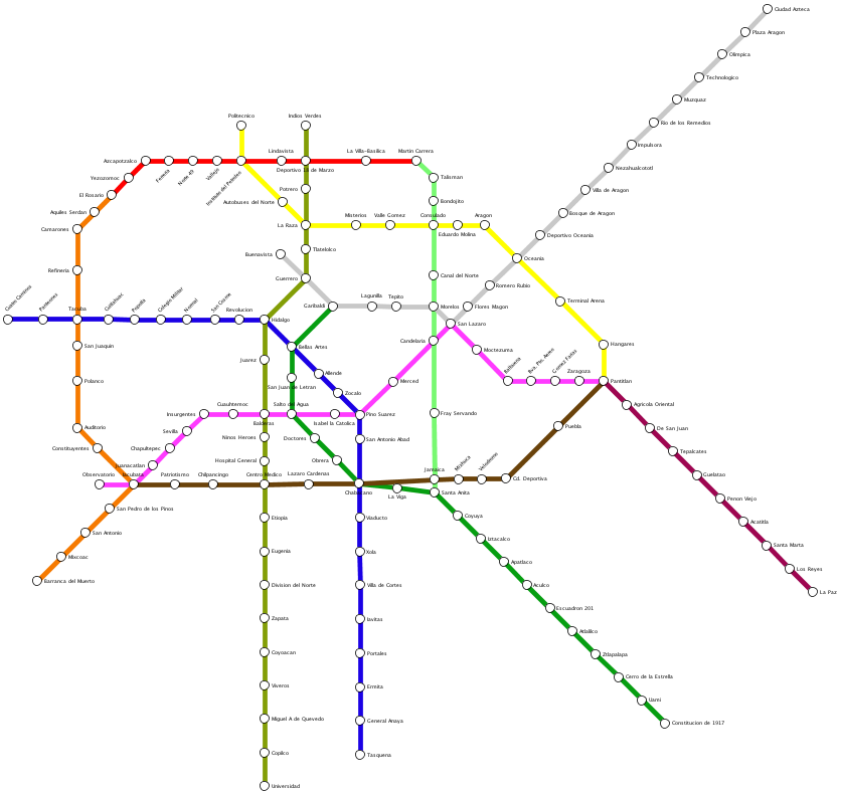


Figure 101: Optimised Mexico City – Octilinear force-directed layout.

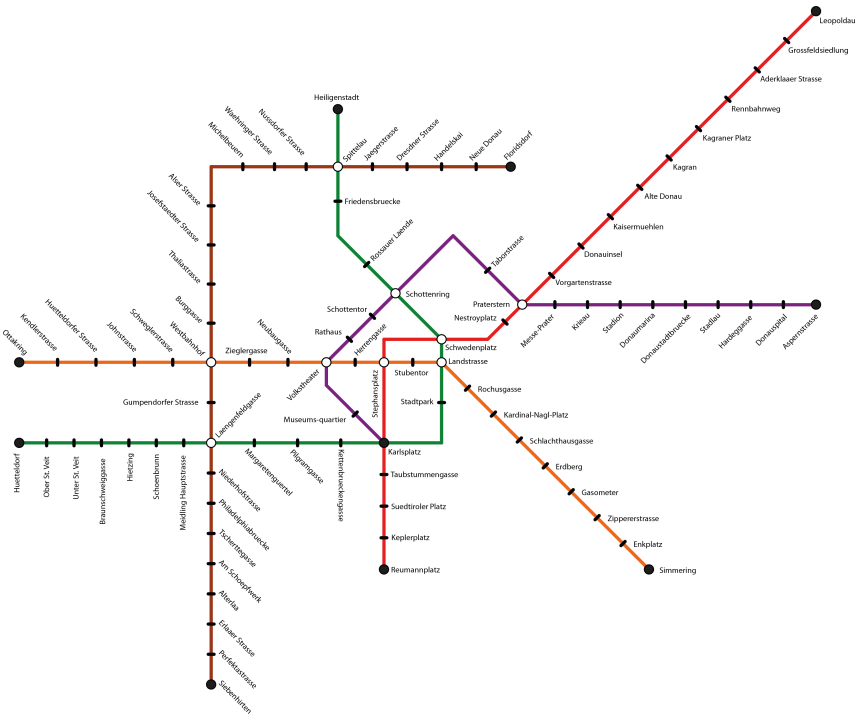


Figure 102: Optimised Vienna – Multi-criteria hill climber.

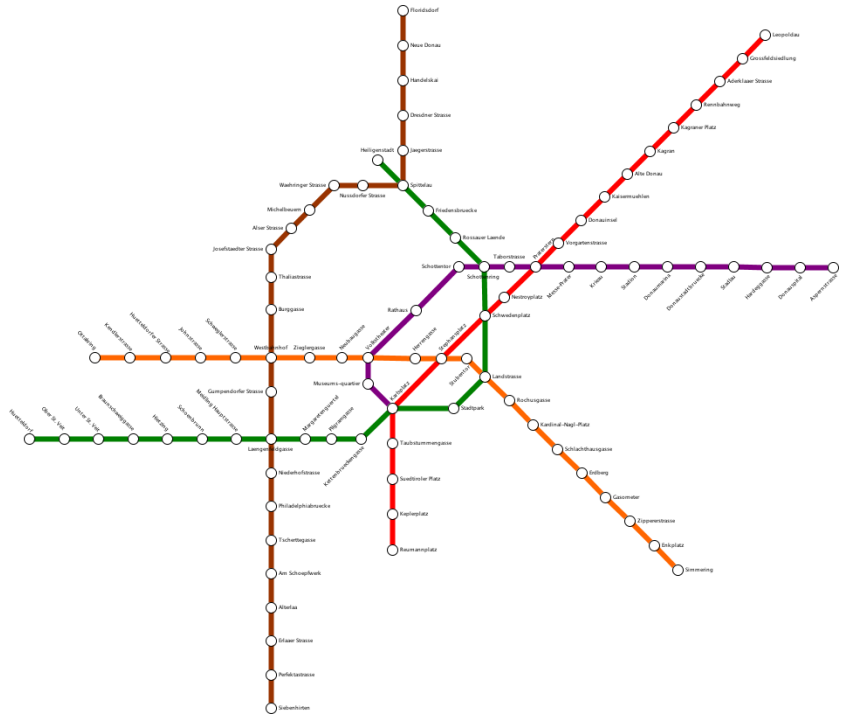


Figure 103: Optimised Vienna – Octilinear force-directed layout.

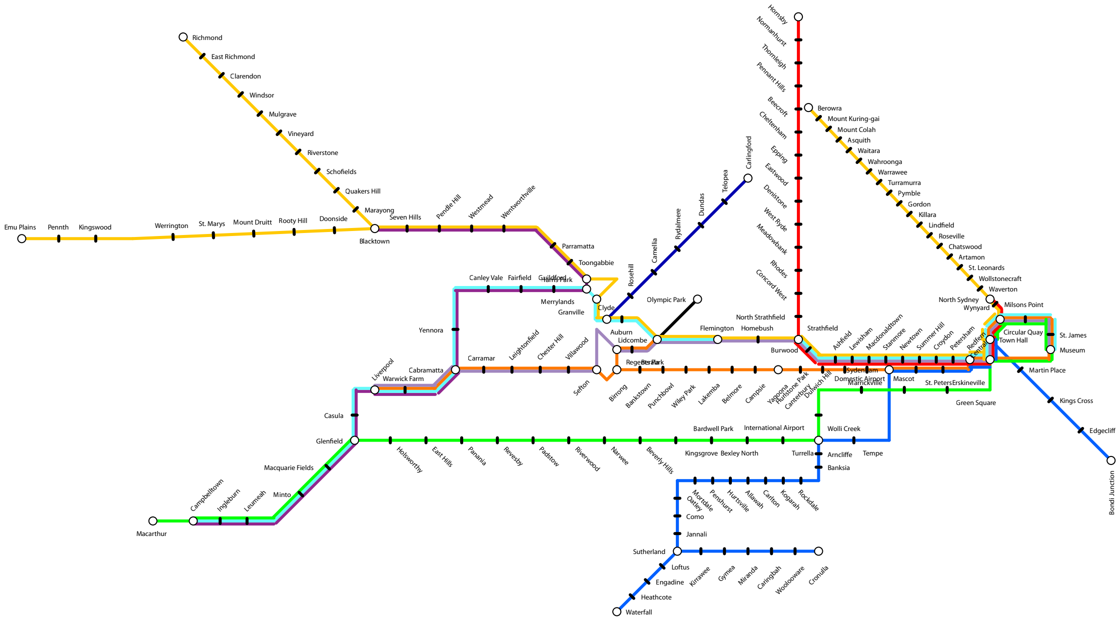


Figure 104: Optimised Sydney – Multi-criteria hill climber.

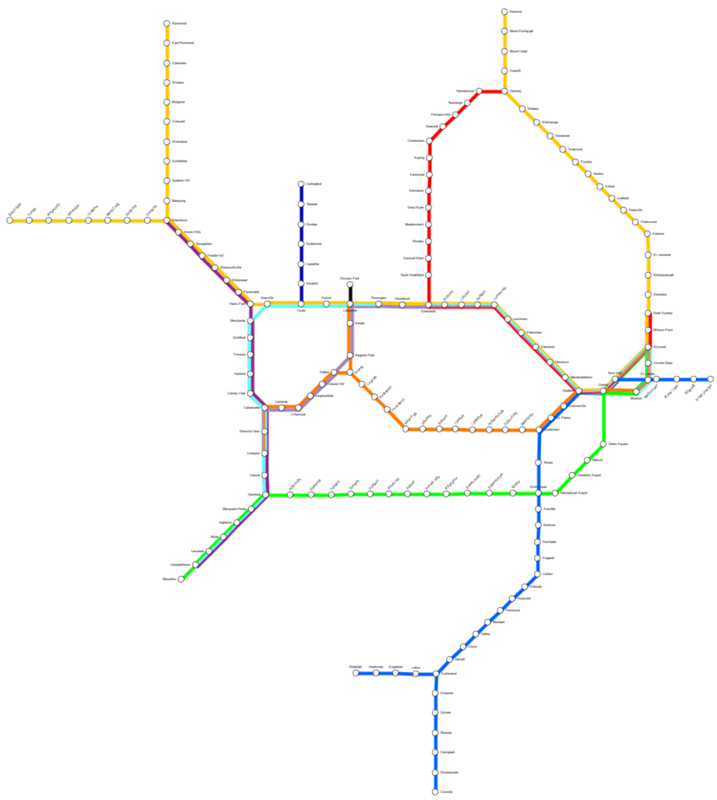


Figure 105: Optimised Sydney – Octilinear force-directed layout.

Chapter 7

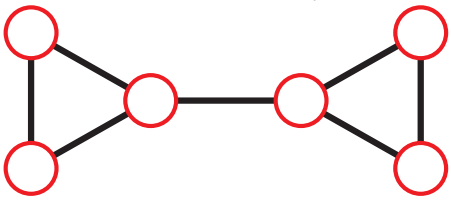
Force-directed Layout with Mental Map Preservation

This section details how a Deluanay triangulation (Delaunay 1934) is used to constrain node movement during optimisation and thus help preserve a user's mental map.

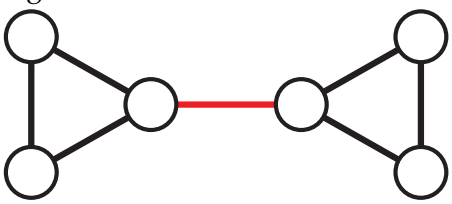
A beneficial property of Delaunay triangulations is that the circumcircle (see Section 7.1.1) of each triangle will contain no other vertices. This property ensures that the interior angles of each triangle are maximised – thereby minimising the number of long, thin triangles. This also extends to 3-dimensions, where the circumsphere will contain no other vertices. This property is especially useful in many applications including 2D/3D interpolation between sample points (de Berg et al. 2008), or polygon-creation when modelling terrain or other objects represented by vertices (Shewchuk 2015). For our purposes it is also the strongest notion of a proximity graph, being the super graph of other proximity graphs such as the Gabriel (Gabriel and Sokal 1969), nearest-neighbour and Euclidean minimum spanning tree graphs.

As mentioned previously in Section 2.2.4, current mental map preservation techniques restrict absolute node movement without regard to proximity relations, and we have not identified any use of a Deluanay triangulation to enforce this. An advantage of restricting nodes by their proximity to other nodes, rather than absolute node movement, is that it more readily allows certain changes to

(a) Schematic with two node clusters separated by an over-length edge.



(b) Absolute positioning effect.



(c) Proximity positioning effect.

Figure 106: Red objects indicate those that negatively affect the similarity measure in absolute (106b) and proximity (106c) based similarity calculations.

a schematic which can be argued do not inhibit mental map preservation, and therefore should be possible – for example the movement of a cluster of nodes. Figure 106 shows a series of three figures; Figure 106a shows an example of a schematic with two distinct node clusters separated by an over-length edge. Our layout method handles such a situation by producing the schematic as seen in Figures 106b and 106c. Such modifications, which preserve the proximity of nodes within a cluster and move entire clusters together should not have a significant negative effect the users' mental map; however, depending on cluster size, these movements will have a large effect on the similarity measure when calculating based on absolute node positions. The red objects highlighted in Figures 106b and 106c indicate those that negatively affect the similarity measure in each case, absolute and proximity, respectively.

7.1 Delaunay Triangulation Calculation

We use the Bowyer-Watson algorithm for calculating a Delaunay triangulation. The method was developed simultaneously by Adrian Bowyer and David Watson and published in the same issue of The Computer Journal – these papers

are (Bowyer 1981) and (Watson 1981) respectively.

The algorithm uses an insertion method to add each node in turn until all nodes are added and the triangulation is complete. A frame is first constructed using four invisible nodes at each corner of the schematic, and two corner nodes are joined to create two large initial triangles as shown in Figure 107a.

Each triangle is represented by a Triangle object. When each is constructed the circumcircle of the three corners is calculated (Section 7.1.1) and stored. The circumcircles of the two initial triangles are shown in Figure 107b – they happen to share the same circumcircle so only one circle can be seen.

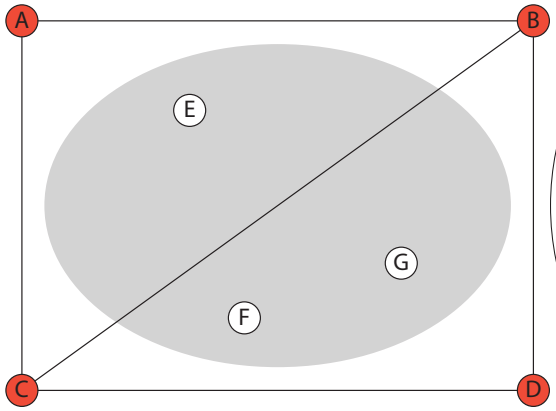
After initial setup, the triangulation is calculated as follows:

1. `For each non-frame node` n.
2. `Identify all triangles for which` n `is contained within their circumcircle.`
3. `Construct a perimeter around` n `by combining all edges from all identified triangles. Remove all instances of any edges which are present more than once` (Figure 107c – red edges removed).
4. `For each perimeter edge` e `of the polygon in which the node exists, construct a new triangle` $e_{start}e_{end}n$ (Figure 107d).

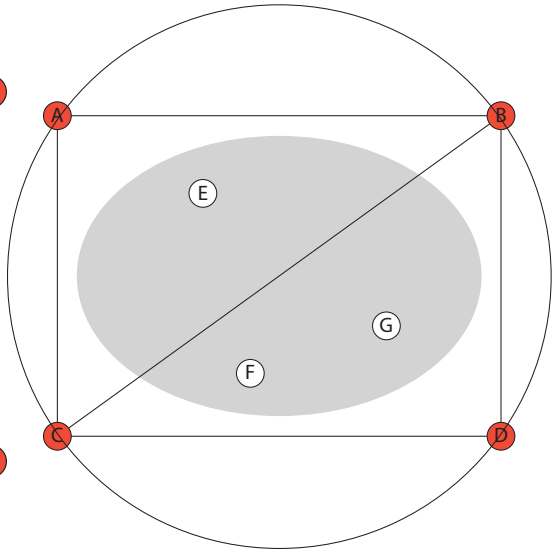
The triangulation is complete once all nodes have been added via this method (Figures 107e and 108). The four corner nodes that make up the frame are left in place, however they are not part of the schematic and can be thought of as anchors as they are unable to move (not being affected by any layout forces) – this has the beneficial effect of preventing strange behaviour caused by the Delaunay frame not being connected to anything, such as rotation or translation of the entire schematic.

7.1.1 Circumcircle Calculation

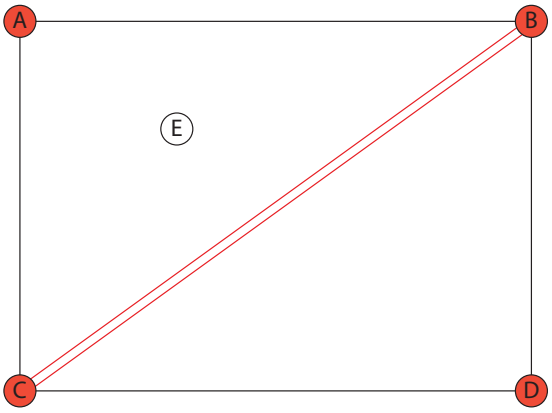
Any three points in a 2D plane lie on the circumference of a single circle known as their circumcircle. Given three points $p_1p_2p_3$ we can calculate the centre c and radius r of their circumcircle as follows:



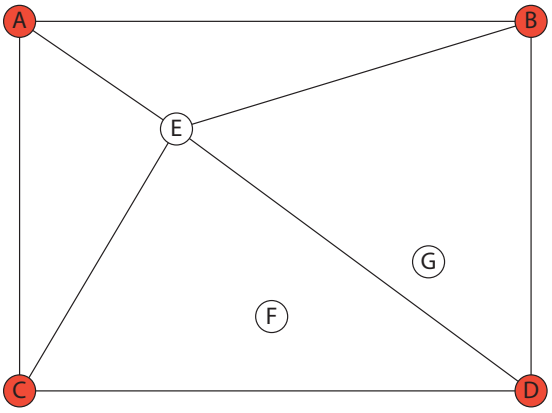
(a) Step 1



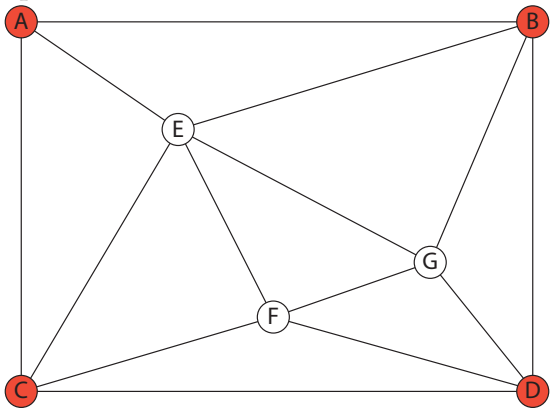
(b) Step 2



(c) Step 3



(d) Step 4



(e) Step 5

Figure 107: Construction of a Delaunay Triangulation.

1. Calculate m_{p1p2}, the midpoint of p_1p_2.
2. Calculate the equation of the line perpendicular to $p1p2$ which passes through m_{p1p2}.
3. Repeat steps 1 and 2 for $p2p3$.
4. $c =$ The intersection point of the resulting two lines.
5. $r =$ The distance between c and $p1$.

7.2 Using the Triangulation as a Frame

Figure 108 shows an example of a Delaunay triangulation applied to the Vienna metro map – the original schematic can be seen on top of the light-red Delaunay edges. The generated triangulation edges are then used during layout as a frame to hold nodes in place. Each edge is modelled as a Hookean spring, with a length at rest equal to its initial length, and a spring strength equal to a user defined value, k (further information in Section 7.6). This spring strength value can be varied in order to affect the level of mental map preservation; for example using a low k value will apply weak springs which are unable to hold nodes in place against strong movement forces.

Figure 109 shows an example of how a Delaunay edge affects connected nodes. The blue line section indicates the spring length at rest, and the red sections indicate extension. The force exerted by the edge is calculated using $F = k(d2 - d1)$. The unit vector for AB is calculated and its x and y components multiplied by F to calculate node movement for A. An equal and opposite force needs to be applied to B, so x and y unit vector components are multiplied by $-F$ to calculate its movement. In order to help stop nodes moving large distances in one step, F is capped at 25, which in turn ensures node movement is limited.

When running in our layout method, Delaunay frame forces are calculated and applied last, after the calculation of spring embedder and edge rotation forces. This is to ensure that all node movements from other methods can be counteracted by the mental map preserving frame.

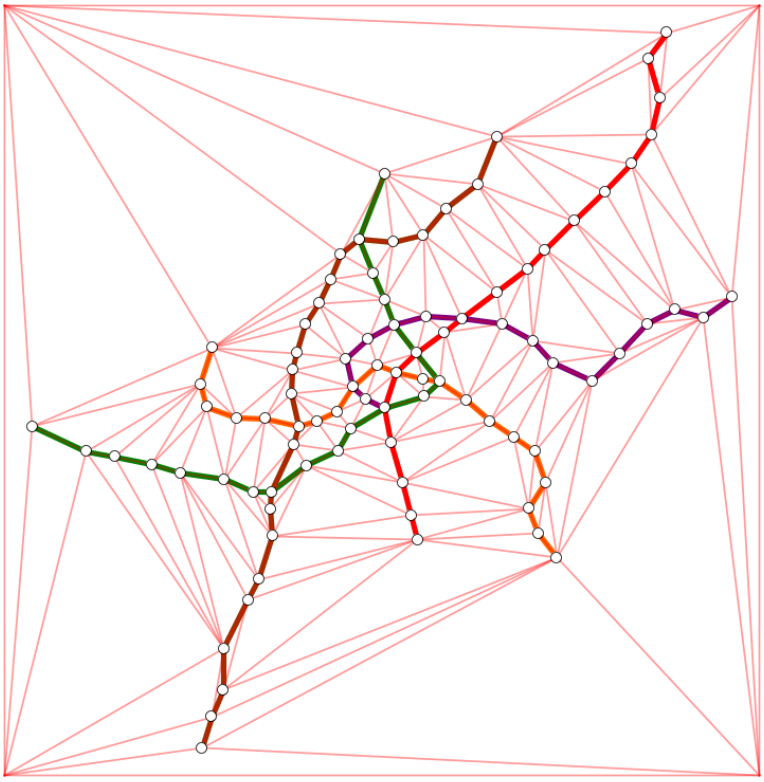


Figure 108: Delaunay triangulation of the Vienna metro map (including triangulation frame).

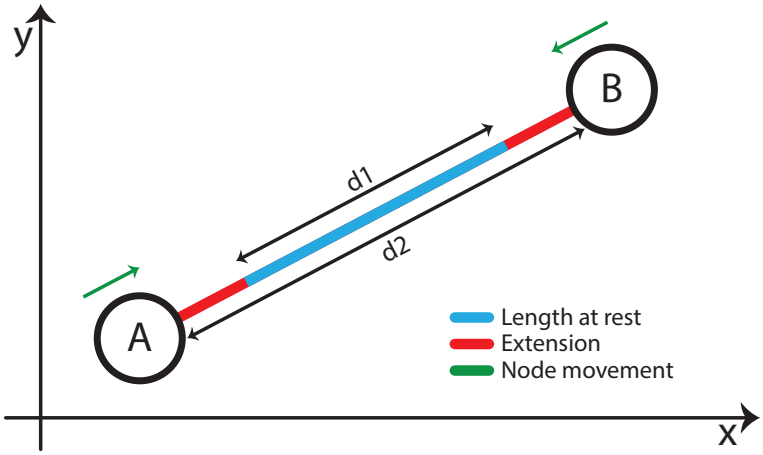


Figure 109: Example of Hooke's law affecting nodes connected by Delaunay edges.

7.3 Combination with Octilinear Forces

In order to allow the Delaunay frame force to be used in conjunction with the existing layout algorithm, it is required that any force produced by the frame is subject to the same force coefficient as the standard spring embedder forces. This is due to the necessity to reduce the force to zero before layout is complete, in order to ensure the angular forces have the desired effect. Delaunay force calculations therefore use a new force coefficient, F_{Del}, which mirrors the value of F_{Spr}. Adding F_{Del} into the force calculation, the result is shown in Equation 34.

$$F = F_{Del}kx \tag{34}$$

where x is the Delaunay edge length displacement and k is the spring strength of the Delaunay edges.

7.4 Node Oscillation in High-Strength Frames

In order to hold nodes in place during the layout phase for 100% mental map preservation (MMP), it is necessary to set a high Delaunay frame strength value k. However, when set to a high value, nodes in the schematic have a tendency to oscillate. This node oscillation happens because the application of forces in a computer program must occur in steps rather than continuously. In our case, using multiple force types, these forces can also only be applied sequentially. This characteristic allows nodes to move without regard to Delaunay frame forces during the initial spring embedder calculations, and move past their stable destinations in a single step. Figure 110 illustrates this point; consider the two nodes A and B_1 connected by both a schematic edge (black) and a Delaunay edge (red). At high levels of MMP the Delaunay edge acts as a very stiff spring with an equilibrium length of 10 units, unable to compress or extend without a large force, thereby holding the two nodes apart. However, the schematic edge is currently overextended and is applying a pulling force to both nodes to attempt to reduce its extension. The resting length for this schematic edge is 5 units – when B_1 is in the position of B_2. These two forces conflict and in an

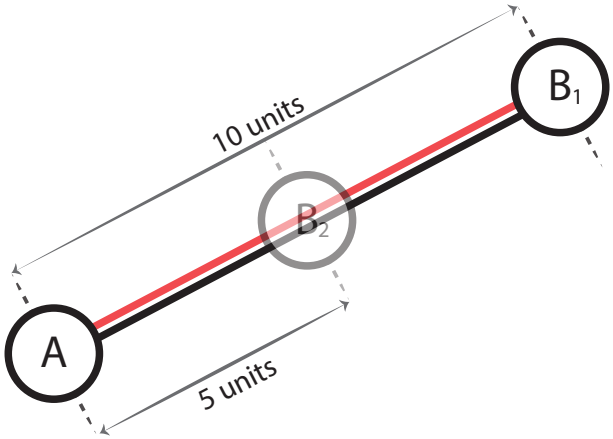


Figure 110: Example of how high frame strength can cause node oscillation.

environment where forces are applied continuously the Delaunay edge would successfully hold the nodes apart due to its much larger spring force. Problems arise from this, however, when forces are applied independently as the schematic edge would initially pull the two nodes closer together, compressing the high-force Delaunay edge. When subsequently calculating the effect of the Delaunay edge, it will exert a large pushing force upon both nodes as it uncompresses, moving one or both nodes past their stable position and extending the edge. This starts the node oscillation.

Node oscillation is also an issue in standard force-directed layouts, and a solution ideal for use in our situation is described by Jamey Lewis in (Lewis 2014). The author uses a force coefficient termed "temperature" to dampen node movements under certain conditions. This node dampening varies from standard dampening in a couple of significant ways. Firstly, the dampening coefficient (temperature) is able to both increase as well as decrease within a bounded range. Secondly, the temperature varies based on the change in velocity of the node, rather than simply decreasing over time. This also results in each node in the schematic having an individual temperature, which is desirable behaviour as each node has a different oscillation force point.

The temperature varies between 0 and 1 with a resolution of 0.1. All temperatures initially start at a value of 0.5, and the temperature is used as a coefficient to any node movement. This means that nodes initially move at half speed, and

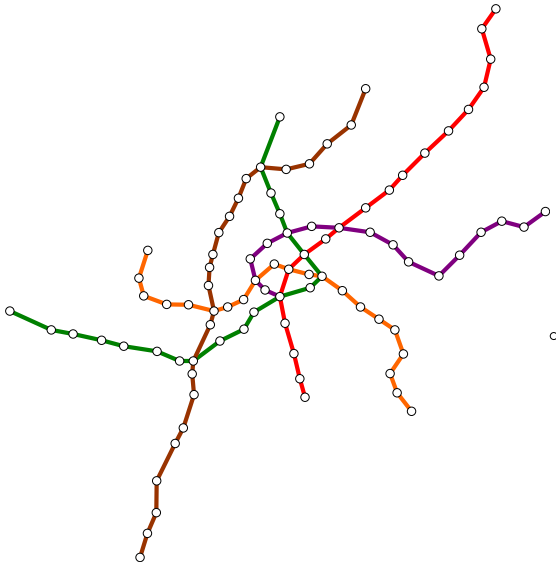
can vary between no movement and full speed movement. To combat node oscillation, the temperature varies based on successive movements in a similar direction. If the node moves with a vector within 45° of the previous movement vector (forwards), the temperature is increased by 0.1. If the node moves with a vector of greater than 135° from the previous movement vector (backwards), the temperature is reduced by 0.1. This has the effect of slowing down oscillating nodes, whilst not penalising normally moving nodes.

7.5 Similarity Method

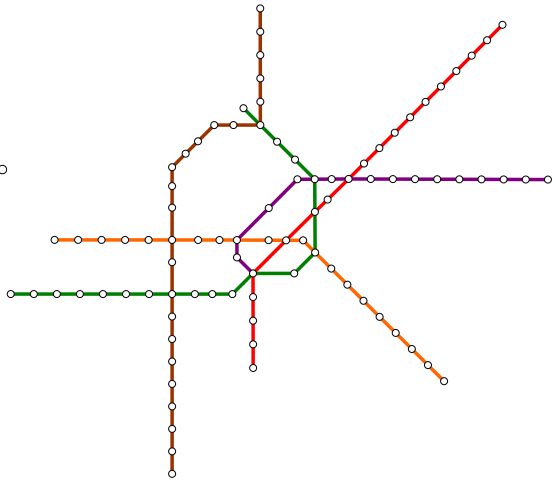
In order to quantify a level of MMP, it is necessary to have a measure of similarity between two subsequent stages of a dynamic graph. This measured value must represent the level of mental map preservation between the two stages. It can be argued that moving a cluster of nodes is not a large change to the mental map (Misue et al. 1995) (providing the relative node positions within the cluster remain unchanged). Therefore the calculation should not be based on original node positions, but rather node proximities. Our Delaunay triangulation edges are well suited for this purpose and therefore we calculate our similarity measure based on the mean edge length difference in these. The value produced is a proximity based measure, representing how nodes have moved in relation to their neighbours. Equation 35 shows the calculation used to produce the value. If no nodes are moved during optimisation this function will return a value of zero, indicating full MMP. Conversely, a high value indicates a low level of MMP.

$$EdgeSimilarity = \frac{\sum_{i=0}^{|e|} abs\left(len\left(e[i]\right) - len_{orig}\left(e[i]\right)\right)}{|e|} \tag{35}$$

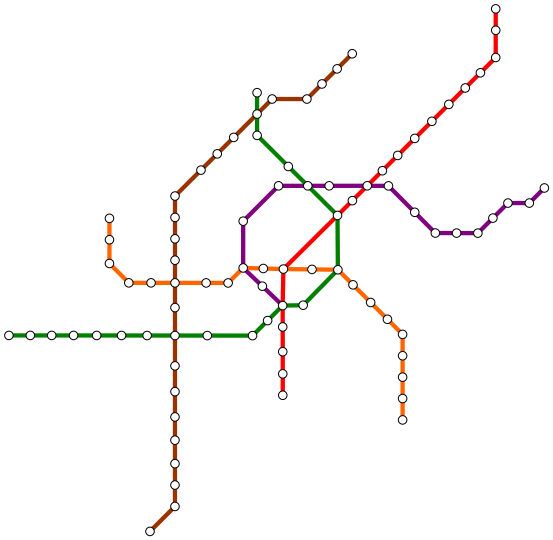
where e is an array of Delaunay frame edges (excluding edges connected to frame nodes), $len(e[i])$ is the current length of edge $e[i]$, $len_{orig}(e[i])$ is the original length of edge $e[i]$, and $abs()$ returns the absolute value of its parameter.



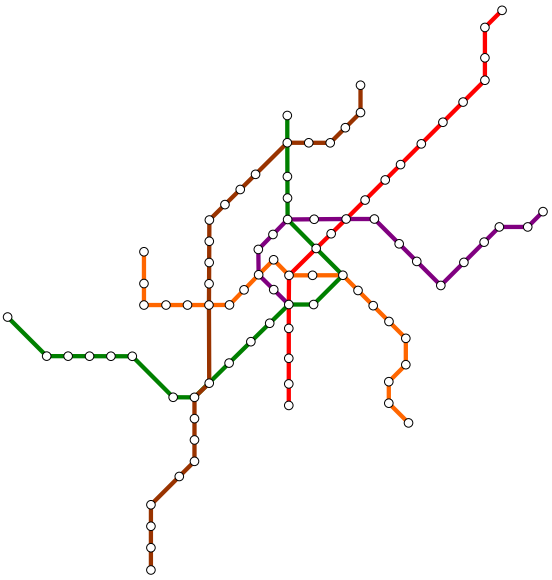
(a) Vienna - Geographic



(b) Vienna - 0% MMP



(c) Vienna - 50% MMP



(d) Vienna - 100% MMP

Figure 111: Vienna schematic optimised using varying levels of mental map preservation. Figure 111a shows the input schematic; Figures 111b, 111c, and 111d have used the linearised slider to set their MMP levels to 0%, 50%, and 100% respectively.

7.6 Frame Strength

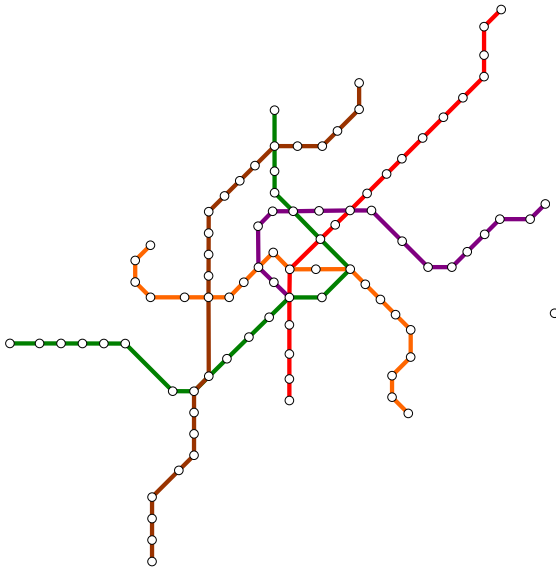
The value of k used when calculating Delaunay frame forces can be manually modified by the user. This allows the user to specify exactly how much force they require the Delaunay edges to exert, altering the layout of the schematic by holding nodes in positions relative to each other. Using a k value of 0 would make no difference to the layout, and produce a result with 0% MMP. In order to achieve 100% MMP, the user must set a high value for k, such as 1.

To achieve a custom level of mental map preservation between 0% and 100% the user must set k correctly. This is difficult for a user to manually perform, as percentages of the maximum frame strength do not linearly map to percentages of the post-layout calculated similarity metric. For example, if 0% frame strength produces a result with a similarity value of X, and 100% produces a value of Y; 50% frame strength will *not* produce a layout with a similarity value equal to $\frac{X+Y}{2}$.

In order to help users easily set a desired level of mental map preservation in the output schematic, we have mapped the value of k to a linear percentage slider on the application GUI. The details of this slider are given in the next section. Figure 111 illustrates varying levels of mental map preservation using our linearised slider with the Vienna metro map and it can be seen that as a higher preservation is used, the schematic retains more of its original features (the purple periphery line section, extending to the right, is a good example of this across all MMP levels). Octilinearity is always enforced, and is the main cause of node displacement during a 100% MMP layout.

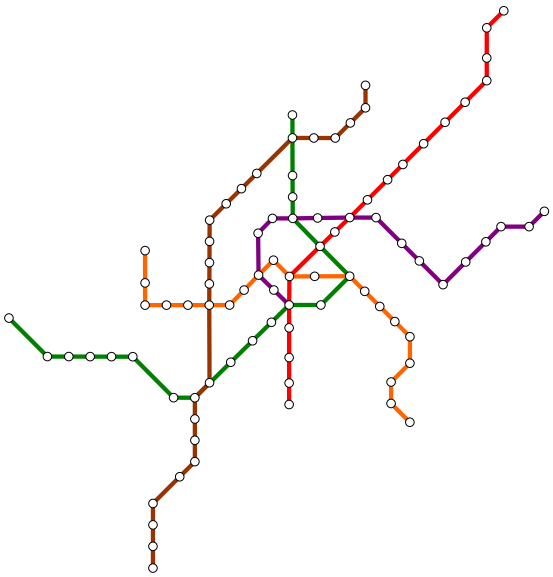
7.7 Linearising the Mental Map Slider

The mental map slider is a user controllable slider in the GUI of our software that can be manually changed to vary the level of mental map preservation desired from the layout method. As mentioned previously, there is not a natural linear mapping between the percentage of max frame strength and the resultant similarity value – the first 20% affects the result far more than the top 80% (Figures 112 and 113 illustrate this). This makes it difficult for users to gauge

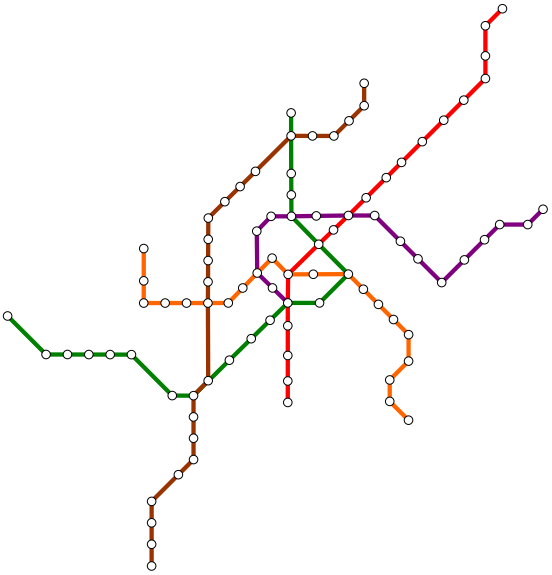


(a) Vienna - 25% frame strength, nonlinear

(b) Vienna - 50% frame strength, nonlinear



(c) Vienna - 75% frame strength, nonlinear



(d) Vienna - 100% frame strength, nonlinear

Figure 112: Vienna schematic optimised at varying percentages of the maximum frame strength value. These schematics show how without linearising the frame strength slider, very little schematic change occurs after the first 20%.

a weak/medium/strong preservation value and it is therefore desirable to linearise the slider scale to allow more intuitive strength setting. To do this, it was required to plot a graph of percentage frame strength against the resultant similarity value after layout (Figure 113). These results were calculated based on 11 example schematics as listed to the right of the figure. It is important to note that only the first stage of layout (standard spring embedder) is being run before the difference method value computed. This is because the force coefficient for Delaunay frame edges, F_{Del}, is reduced over time in line with the spring embedder force constant, F_{Spr}, and therefore not in effect for the latter iterations. The graph is still changing in these latter iterations due to angular forces, but it is not accounting for mental map preservation (due to edge octilinearity being a hard constraint). This results in the edge alignment stage skewing the difference measure, effectively adding unwanted noise to the desired graph. Figure 113 also shows the average similarity line, along with a best-fitting function curve (Equation 36) calculated using Microsoft Excel using a least-squares method.

$$y = -6.541 \ln(x) + 48.723 \tag{36}$$

which rearranges to:

$$x = e^{\frac{y-48.723}{-6.541}} \tag{37}$$

The maximum $(x = 0.01)$ and minimum $(x = 100)$ y values (y_{max} and y_{min}) of the function curve are calculated using Equation 36. This is shown in Equations 38 and 39. These values are required to calculate percentage values between y_{max} and y_{min}, along with the range y_{range} (Equation 40). We cannot use $x = 0$ for y_{max} calculation because $\ln(0)$ is undefined.

$$y_{max} = -6.541 \ln(0.01) + 48.723 = 78.845 \tag{38}$$

$$y_{min} = -6.541 \ln(100) + 48.723 = 18.601 \tag{39}$$

$$y_{range} = y_{max} - y_{min} = 60.244 \tag{40}$$

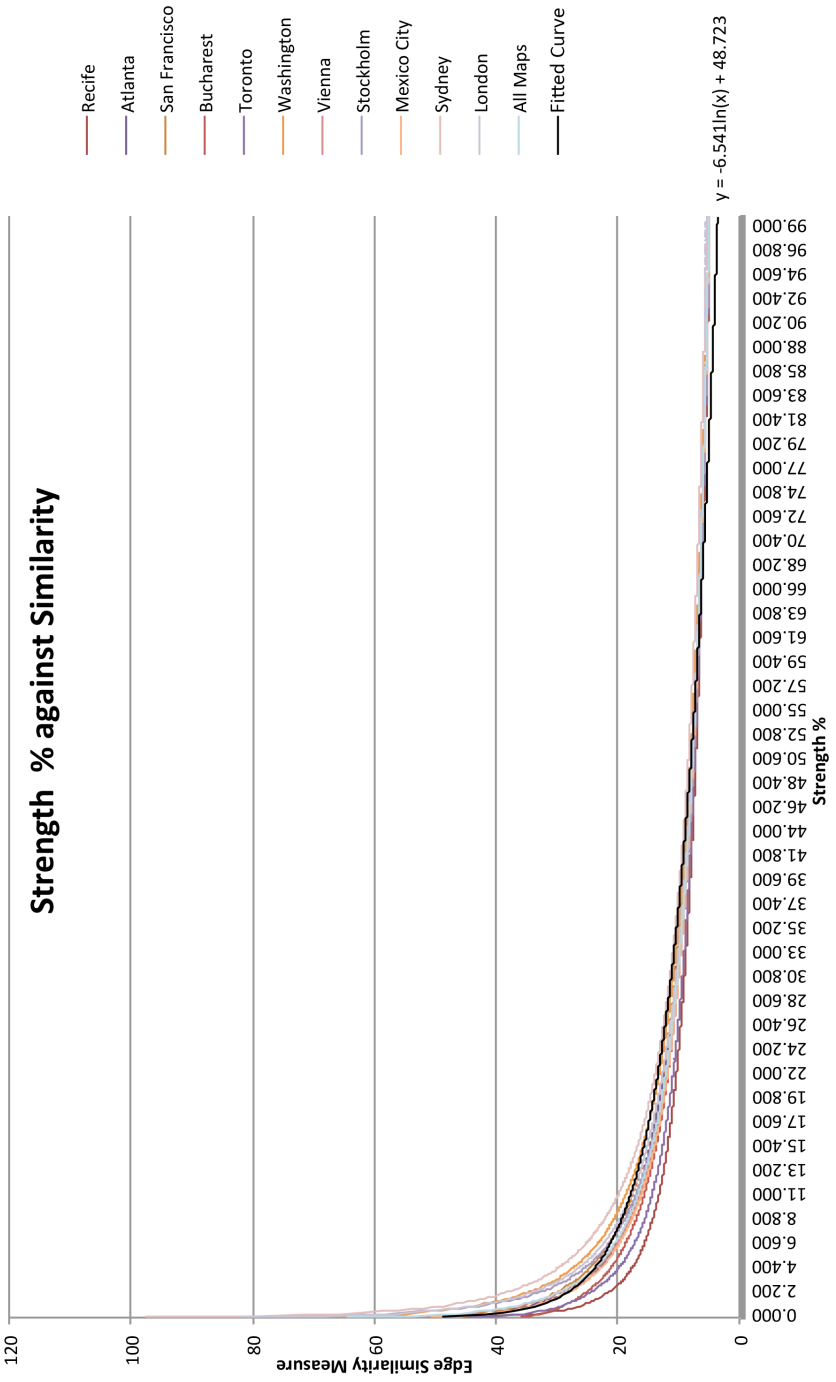


Figure 113: Graph similarity against percentage frame strength.

Example: if a k spring strength value of 0.125 is used for 100% MMP (we term this k_{100}); and we desire 25% mental map preservation (MMP), then this is $100 - 25 = 75\%$ of y_{range}. Therefore we need to find the strength percentage which produces a result of 75% of y_{range}, $+y_{min}$. Equation 41 shows the calculation to find the desired y-value.

$$0.75y_{range} + y_{min} = 0.75 \times 60.244 + 18.601 = 63.784 \tag{41}$$

From this y-value we can use Equation 37 to find x (Equation 42).

$$x = e^{\frac{63.784-48.723}{-6.541}} = 0.1 \tag{42}$$

This x-value represents the required strength percentage to produce the correct value (75%). From this we can set the correct value for k based on the schematic's k_{100} (Equation 43), x% of k_{100}.

$$k = \frac{x}{100} \times k_{100} = 1.25 \times 10^{-4} \tag{43}$$

Exceptions to this rule are 0% MMP where $k = 0$, and 100% MMP where $k = k_{100}$.

7.8 Summary

In this chapter we have shown how we use a Delaunay triangulation to construct a frame over the schematic in order to constrain node proximities during layout. We have explained how we have integrated these new forces into our existing algorithm, and have detailed how we created a user controllable slider that maps linearly to the similarity value of the resultant schematic.

Following on from this work, we performed a study to examine the effect of mental map preservation in dynamic schematic layout on user comprehension. The details and results of this study are covered the next section.

Chapter 8

Mental Map Preservation Comprehension Study

This chapter describes the procedure for a study to answer our research question: "Can mental map preservation improve schematic comprehension in dynamic schematic layout?". The hypothesis was that varying the level of mental map preservation would have an effect on user response time and accuracy. That is, a schematic re-optimised with high mental map preservation is easier to comprehend (with regards to accuracy and speed of use) than a schematic re-optimised with low mental map preservation – or vice versa.

Intuitively, a high level mental map preservation during re-optimisation of a schematic will aid the user's comprehension, as discussed previously in Section 2.2.4 and shown in Figure 26; however, previous research into the effect of mental map preservation has not shown any significant effect (Section 2.2.4), and so we hoped to augment these findings.

The style of study performed here is based up those performed in (Purchase and Samra 2008) and (Saffrey and Purchase 2008), in which participants are shown a sequence of graphs and asked node/path-finding tasks to evaluate response time and accuracy. The software used for this testing, along with schematic and question data, test documents and anonymised raw data can be downloaded from: `http://www.cs.kent.ac.uk/projects/fdol/study/`

Type	Schematic	Final Q.	Nodes	Edges	Lines	Node+	Node-	Edge+	Edge-	Line+	Line-
Example	em0	-	26	37	4	5	0	7	0	1	0
Test	tm0	-	19	20	2	5	4	9	5	0	0
Test	tm1	-	19	20	3	4	0	7	0	1	0
Test	tm2	-	25	33	4	4	5	12	11	1	1
Test	tm3	-	28	41	5	1	5	1	10	0	1
Test	tm4	-	28	28	6	1	2	2	6	0	1
S-A	m3	X	28	30	4	7	0	13	0	1	0
S-R	m5	Y	28	29	4	0	7	1	15	0	1
S-A-R	m4	Z	28	29	4	7	7	12	14	1	1
M-A	m6	Z	35	38	5	7	0	13	0	1	0
M-R	m1	X	35	35	5	0	7	1	14	0	1
M-A-R	m7	Y	35	36	5	7	7	14	10	1	1
L-A	m8	Y	42	43	6	7	0	12	0	1	0
L-R	m0	Z	42	43	6	0	7	0	11	0	1
L-A-R	m2	X	42	43	6	7	7	14	11	1	1

Table 11: Schematic metrics.

8.1 Schematics

We used three sizes of schematic as follows: Small (S) 4 lines, 28 nodes; Medium (M) 5 lines, 35 nodes; and Large (L) 6 lines, 42 nodes. We used three modification types as follows: Line addition (A) +1 line, +7 nodes; Line removal (R) -1 line, -7 nodes; Line addition and removal (A-R) +1 line, +7 nodes, -1 line, -7 nodes. These variations create nine unique tasks: {S-A, S-R, S-A-R, M-A, M-R, M-A-R, L-A, L-R, L-A-R}. Schematic data, including one example and five test schematics, is shown in Table 11. Each task was re-optimised after modification with three levels of mental map preservation (MMP) – these are 0%, 50% and 100% (explained in the previous chapter and illustrated in Figure 111).

The previous studies on which we based our design used a within-subject methodology to reduce subject variability; however, we felt that in order to eliminate additional variables we would switch to using a between-subject methodology, meaning that each participant was only part of a single test group (one MMP variant). The reason for this is that it we felt it was more important to use the same graphs and questions for each group in order to minimise the number of variables and only change the level of MMP. Participants could therefore not be used to test multiple levels of MMP, as they would already be familiar with the maps and questions presented. To combat the increased risk of subject variability, we used a far greater number of participants – (Purchase

and Samra 2008) and (Saffrey and Purchase 2008) used 30 and 21 subjects respectively whereas we used 60.

In order to help alleviate the effect of a learning curve when answering questions, we first had participants complete five training schematics. All data recorded on these training schematics was discarded before evaluation, and they began with a lower complexity (tm0 = 2 lines) building up to the level of actual test schematics (tm4 = 6 lines). Schematic data was generated using the Flickr API which we used to retrieve a number of photos, represented as nodes, and their associated tags, represented by coloured lines between nodes. Nodes that shared the same tag appeared along the same line, much like LineSets (Alper et al. 2011). Figure 114 shows an example of a schematic used in the study along with it's modification and re-optimisations.

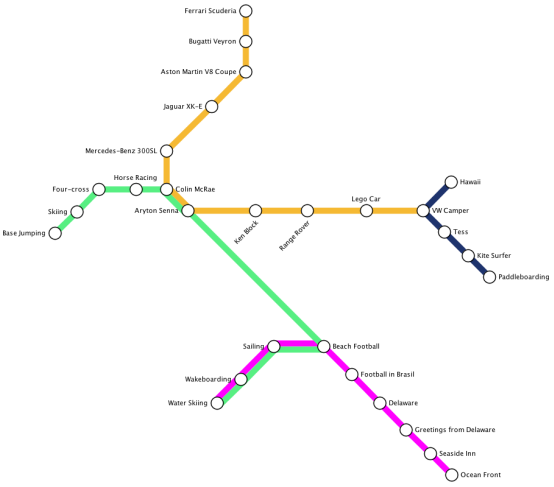
8.2 Questions

Each task consists of two stages of questions. Questions are multiple-choice with a selection of four possible answers. Rather than use schematic terms (node and line), in questions we use terms appropriate to the data set, these being photo and tag respectively.

Firstly, the unmodified schematic is shown to the participant and five questions involving basic photo/tag finding tasks specific to the schematic are asked. These questions are to familiarise the participant with the schematic before any modification is made. Example questions for stage 1 are as follows (these questions are specific to the example schematic in Figure 114):

1. How many tags are associated with the photo titled "Range Rover"?
2. Which of the highlighted photos contains the most tags?*
3. How many tags are associated with the photo titled "Colin McRae"?
4. How many photos contain the yellow tag?
5. How many photos contain all of the following tags: pink and dark blue?

(a) Original Schematic (m1, M-R).



(b) Modified Schematic – 100% MMP.

(c) 50% MMP.

(d) 0% MMP.

Figure 114: Comparison of MMP variants used in the study. 114b shows the modified schematic, in which the magenta line has been removed (-4 nodes), along with removal of three more nodes: “Bodyboarding” (upper-right, navy), “Maracana” (centre, green/magenta), and “Goal!” (centre, green/magenta). The yellow line has also been removed from nodes “Ken Block” and “Range Rover” (centre, green/yellow).

*This question (2) references highlighted tags; although these have been omitted from the example schematic shown, in practice we highlight the label of 4 potential nodes with a yellow background during this question. An example of this can be seen in Figure 115.

Following these questions, the schematic undergoes a modification and re-optimisation. The schematic is re-optimised in line with the users designated MMP group. A single stage 2 question is then asked on this re-optimised schematic. There are three types of stage 2 question, X, Y and Z, which are assigned to schematics so that each type is asked once for each schematic size and each modification type – question assignment can be seen in Table 11. Stage 2 questions are as follows:

X. How many photos contain the "..." or "..." tags and not the "..." tag?

Y. What is the minimum number of tag changes required to travel between the highlighted nodes?

Z. Which tag contains the most photos?

Only stage 2 question answers are used in the data analysis; stage 1 questions serve only the purpose of familiarising the participant with the schematic before the modification and re-optimisation is performed.

8.3 Software

In order to carry out our tests, we developed a custom piece of software with which participants can interact to answer questions. Our software presents participants with the schematic, question, and possible answers for each task in sequence. The modification and re-optimisation process of the schematic is performed by the software as a fade in/out of lines/nodes followed by an animated linear interpolation between node positions taking four seconds.

Figure 115 shows our software during a test; the schematic is shown to participants in a large panel to the left, with a smaller right panel containing a progress bar, the current question and 4 option button answers. This image also shows an example of how we highlight nodes during questioning.

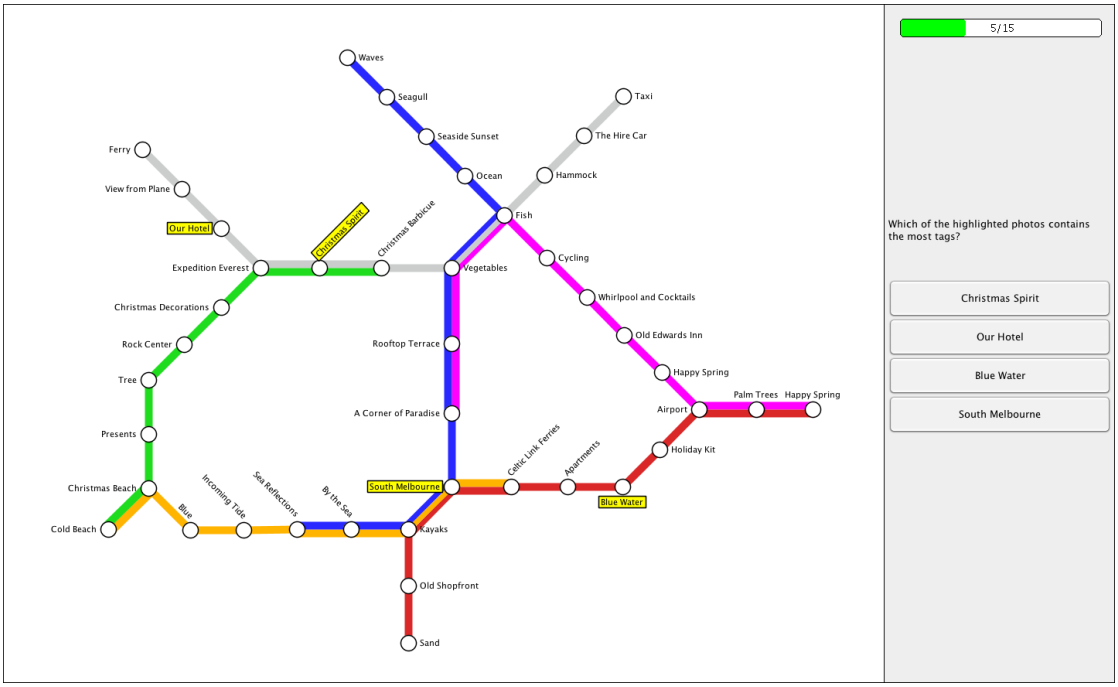


Figure 115: Our testing application used to present schematics and questions.

As the participant is asked questions throughout the test, the time taken for each answer is recorded along with their chosen answer. The software also collects participant details at the start of the test including their Kent login, age, gender, and subject. Once the participant has completed all tasks, the software outputs a log file to the system; an example output log section can be seen in Table 12 which shows anonymised data for 4 tasks of a single participants test. Tasks can be identified from this table by the map column, which shows the values tm3, tm4, m0, and m1; these represent schematics Test 3, Test 4, Map 0, and Map 1 respectively (details of each schematic can be seen in Table 11). Further information we can gather from this table, besides participant details, includes the level of MMP this user was assigned (indicated by the variant of the final question for each schematic – in this case reop_50 indicates this participant was part of the 50% MMP group), their answer against the correct answer (the correct column indicates if these match – redundant but included for faster readability) and the time taken to answer in milliseconds.

userID	age	gender	course	map	variant	questionID	answer	correctAnswer	correct	time(ms)
...										
1.31107E+11	20	f	German and Spanish	tm3	base	0_base	2	2	1	5647
1.31107E+11	20	f	German and Spanish	tm3	base	1_base	1	1	1	7712
1.31107E+11	20	f	German and Spanish	tm3	base	2_base	3	3	1	25427
1.31107E+11	20	f	German and Spanish	tm3	base	3_base	0	0	1	2889
1.31107E+11	20	f	German and Spanish	tm3	base	4_base	1	1	1	5585
1.31107E+11	20	f	German and Spanish	tm3	reop_50	0_reop	1	1	1	12906
1.31107E+11	20	f	German and Spanish	tm4	base	0_base	2	2	1	12202
1.31107E+11	20	f	German and Spanish	tm4	base	1_base	1	1	1	22729
1.31107E+11	20	f	German and Spanish	tm4	base	2_base	1	1	1	6969
1.31107E+11	20	f	German and Spanish	tm4	base	3_base	2	2	1	11758
1.31107E+11	20	f	German and Spanish	tm4	base	4_base	0	0	1	6445
1.31107E+11	20	f	German and Spanish	tm4	reop_50	0_reop	1	1	1	24130
1.31107E+11	20	f	German and Spanish	m0	base	0_base	3	3	1	19582
1.31107E+11	20	f	German and Spanish	m0	base	1_base	1	1	1	8946
1.31107E+11	20	f	German and Spanish	m0	base	2_base	2	2	1	2923
1.31107E+11	20	f	German and Spanish	m0	base	3_base	0	0	1	8749
1.31107E+11	20	f	German and Spanish	m0	base	4_base	2	2	1	5181
1.31107E+11	20	f	German and Spanish	m0	reop_50	0_reop	3	3	1	16455
1.31107E+11	20	f	German and Spanish	m1	base	0_base	2	2	1	8232
1.31107E+11	20	f	German and Spanish	m1	base	1_base	1	1	1	8000
1.31107E+11	20	f	German and Spanish	m1	base	2_base	1	1	1	4910
1.31107E+11	20	f	German and Spanish	m1	base	3_base	3	3	1	5340
1.31107E+11	20	f	German and Spanish	m1	base	4_base	0	0	1	13829
1.31107E+11	20	f	German and Spanish	m1	reop_50	0_reop	2	0	0	26526
...										

Table 12: Example anonymised software output section.

8.4 Testing Procedure

In order to test the effectiveness of varying levels of MMP we chose to test three levels, these are 0%, 50%, and 100%. To do this we used a between-subject methodology with a third of our participants in each group. Our testing software assigned participants to groups, cycling through in order; our first participant was assigned 0%, second 50%, and so on, cycling around from 100% to 0%. Using such a method ensured our group assignment was unbiased.

Participants were first given a test instructions document which was also read to them; this document can be found online at the URL listed previously. As part of this document, an example task is shown and it's questions are walked through with the participant. This example task was used to familiarise the participant with the type of questions that would be asked, along with the interface of the testing software. Once this example task had been completed and the remainder of the instructional document read, the participant was given a chance to ask any questions they may have. We did not reveal any information relating to our hypothesis to participants until the test was fully

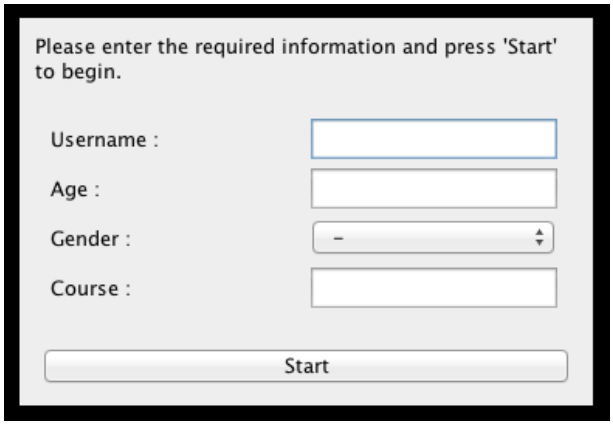


Figure 116: The data entry form for participants to input their details.

completed. Following this, the software displayed a small data entry window (Figure 116) asking the user to enter the following information: Kent login, age, gender and subject. Once the user had entered their details and pressed "Start", the first training schematic was shown. Once the five training tasks had been completed, there were nine more tasks to complete (as seen in Table 11); the results of these last nine tasks are those used in our data analysis. Each participant saw the same order of schematics with the same questions; the only variation being the re-optimisation of each schematic which was dependent on the participants mental map preservation group (the same final question was still asked, as the re-optimisation only alters the position of nodes).

After these nine tasks, there was one final stage of questions in the form of a paper questionnaire combined with a number of schematics shown on the software. The user was read a short script and provided with the questionnaire. The questionnaire involved viewing three schematics, each of which could be re-optimised as many times as desired with each of the three levels of MMP (0%, 50%, and 100%) by pressing buttons to the right (the screen that shows during this task is shown in Figure 117); and then subjectively ranking these re-optimisations from best to worst. The questionnaire also asked a number of qualitative questions about the schematics and tasks. Finally, the participant was paid £7 for their time, and provided with a debrief sheet explaining the purpose of our study.

As mentioned previously, each task consisted of six questions in the form of

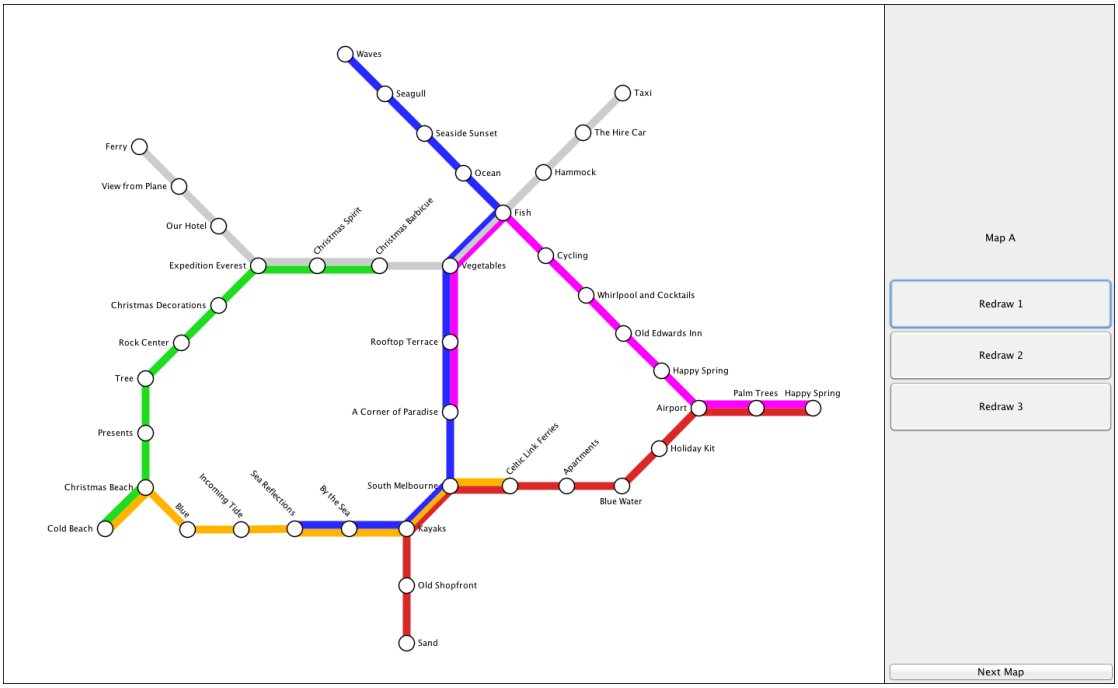


Figure 117: Software interface for the final mental map preservation ranking task.

five familiarisation questions followed by a modification and re-optimisation, and then another question. At the start of each task, the software gave participants five seconds to examine the newly displayed schematic before the question and answer buttons were made visible. This provided a further small opportunity for participants to briefly familiarise themselves with available line colours and basic structure before any questions were presented which may distract them. Measuring the time taken for participants to answer each question was crucial for our testing process. However, the participant could rest between each task when no question was being presented.

We first performed a pilot study for which we used nine subjects (three per condition). This study highlighted a number of small mistakes such as spelling and question ambiguity, but no major flaws were identified. All issues found were corrected before starting the main study. During the main study we used a total of 60 participants recruited from the University of Kent job centre. These participants were from a wide range of disciplines and ranged from 16 to 34

Table 13: Mean response time and mean number of errors for the three MMP conditions, over all non-test schematics and all post-modification questions.

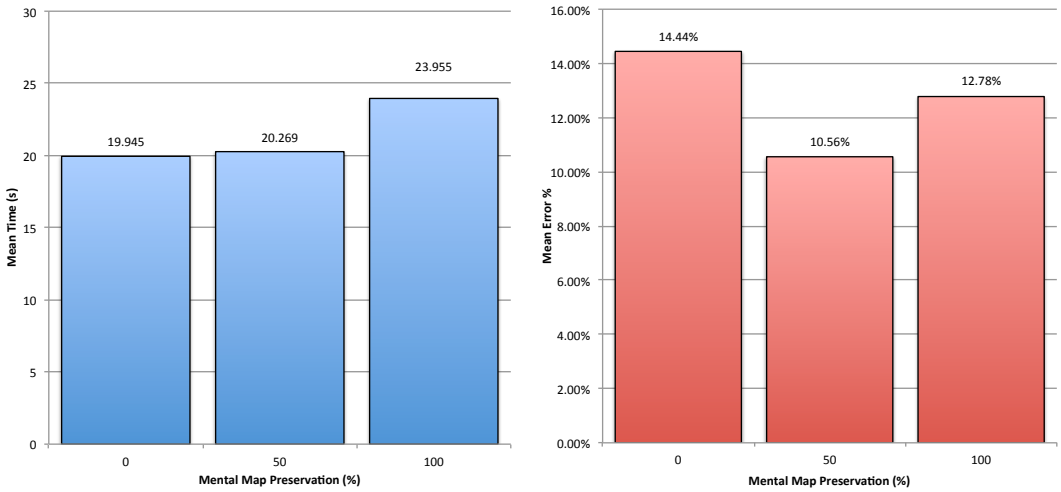
MMP	Time (s) $\bar{x}$	σ	Error (%) $\bar{x}$	σ
0%	19.95	4.31	14.44	13.54
50%	20.27	4.81	10.56	12.73
100%	23.96	7.07	12.78	13.13

years of age ($\bar{x}$ = 20.83, σ = 2.88), 20 male and 40 female.

8.5 Results

Statistical analysis techniques were performed following guidelines in (Purchase 2012). For a subjective interpretation of these results, see the following section. In our analysis, data from questions which were answered incorrectly were also included. This is because response time is intended to be an indication of cognitive effort required, independent of whether or not the effort resulted in a correct answer. Figure 118 shows charts for the mean time and mean error against condition; Table 13 presents the data in tabular form with standard deviations. We used the Shapiro-Wilk method to determine if our data followed a normal distribution, to which neither response time nor error rate did for all conditions. We therefore then performed a Kruskal-Wallis non-parametric analysis of variance test. A non-parametric Levene's test was first used to verify the equality of variances in the samples (homogeneity of variance assumption). In the following tests, we use a p-value of < 0.05 to indicate significance.

Kruskal-Wallis: There were no significant differences in response time ($p = 0.207$) or error rate ($p = 0.593$) as represented by the error data according to condition under a non-parametric independent measures Kruskal-Wallis test. These results indicate that the difference between the conditions can likely be attributed to random chance, rather than being due to the differing nature of the conditions.



(a) Chart of mean time (s) against condition.

(b) Chart of mean error (%) against condition.

Figure 118: Test data shown as charts. Lower is better for both time and error.

8.6 Interpretation of Results

No significance was found between the level of MMP and user response time or accuracy; this result provides evidence against our hypothesis. A possible explanation for this is due to multiple impacts on diagram comprehension. One impact occurs when the mental map is preserved – the layout is compromised by the unoptimised modifications, making analysis of the diagram difficult because of features such as increased line bends and less effective node positioning. The alternative extreme is that the mental map is not preserved, so the diagram changes a great deal, impacting comprehension because the participant needs to re-examine parts of the diagram that have changed. One hypothesis is that these two conflicting impacts on comprehension are broadly equal, and so it is not important which approach is taken for dynamic data. There have been a number of studies on the effect of MMP on user readability as covered in Section 2.2.4; however, none have found conclusive evidence of any effect, supporting this hypothesis.

An important assumption made was that during the first five questions the user builds a sufficient mental map of the schematic. Previous work in (Tory,

Swindells and Dreezer 2009) as referenced by (Archambault and Purchase 2012) tested the memorability of node diagrams. Their study found that users have an accuracy of 87.1% in recognising a specific diagram in a set of 8 after an initial exposure of 12 seconds. The mean time our participants viewed each schematic for before modification was 56.13 seconds; this supports our assumption that the first five questions were sufficient for participants to build a mental map.

During the questionnaire of our study we asked a number of optional qualitative questions on the schematics and tasks to ensure there were no consistent problems in the study. However, the results to these did not indicate any problems and were not useful in the evaluation of our results.

8.7 Summary

This chapter has covered the implementation and execution of a study performed to evaluate the research question of "Can varying mental map preservation improve diagram comprehension in dynamic schematic layout?". This study was performed with the intention of augmenting the current inconclusive research into the effect of mental map preservation, covered in Section 2.2.4. From our results we have found no significant difference in either response time or error rate based on the level of MMP, in line with previous studies. We have suggested that this insignificance may be due to both extremes imposing conflicting factors on comprehensibility.

Chapter 9

Conclusion & Future Work

This chapter provides a summary of each chapter of work, highlighting our contributions. We also introduce a number of potential future work ideas that could be explored.

9.1 Contributions

We first developed a piece of software, written for touch screen Android devices, facilitating the drawing of metro map style schematics. Our software uses a gesture-based input system, allowing users to quickly create schematics using a series of simple gestures. Despite a number of limitations from using early tablet devices such as performance issues with complex or large schematics, our gesture-based input mechanism proved effective for schematic diagram creation (Chapter 3).

Using our gesture-based input software, we also implemented a multi-criteria hill climbing optimiser for automatic layout of drawn schematics. This method was based upon previous work by Stott et al. (Stott et al. 2010), however we made a number of modifications to increase the quality of output and optimisation speed. This included the implementation of a number of new criteria, clustering methods, post-processing steps, and many performance optimisations (Chapter 4).

The performance optimisations made to our layout method allowed us to

implement functionality to batch-optimise schematics with no user input in a reasonable time frame. We knew that hill climbing techniques were prone to local optima, and so we used this batch-processing functionality to experiment how varying the initial parameter settings affected the final layout. We identified patterns that showed each schematic performed best under a specific set of parameter values, rather than all schematics performing best when the search space was maximised. This indicated a possible relationship between characteristics of the input schematic and optimal parameter values (Chapter 5).

We then examined force-directed layout techniques due to a number of advantages they provide over search-based layout techniques including performance, simplicity, predictability, and interactivity (Chapter 6). As explained in the background chapter, the existing method for force-directed schematic layout also leaves much room for improvement. We developed a new force-directed system capable of effective schematic layout with edges constrained to a defined angular resolution. Our method is cable of producing results comparable to current alternatives, whilst also providing the benefits of force-directed layout.

We implemented our force-directed layout technique into a new piece of software. This software allows the creation of node link diagrams and provides additional support for aesthetic features specific to metro map style schematics such as coloured and parallel edges. The software is written in Java, has been tested on Windows and Macintosh, and is freely available for download at: `http://www.cs.kent.ac.uk/projects/fdol/`.

We then extended our work on force-directed layout by implementing a novel mental map preservation technique. Our technique uses a sprung Delaunay triangulation frame to constrain nodes by their proximity to other nearby nodes. This is in contrast to the standard approach of constraining nodes by absolute distance from their original positions. Our technique also allows us to accurately control the level of mental map preservation applied by adjusting the strength of our Delaunay frame (Chapter 7).

We used our implementation of mental map preservation to test the hypothesis that enforcing a higher level of mental map preservation increases the comprehensibility of a schematic after a re-optimisation in response to a modification. Our study showed no significant results in terms of error or accuracy, in line with previous studies (Chapter 8).

9.2 Future Work

This section covers a number of potential ideas for extending the work covered in this thesis.

9.2.1 User Study on Layouts between Multiple Techniques

Over the past decade there have been a multitude of novel techniques for automated metro map layout, and most of the literature published with these methods contains a comparison between the new and existing layout techniques. However, these evaluations are written by the authors and mostly provide a few comparison examples highlighting the areas in which they vary, but leaving much of the decision up to the reader.

A major problem with the evaluation of new schematic layout techniques is that there is no objective measure by which a layout can be rated. Although most search-based methods do use a number of criteria to rate the current configuration with an objective function, these values are not comparable between layout techniques due to differences in many aspects of implementation. However, even if these values were comparable, the measured fitness of the schematic is based upon the assumption that the implemented criteria and their weightings do indeed represent how users would rate a layout. Therefore, the objective function is useful for the algorithm to maximise a number of criteria, but cannot be used to compare schematics.

Because schematic layout evaluation is subjective, it is necessary to perform a user study to determine favourable and effective maps; but due to factors such as time, there have been very few user studies on the comprehension of layouts produced by automated methods, in particular a comparison between

them. It would be therefore be beneficial to perform a user study evaluation of a number of current layout techniques, as well as officially published schematics, to evaluate user preference, accuracy, and performance across a number of layouts. Different authors have chosen to implement various criteria into their layout methods, and so such a study may also highlight effective combinations of criteria.

9.2.2 Schematic Aesthetics

Metro map appearance is subjective, and for many users the preference of one map over another is entirely dependent on aesthetic appearance, rather than usability. Of course, layout does play a part in this decision as it is possible to determine the usability of a schematic by visual inspection to a certain extent, but for two schematics of a visually similar difficulty the decision will be based on aesthetic preference.

Aesthetic preference refers to aspects such as the colours, styles, fonts, curves, and sizes used; rather than layout decisions such as maximising the number of straight lines. Figure 119 illustrates how a number of small aesthetic changes can produce a drastically different schematic.

Certain features, including resized junctions and line curves, unarguably have a positive effect on schematic aesthetics, and are desirable additions to any layout system. Other features such as fonts, colours, and station style also change the appearance of a schematic but the effects of these are subjective. Figure 119b is clearly very different from the original Figure 119a, and this may or may not appeal to individuals.

The aesthetics of metro maps has received little attention in the area of automated layout, with authors understandably focusing purely on the resulting layout. However, on such a subjective visual diagram perhaps the importance of this should not be underestimated. As well as determining what people find attractive, it would be interesting to test if any aesthetic properties can significantly affect the comprehension of a schematic in either a positive or negative way.

Figure 119: An example of how small aesthetic changes can drastically change the appearance (either for good or bad). Both examples show a section of the Stockholm metro map as produced by our multi-criteria hill climbing optimiser. Figure (b) has been modified by hand to include: curved line bends, enlarged junctions where applicable, different station markers, a different font/colour, and a different background colour.

9.2.3 Extension of Graphs – FDOL Software

The software we developed for the implementation and testing of our force-directed layout method, along with our Delaunay triangulation mental map preservation provides a solid framework for the creation and manipulation of node-link diagrams. During development it was important to structure the software to allow easy implementation of additional layout methods. Due to a number of factors such as time constraints, certain features were implemented in a less-cohesive fashion. However, with a small amount of effort the program could be refactored to become a foundation for the fast implementation of any node-link layout method.

Our software already supports a number of features specific to metro map layout such as edge colouring, parallel edges (multiple lines of different colours running between the same stations), and a layout method developed specifically for metro maps. More focus could be placed on emphasising the software as a tool for the design and layout of metro map style schematics. Additional existing metro map layout techniques could be implemented, along with

more drawing improvements such as polygonal edges using rounded corners and junctions that adjust their size depending on the number of lines that pass through them. Implementation of alternate layout techniques would be difficult, as each would need to be modified to work with a different underlying graph structure (particularly if features such as polygonal edges are allowed, which are not considered in most current layout methods); but a single piece of software capable of demonstrating multiple layout techniques would be very desirable. There are still currently no public software tools, as far as we are aware, developed specifically for the layout of metro maps. However, the layout style is slowly becoming more popular, and a tool which supports the creation of such maps may help speed up this process.

Along with additional layout methods, our implemented mental map preservation technique could be improved. Currently, in order to achieve a desired percentage of mental map preservation, we scale the frame strength value based upon a predetermined formula derived from averaging a number of example schematics (as explained in Section 7.7 and shown in Figure 113). The result of this is that our percentage slider is not as accurate as it could be for all schematics. This leaves room for improvement as it would be possible to recalculate this formula for any individual map at the time of optimisation. The maximum frame strength value used could also be similarly calculated specifically for each schematic.

The current input mechanism uses a mouse, and involves double-clicking or right-click-dragging to create nodes and edges. Our initial software, SchemaSketch, has shown that a gesture based input system is effective for drawing schematics and this could be implemented into the Graphs software. Although this software has been developed for use on PCs, it is becoming increasingly common for even these to also use touch screens. Alternatively, users could draw gestures using either a graphics tablet, touchpad, or mouse.

The Graphs – FDOL software code is freely available at: `https://github.com/d-chivers/Graphs`.

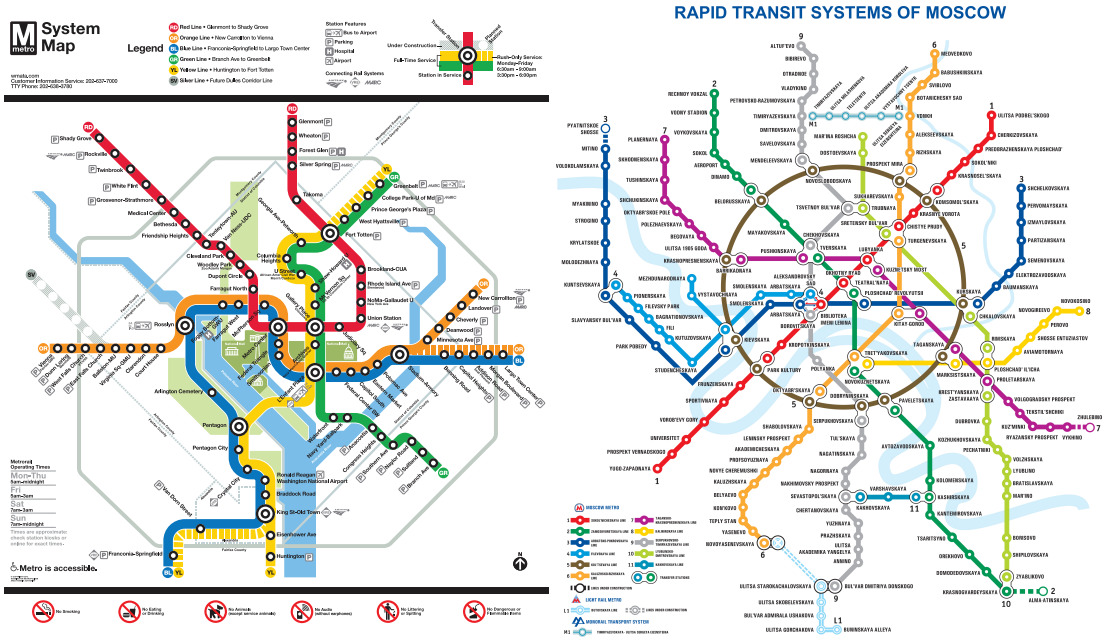
9.2.4 Extension of Force-directed Layout Method

Our octilinear force-directed layout method uses a two stage process to transition between two types of force in order to apply a second criterion. In alternate search based methods there are typically many more criteria applied during the layout that we do not consider. It would therefore be interesting to explore how these extra criteria could be implemented into our algorithm with the use of additional forces. Care must be taken when implementing additional forces, as conflicting criteria can negatively impact the final layout. It is perhaps more feasible to implement additional criteria as node movement constraint rules, rather than forces acting on nodes.

Force-directed layouts find local optima, and our implementation does nothing to try to avoid this. Along with local optima issues, force-directed methods are typically only effective upon graphs with up to around 100 nodes – a medium sized metro map. Both of these issues could potentially be solved with the use of a multi-level approach, as described in (Walshaw 2001). The multilevel approach does not guarantee a global optimum, but has been shown to improve the local optima found. Along with this, it is capable of optimising graphs with many thousands of nodes. Our method could be extended to use a multilevel approach to improve the produced layouts, as well as provide support for much larger schematics.

Currently, labelling of schematics is performed post layout and has no effect at all on the position of nodes in the schematic. This approach suffers when there is not enough room left to position labels without introducing occlusion. It is therefore desirable to account for labels during the layout process. This could be implemented with additional forces around labels to prevent occlusion with other objects, and either more forces or a movement constraint to hold labels near their parent node.

In the last few years many published metro maps have taken a more geographic stance, often including overground features such as rivers, parks, and other landmarks. It would be interesting to experiment with how these features could be incorporated into an automated layout system, as this has not yet been attempted. The landmarks would need to impose topographical rules upon



(a) Washington – Radial

(b) Moscow – Orbital

Figure 120: Radial and orbital map characteristics can be seen in the Washington (120a) and Moscow (120b) metro maps respectively.

nearby nodes in order to maintain some level of geographical accuracy in their vicinity. The shape of features such as parks and rivers is also flexible, meaning they can be slightly modified to allow an improved layout surrounding them. Parks are often simply an enclosed polygon of nodes, and perhaps rivers could be created using a Bézier curve with control points subject to the forces from nearby nodes.

9.2.5 Metro Map Characterisation

From our experimentation in parameter manipulation we concluded that different maps favour different optimisation values. However, we were unable to identify any relationship between a schematic and its optimal parameter settings. It would be useful to develop a number of characteristics which can be used to describe particular metro maps – a few examples currently exist including radial and orbital maps. Radial describes a metro map consisting of a dense central area with many lines extending outwards from the centre, for example

Washington (Figure 120a). Orbital maps are characterised by a predominant circular route around the city, such as is the case for Moscow (Figure 120b).

Characterisations of metro maps are limited, and those that exist are not formally defined. This means that there is no metric to algorithmically determine if a map is radial or orbital. If a number of additional characteristics could be developed and all of these formally defined, it may be possible to identify a relationship between measurable map characteristics and optimal optimiser parameters. If such a relationship exists, it would allow auto-tuning of layout algorithms – improving both performance and the final layout.

9.2.6 Combination of Layout Methods

There are now many different layout techniques which operate in isolation from each other. However, there has been no research into how a combination of multiple techniques may be able to benefit automated layout.

An example of prime candidates for combination could be a force-directed technique (faster, lower quality) and a search-based technique (slower, higher quality). The two techniques could either be run sequentially, for example using a force-directed method as a pre-processing step for a search based method; or semi-simultaneously, for example the force-based method could be applied between iterations of the search-based method.

It is unknown what effect such a combination would have, but it is entirely possible that multiple methods may work effectively together and aid in escaping local optima configurations.

9.2.7 Definition of Standard Data Sets and Evaluation Metrics

Objective evaluation of produced layouts is currently very difficult and one of the biggest issues in this field. It would be hugely beneficial to establish defined data sets and a number of evaluation metrics for this purpose. The Sydney metro map has become an unofficial stand-in for this task, however there is no single agreed-upon data set for this map, resulting in variations between

authors. There are also no objective evaluation metrics for it, and so any comparison not based upon empirical study is purely subjective.

Many layout methods use an objective function, but most of these values cannot be compared due to implementation differences. Any evaluation metrics devised to objectively evaluate a layout would have to be independent from implementation details and unambiguously defined. An example of such metrics could be the sum of all line bends in degrees for all lines within the schematic, or the absolute number of line crossings.

The main problem with establishing data sets and evaluation metrics is that it is not agreed upon which criteria are required for the most effective layout, so authors often do not even use the same criteria. Different schematic maps also appear to work well with different criteria and/or criteria weights. A process would have to be defined for determining the best set of criteria for a specific schematic, and the best possible layout that can be found via empirical testing that minimises the metrics. Further, if the defined criteria do not accurately represent an effective schematic in terms of user comprehension, then the tests would only serve to illustrate the technical effectiveness of a method in fulfilling criteria, rather than achieving an improved layout.

Definition of objective evaluation metrics would be very beneficial, however it is an extremely difficult task as the "best" criteria are not agreed upon and most likely vary between schematics – possibly even between individuals. Definition of standard data sets is possible, and this would go some way in helping the task of layout comparisons. The difficulty in this task lies within achieving both knowledge and adoption of the data sets into the community.

Bibliography

Alper, B. et al. (2011). Design study of linesets, a novel set visualization technique. *IEEE Transactions on Visualization and Computer Graphics*, 17(12), pp. 2259–2267.

Anand, S. et al. (2007). Automated schematic map production using simulated annealing and gradient descent approaches. In *Proceedings of the* 15^{th} *annual GIS Research UK Conference*, vol. 7.

Arbesman, S. (accessed 15/11/2014). Milky way transit authority. `http://www.arbesman.net/milkyway/`.

ArcGIS (accessed 01/06/2015). `http://www.esri.com/software/arcgis/extensions/schematics`.

Archambault, D. and Purchase, H. C. (2012). The mental map and memorability in dynamic graphs. In *Proceedings of the Pacific Visualization Symposium (PacificVis12)*, pp. 89 – 96.

Archambault, D., Purchase, H. C. and Pinaud, B. (2011). Animation, small multiples, and the effect of mental map preservation in dynamic graphs. *IEEE Transactions on Visualization and Computer Graphics*, 17(4), pp. 539–552.

AutoCAD (accessed 01/06/2015). `http://www.autodesk.co.uk/products/autocad`.

Avelar, S. and Müller, M. (2000). Generating topologically correct schematic maps. In *Proceedings of the* 9^{th} *International Symposium on Spatial Data Handling*, pp. 4–28.

Barnes, J. and Hut, P. (1986). A hierarchical $o(nlogn)$ force-calculation algorithm. *Nature*, 324, pp. 446–449.

Bartel, G. et al. (2011). An experimental evaluation of multilevel layout methods. In *Proceedings of the* 18^{th} *International Symposium on Graph Drawing (GD 2010)*, *Lecture Notes in Computer Science*, vol. 6502, Springer-Verlag, pp. 80–91.

Beck, H. (accessed 15/11/2014). 1931 underground map. `http://www.ltmuseumshop.co.uk/posters/londons-transport-system/product/1931-underground-map-by-harry-beck-poster.html`.

Bowyer, A. (1981). Computing dirichlet tessellations. *The Computer Journal*, 24(2), pp. 162–166.

Brandes, U. and Wagner, D. (1998). Using graph layout to visualize train interconnection data. In *Proceedings of the* 6^{th} *International Symposium on Graph Drawing (GD 1998)*, *Lecture Notes in Computer Science*, vol. 1547, Springer-Verlag, pp. 44–56.

Chivers, D. and Rodgers, P. (2011). Gesture-based input for drawing schematics on a mobile device. In *Proceedings of the* 15^{th} *International Conference on Information Visualization (IV 2011)*, IEEE Computer Society, pp. 127–134.

Chivers, D. and Rodgers, P. (2013). Exploring local optima in schematic layout. In *Visual Languages and Computing (VLC) in the* 19^{th} *International Conference on Distributed Multimedia Systems (DMS 2013)*, Knowledge Systems Institute, pp. 168–175.

Chivers, D. and Rodgers, P. (2014a). Improving search-based schematic layout by parameter manipulation. *International Journal of Software Engineering and Knowledge Engineering*, pp. – in print.

Chivers, D. and Rodgers, P. (2014b). Octilinear force-directed layout with mental map preservation for schematic diagrams. In *Proceedings of the* 8^{th} *International Conference on the Theory and Application of Diagrams (Diagrams 2014)*, *Lecture Notes in Computer Science*, vol. 8657, Springer-Verlag, pp. 1–8.

Davidson, R. and Harel, D. (1996). Drawing graphs nicely using simulated annealing. *ACM Transactions on Graphics*, 15(4), pp. 301–331.

de Berg, M. et al. (2008). *Computational Geometry: Algorithms and Applications – Chapter 9*. Springer-Verlag, 3rd edn.

Delaunay, B. (1934). Sur la sphère vide. a la mèmoire de georges voronoï. *Bulletin de l'Académie des Sciences de l'URSS, Classe des sciences mathématiques et naturelles*, (6), pp. 793–800.

Diehl, S. and Görg, C. (2002). Graphs, they are changing – dynamic graph drawing for a sequence of graphs. In *Proceedings of the* 10^{th} *International Symposium on Graph Drawing (GD 2002)*, *Lecture Notes in Computer Science*, vol. 2528, Springer-Verlag, pp. 23–31.

Dobkin, D. P. et al. (1997). Implementing a general-purpose edge router. In *Proceedings of the* 5^{th} *International Symposium on Graph Drawing (GD 1997)*, *Lecture Notes in Computer Science*, vol. 1353, Springer-Verlag, pp. 262–271.

Douglas, D. H. and Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2), pp. 112–122.

Dwyer, T., Hurst, N. and Merrick, D. (2008). A fast and simple heuristic for metro map path simplification. In *Proceedings of the* 4^{th} *International Symposium on Advances in Visual Computing (ISVC 2008)*, *Lecture Notes in Computer Science*, vol. 5359, Springer-Verlag, pp. 22–30.

Eades, P. (1984). A heuristic for graph drawing. *Congressus Numerantium*, 42, pp. 149–160.

Eades, P. and Feng, Q. (1997). Multilevel visualization of clustered graphs. In *Proceedings of the* 4^{th} *International Symposium on Graph Drawing (GD 1996)*, *Lecture Notes in Computer Science*, vol. 1190, Springer-Verlag, pp. 101–112.

Eades, P. and Garvan, P. (1996). Drawing stressed planar graphs in three dimensions. In *Proceedings of the* 3^{rd} *International Symposium on Graph Drawing (GD 1995)*, *Lecture Notes in Computer Science*, vol. 1027, Springer-Verlag, pp. 212–223.

Eades, P. et al. (1991). Preserving the mental map of a diagram. In *Proceedings of Compugraphics*, pp. 24–33.

EdrawSoft (accessed 15/11/2014). `http://www.edrawsoft.com`.

Fink, M. et al. (2013). Drawing metro maps using bézier curves. In *Proceedings of the* 20^{th} *International Symposium on Graph Drawing (GD 2012)*, *Lecture Notes in Computer Science*, vol. 7704, Springer-Verlag, pp. 463–474.

Finkel, B. and Tamassia, R. (2005). Curvilinear graph drawing using the force-directed method. In *Proceedings of the* 12^{th} *International Symposium on Graph Drawing (GD 2004)*, *Lecture Notes in Computer Science*, vol. 3383, Springer-Verlag, pp. 448–453.

Forrester, D. et al. (2005). Graphael: A system for generalized force-directed layouts. In *Proceedings of the* 12^{th} *International Symposium on Graph Drawing (GD 2004)*, *Lecture Notes in Computer Science*, vol. 3383, Springer-Verlag, pp. 454–464.

Freeman, I. J. and Plimmer, B. (2007). Connector semantics for sketched diagram recognition. In *Proceedings of the* 8^{th} *Australasian Conference on User Interface (AUIC 2007)*, vol. 64, pp. 71–78.

Frick, A., Ludwig, A. and Mehldau, H. (1995). A fast adaptive layout algorithm for undirected graphs. In *Proceedings of the DIMACS International Workshop (GD 1994)*, *Lecture Notes in Computer Science*, vol. 894, Springer-Verlag, pp. 388–403.

Fruchterman, T. and Reingold, E. (1991). Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11), pp. 1129–1164.

Gabriel, K. R. and Sokal, R. R. (1969). A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18(3), pp. 259–270.

Goodrich, M. T. and Wagner, C. G. (1998). A framework for drawing planar graphs with curves and polylines. In *Proceedings of the* 6^{th} *International Symposium on Graph Drawing (GD 1998)*, *Lecture Notes in Computer Science*, vol. 1547, Springer-Verlag, pp. 153–166.

Group, R. (accessed 15/11/2014). Digital workplace & marketing technology map. `http://www.realstorygroup.com/vendormap/`.

Gusaite, M. (2006). *Dynamic Scene Analysis and Beautification for Hand-drawn Sketches*. Master's thesis, Kaunas University of Technology.

Gutwenger, C. and Mutzel, P. (1998). Planar polyline drawings with good angular resolution. In *Proceedings of the* 6^{th} *International Symposium on Graph Drawing (GD 1998)*, *Lecture Notes in Computer Science*, vol. 1547, Springer-Verlag, pp. 167–182.

Hahn, W. C. and Weinberg, R. A. (accessed 23/04/2015). A subway map of cancer pathways. `http://www.nature.com/nrc/posters/subpathways/index.html`.

Hestenes, M. R. and Stiefel, E. (1952). Methos of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6), pp. 409–436.

Hobbs, M. and Rodgers, P. (1998). Representing space: A hybrid genetic algorithm for aesthetic graph layout. In *Frontiers in Evolutionary Algorithms (FEA 1998) in Proceedings of the* 4^{th} *Joint Conference on Information Sciences (JCIS 1998)*, vol. 2, pp. 415–418.

Hong, S.-H., Merrick, D. and do Nascimento, H. A. (2005). The metro map layout problem. In *Proceedings of the* 12^{th} *International Symposium on Graph Drawing (GD 2004)*, *Lecture Notes in Computer Science*, vol. 3383, Springer-Verlag, pp. 482–491.

Hong, S.-H., Merrick, D. and do Nascimento, H. A. D. (2006). Automatic visualisation of metro maps. *Journal of Visual Languages and Computing*, 17(3), pp. 203–224.

Hu, Y. (2005). Efficient, high-quality force-directed graph drawing. *The Mathematica Journal*, 10(1), pp. 37–71.

iMapBuilder (accessed 15/11/2014). `http://www.imapbuilder.com`.

Kamada, T. and Kawai, S. (1989). An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1), pp. 7–15.

Kara, L. B. and Stahovich, T. F. (2004). Hierarchical parsing and recognition of hand sketched diagrams. In *Proceedings of the* 17^{th} *ACM Symposium on User Interface Software and Technology (UIST 2004)*, pp. 13–22.

Lewis, J. (accessed 15/11/2014). The mathematics of force-directed placement. `http://academics.smcvt.edu/jellis-monaghan/Student%20Research/Hudson%20Spring%20Embedder.ppt`.

Lynskey, D. (accessed 23/04/2015). Going underground. `http://www.theguardian.com/culture/culturevultureblog/2006/feb/03/post51`.

Lyons, K. A. (1992). Cluster busting in anchored graph drawing. In *Proceedings of the Conference of the Centre for Advanced Studies on Collaborative Research (CASCON 1992)*, vol. 1, pp. 7–17.

Merrick, D. and Gudmundsson, J. (2007). Path simplification for metro map layout. In *Proceedings of the* 14^{th} *International Symposium on Graph Drawing (GD 2006)*, *Lecture Notes in Computer Science*, vol. 4372, Springer-Verlag, pp. 258–269.

Misue, K. et al. (1995). Layout adjustment and the mental map. *Journal of Visual Languages and Computing*, 6(2), pp. 183–210.

Nielsen, J. (1993). *Usability Engineering*. Morgan Kaufmann.

Nöllenburg, M. (2014). A survey on automated metro map layout methods. *Schematic Mapping Workshop*.

Nöllenburg, M. and Wolff, A. (2006). A mixed-integer program for drawing high-quality metro maps. In *Proceedings of the* 13^{th} *International Symposium*

on Graph Drawing (GD 2005), *Lecture Notes in Computer Science*, vol. 3843, Springer-Verlag, pp. 321–333.

Ovenden, M. (2005). *Metro Maps of the World*. Capital Transport, 2nd edn.

Piggot, P. and Suraweera, F. (1995). Encoding graphs for genetic algorithms: An investigation using the minimum spanning tree problem. In *Proceedings of the AI 1993 and AI 1994 Workshops on Evolutionary Computation*, *Lecture Notes in Computer Science*, vol. 956, Springer-Verlag, pp. 305–314.

Purchase, H. C. (2012). *Experimental Human-Computer Interaction*. Cambridge University Press.

Purchase, H. C., Hoggan, E. and Görg, C. (2007). How important is the "mental map"? – an emperical investigation of a dynamic graph layout algorithm. In *Proceedings of the* 14^{th} *International Symposium on Graph drawing (GD 2006)*, *Lecture Notes in Computer Science*, vol. 4372, Springer-Verlag, pp. 184–195.

Purchase, H. C. and Samra, A. (2008). Extremes are better: Imvestiating mental map preservation in dynamic graphs. In *Proceedings of the* 5^{th} *International Conference on Diagrammatic Representation and Inference (Diagrams 2008)*, *Lecture Notes in Computer Science*, vol. 5223, Springer-Verlag, pp. 60–73.

Quigley, A. and Eades, P. (2001). Fade: Graph drawing, clustering, and visual abstraction. In *Proceedings of the* 8^{th} *International Symposium on Graph Drawing (GD 2000)*, *Lecture Notes in Computer Science*, vol. 1984, Springer-Verlag, pp. 197–210.

Roberts, M. J. (2012). *Underground Maps Unravelled: Explorations in Information Design*. Self published.

Roberts, M. J. et al. (2013). Objective versus subjective measures of paris metro map usability: Investigating traditional octolinear versus all-curves schematics. *International Journal of Human-Computer Studies*, 71(3), pp. 363–386.

Rubine, D. (1991). Specifying gestures by example. In *Proceedings of the* 18^{th} *Annual Conference on Computer Graphics and Interactive Techniques*, pp. 329–337.

Saffrey, P. and Purchase, H. (2008). The "mental map" versus "static aesthetic" compromise in dynamic graphs: A user study. In *Proceedings of the 9th Australasian Conference on User Interface (AUIC 2008)*, vol. 76, pp. 85–93.

Schweitzer, F. et al. (1997). Network optimization using evolutionary strategies. *Journal of Evolutionary Computation*, 5(4), pp. 419–438.

Shewchuk, J. (accessed 02/06/2015). Lecture notes on delaunay mesh generation. `http://www.cs.berkeley.edu/~jrs/meshpapers/delnotes.pdf`.

Stott, J. and Rodgers, P. (2004). Metro map layout using multicriteria optimization. In *Proceedings of the 8th International Conference on Information Visualisation (IV 2004)*, pp. 355–362.

Stott, J. et al. (2010). Automatic metro map layout using multicriteria optimization. *IEEE Transactions on Visualization and Computer Graphics*, 17(1), pp. 101–114.

Sugiyama, K. and Misue, K. (1995). A simple and unified method for drawing graphs: Magnetic-spring algorithm. In *Proceedings of the DIMACS International Workshop (GD 1994)*, *Lecture Notes in Computer Science*, vol. 894, Springer-Verlag, pp. 364–375.

Tory, M., Swindells, C. and Dreezer, R. (2009). Comparing dot and landscape spatializations for visual memory differences. *IEEE Transactions on Visualization and Computer Graphics*, 15(6), pp. 1033–1040.

Tunkelang, D. (1998). Jiggle: Java interactive graph layout environment. In *Proceedings of the 6th International Symposium on Graph Drawing (GD 1998)*, *Lecture Notes in Computer Science*, vol. 1547, Springer-Verlag, pp. 413–422.

Tutte, W. (1963). How to draw a graph. In *Proceedings of the London Math Society*, vol. 3, pp. 743–767.

Walshaw, C. (2001). A multilevel algorithm for force-directed graph drawing. In *Proceedings of the 8th International Symposium on Graph Drawing (GD 2000)*, *Lecture Notes in Computer Science*, vol. 1984, Springer-Verlag, pp. 171–182.

Wang, Y.-S. and Chi, M.-T. (2011). Focus+context metro maps. *IEEE Transactions on Visualization and Computer Graphics*, 17(12), pp. 2528–2535.

Ware, J. M. and Richards, N. (2013). An ant colony system algorithm for automatically schematizing transport network data sets. In *Proceedings of IEEE Congress on Evolutionary Computing*, pp. 1892–1900.

Ware, J. M. et al. (2006). Automated production of schematic maps for mobile applications. *Transactions in GIS*, 10(1), pp. 25–42.

Watson, D. F. (1981). Computing the n-dimensional delaunay tessellation with application to voronoi polytopes. *The Computer Journal*, 24(2), pp. 167–172.

Wu, H.-Y. et al. (2012). Travel-route-centered metro map layout and annotation. *Computer Graphics Forum*, 31(3), pp. 925–934.

Wu, H.-Y. et al. (2013). Spatially efficient design of annotated metro maps. In *Proceedings of the* 15^{th} *Eurographics Conference on Visualization (EuroVis 2013)*, pp. 261–270.

Appendix

Table 14: Fitness statistics by map.

Schematic	Mean	Median	Mode	SD	IQR
Washington	5.162	4.983	4.634	0.793	0.923
Vienna	2.608	2.281	2.189	0.945	1.149
Mexico	4.651	4.379	3.073	1.251	1.621
Sydney	22.027	21.949	21.432	1.683	2.581

Table 15: Fitness statistics by parameter value.

Parameters	Mean	Median	Mode	Standard Deviation	IQR
Grid Spacing					
5	9.557	6.272	3.577	7.740	7.069
6	8.575	5.346	3.548	7.261	7.262
7	8.475	4.894	4.379	7.450	5.741
8	7.973	4.588	3.713	7.140	6.320
9	8.207	4.575	4.190	7.347	6.290
10	8.097	4.156	4.759	7.994	7.309
11	8.112	4.185	3.058	7.852	6.057
12	8.690	4.153	3.073	8.852	7.745
13	8.812	4.938	1.819	8.369	7.487
14	9.118	4.637	4.634	8.618	6.380
15	9.114	5.481	2.189	8.216	7.857
Search Distance					
5	9.612	5.680	-	8.037	6.900
6	9.261	5.481	-	8.098	7.441
7	9.023	5.308	3.705	7.939	7.465
8	8.669	5.004	5.157	7.878	7.435
9	8.630	4.812	3.957	7.916	7.719
10	8.473	4.806	19.985	7.851	7.212
11	8.414	4.635	19.662	7.857	6.954
12	8.225	4.411	3.713	7.836	6.689
13	8.216	4.396	4.379	7.888	6.787
14	8.130	4.365	19.660	7.871	7.089
15	8.078	4.332	4.327	7.875	6.902
Cooling Schedule					
None	8.113	4.386	2.189	7.808	7.210
Linear	8.421	4.657	1.819	7.848	7.880
Exponential	9.301	5.524	5.380	8.012	7.356

Table 16: Mexico results (1 of 3).

Rank	Search Distance	Grid Spacing	Cooling	Iterations	Fitness	Rank	Search Distance	Grid Spacing	Cooling	Iterations	Fitness
1	15	12	None	8	2.653	71	6	10	Linear	8	3.631
2	12	14	None	9	2.660	72	6	10	None	7	3.632
3	10	14	None	11	2.667	73	15	13	None	8	3.684
4	15	14	Linear	10	2.667	74	11	8	None	7	3.690
5	8	12	None	8	2.771	75	11	6	None	8	3.699
6	13	12	None	8	2.802	76	7	11	None	10	3.705
7	14	12	None	7	2.828	77	7	11	Linear	10	3.705
8	14	10	Linear	7	2.847	78	5	10	None	7	3.705
9	13	10	Linear	7	2.856	79	12	8	None	5	3.713
10	15	10	None	9	2.896	80	13	8	None	5	3.713
11	8	10	None	14	2.897	81	12	8	Linear	7	3.713
12	12	11	None	5	2.980	82	13	8	Linear	5	3.713
13	13	11	Linear	5	2.980	83	14	8	Linear	5	3.713
14	14	14	None	7	3.029	84	15	8	Linear	5	3.713
15	11	11	None	5	3.041	85	12	15	None	12	3.741
16	11	11	Linear	5	3.041	86	13	15	None	12	3.741
17	12	11	Linear	5	3.041	87	14	15	None	12	3.741
18	15	14	None	6	3.044	88	15	15	None	12	3.741
19	10	11	None	5	3.051	89	7	12	Linear	6	3.743
20	10	11	Linear	5	3.051	90	6	12	None	4	3.761
21	13	11	None	6	3.058	91	12	11	Exponential	4	3.761
22	14	11	None	6	3.058	92	13	11	Exponential	4	3.761
23	15	11	None	6	3.058	93	13	6	None	7	3.789
24	14	11	Linear	6	3.058	94	8	10	Linear	7	3.791
25	15	11	Linear	6	3.058	95	5	12	None	9	3.799
26	10	12	None	6	3.073	96	6	11	None	9	3.803
27	11	12	None	6	3.073	97	9	14	Linear	7	3.816
28	12	12	None	6	3.073	98	14	5	None	9	3.866
29	11	12	Linear	6	3.073	99	6	11	Linear	6	3.877
30	12	12	Linear	6	3.073	100	9	13	None	6	3.879
31	13	12	Linear	6	3.073	101	12	12	Exponential	6	3.879
32	14	12	Linear	6	3.073	102	13	12	Exponential	6	3.879
33	15	12	Linear	6	3.073	103	15	9	None	8	3.885
34	9	12	None	6	3.074	104	14	10	Exponential	5	3.902
35	10	12	Linear	6	3.074	105	15	10	Exponential	5	3.902
36	9	12	Linear	8	3.143	106	5	10	Linear	7	3.918
37	13	14	None	7	3.144	107	7	9	Linear	7	3.926
38	11	14	Linear	8	3.219	108	7	10	None	7	3.933
39	14	11	Exponential	5	3.221	109	8	10	Exponential	9	3.936
40	15	11	Exponential	5	3.221	110	7	10	Linear	9	3.939
41	12	6	None	8	3.236	111	9	10	None	6	3.957
42	11	14	None	5	3.265	112	9	10	Linear	6	3.957
43	14	14	Linear	5	3.265	113	9	11	Exponential	4	3.976
44	8	14	None	8	3.283	114	7	13	Linear	6	3.998
45	9	14	None	8	3.283	115	11	11	Exponential	6	4.002
46	13	14	Linear	6	3.285	116	10	11	Exponential	6	4.011
47	12	14	Linear	7	3.308	117	6	9	None	7	4.024
48	9	11	None	7	3.372	118	11	6	Linear	8	4.030
49	5	11	None	10	3.377	119	12	10	None	6	4.038
50	9	11	Linear	6	3.385	120	6	13	None	5	4.039
51	7	12	None	7	3.407	121	11	10	None	6	4.042
52	8	12	Linear	7	3.407	122	13	13	None	6	4.056
53	15	12	Exponential	7	3.407	123	14	13	Linear	6	4.056
54	13	10	None	6	3.470	124	8	12	Exponential	5	4.060
55	14	10	None	6	3.470	125	9	12	Exponential	5	4.067
56	15	10	Linear	6	3.470	126	8	11	Exponential	6	4.076
57	10	14	Linear	8	3.502	127	13	6	Linear	10	4.077
58	14	8	None	7	3.512	128	5	13	None	6	4.081
59	15	8	None	7	3.512	129	9	10	Exponential	5	4.088
60	11	8	Linear	6	3.522	130	15	9	Linear	6	4.098
61	15	6	None	8	3.525	131	10	10	None	6	4.107
62	12	6	Linear	10	3.528	132	10	10	Linear	6	4.107
63	14	6	Linear	7	3.531	133	11	10	Linear	6	4.107
64	8	11	None	8	3.532	134	12	10	Linear	6	4.107
65	8	11	Linear	8	3.532	135	14	13	None	5	4.110
66	6	12	Linear	6	3.547	136	15	13	Linear	5	4.110
67	14	6	None	8	3.548	137	12	10	Exponential	8	4.113
68	15	6	Linear	8	3.548	138	10	13	None	6	4.119
69	14	12	Exponential	6	3.554	139	12	13	None	6	4.119
70	15	7	None	9	3.624	140	13	13	Linear	6	4.119

Table 17: Mexico results (2 of 3).

Rank	Search Distance	Grid Spacing	Cooling	Iterations	Fitness	Rank	Search Distance	Grid Spacing	Cooling	Iterations	Fitness
141	7	11	Exponential	6	4.129	211	11	9	None	7	4.558
142	11	13	None	6	4.132	212	11	9	Linear	7	4.558
143	12	13	Linear	6	4.132	213	12	9	Linear	7	4.558
144	8	8	None	10	4.141	214	13	9	Linear	7	4.558
145	10	12	Exponential	5	4.153	215	15	5	None	9	4.575
146	11	12	Exponential	5	4.153	216	5	14	None	7	4.577
147	10	6	None	7	4.154	217	6	14	Linear	7	4.582
148	6	14	None	6	4.163	218	11	8	Exponential	6	4.586
149	7	13	None	6	4.168	219	6	8	None	7	4.591
150	8	13	None	6	4.168	220	9	9	None	5	4.620
151	8	13	Linear	6	4.168	221	14	15	Linear	9	4.627
152	9	13	Linear	6	4.168	222	13	15	Linear	9	4.628
153	6	10	Exponential	8	4.201	223	13	5	Linear	15	4.632
154	14	7	None	5	4.213	224	9	6	Linear	11	4.650
155	13	10	Exponential	9	4.243	225	10	9	Linear	5	4.656
156	10	10	Exponential	5	4.244	226	9	6	None	7	4.666
157	11	10	Exponential	5	4.244	227	7	10	Exponential	8	4.676
158	10	6	Linear	10	4.260	228	7	12	Exponential	6	4.690
159	12	8	Exponential	8	4.280	229	6	11	Exponential	7	4.740
160	13	8	Exponential	8	4.280	230	15	5	Linear	12	4.741
161	5	10	Exponential	10	4.287	231	10	8	Exponential	6	4.779
162	7	9	None	6	4.294	232	11	7	None	7	4.785
163	12	7	None	11	4.296	233	9	8	Exponential	8	4.805
164	10	8	None	7	4.311	234	8	13	Exponential	5	4.820
165	7	6	None	7	4.312	235	9	13	Exponential	5	4.820
166	5	12	Linear	7	4.330	236	8	7	None	6	4.836
167	8	9	Linear	8	4.336	237	7	8	Linear	8	4.842
168	15	9	Exponential	8	4.337	238	10	14	Exponential	6	4.850
169	12	13	Exponential	6	4.341	239	11	14	Exponential	6	4.850
170	13	13	Exponential	6	4.341	240	10	7	None	8	4.886
171	9	8	Linear	7	4.343	241	6	7	None	7	4.886
172	13	5	None	10	4.343	242	5	14	Linear	6	4.900
173	9	8	None	7	4.343	243	15	6	Exponential	9	4.932
174	5	13	Linear	6	4.351	244	10	5	None	10	4.933
175	14	8	Exponential	7	4.351	245	8	15	None	9	4.938
176	7	8	None	8	4.372	246	14	6	Exponential	8	4.947
177	8	9	None	6	4.377	247	8	6	None	8	4.960
178	9	9	Linear	6	4.377	248	9	14	Exponential	6	4.961
179	13	7	None	6	4.379	249	7	15	None	6	4.984
180	13	7	Linear	6	4.379	250	11	7	Linear	7	4.995
181	14	7	Linear	6	4.379	251	6	8	Linear	10	4.996
182	15	7	Linear	6	4.379	252	8	5	None	12	5.009
183	7	14	Linear	6	4.379	253	7	14	Exponential	8	5.048
184	12	9	None	7	4.392	254	8	8	Exponential	7	5.070
185	13	9	None	7	4.392	255	8	9	Exponential	11	5.076
186	14	9	None	7	4.392	256	12	7	Linear	6	5.097
187	14	9	Linear	7	4.392	257	8	7	Linear	8	5.131
188	10	13	Exponential	5	4.393	258	10	9	Exponential	7	5.138
189	11	13	Exponential	5	4.393	259	11	9	Exponential	7	5.138
190	10	13	Linear	5	4.409	260	9	7	Linear	5	5.168
191	11	13	Linear	5	4.409	261	9	9	Exponential	6	5.181
192	15	13	Exponential	5	4.409	262	8	5	Linear	12	5.233
193	14	5	Linear	9	4.419	263	5	11	Linear	5	5.235
194	8	14	Linear	4	4.421	264	6	8	Exponential	9	5.238
195	15	14	Exponential	4	4.421	265	6	14	Exponential	6	5.241
196	14	14	Exponential	6	4.422	266	5	8	None	7	5.245
197	12	14	Exponential	6	4.431	267	7	15	Linear	6	5.248
198	13	14	Exponential	6	4.431	268	12	7	Exponential	11	5.267
199	12	5	None	10	4.445	269	9	7	None	5	5.277
200	15	8	Exponential	7	4.451	270	5	7	None	6	5.356
201	8	8	Linear	7	4.455	271	7	8	Exponential	10	5.369
202	14	9	Exponential	8	4.483	272	13	6	Exponential	10	5.369
203	7	14	None	8	4.483	273	6	12	Exponential	5	5.395
204	12	5	Linear	11	4.493	274	7	13	Exponential	5	5.402
205	14	13	Exponential	5	4.523	275	10	7	Linear	7	5.421
206	10	8	Linear	6	4.523	276	8	7	Exponential	7	5.428
207	6	13	Linear	5	4.539	277	8	14	Exponential	6	5.434
208	12	9	Exponential	5	4.544	278	5	13	Exponential	4	5.441
209	13	9	Exponential	5	4.544	279	15	7	Exponential	8	5.441
210	10	9	None	7	4.558	280	5	14	Exponential	8	5.444

Table 18: Mexico results (3 of 3).

Rank	Search Distance	Grid Spacing	Cooling	Iterations	Fitness	Rank	Search Distance	Grid Spacing	Cooling	Iterations	Fitness
281	11	5	None	6	5.454	323	6	15	None	6	6.314
282	6	7	Linear	7	5.466	324	6	6	Exponential	10	6.371
283	6	9	Linear	6	5.516	325	5	6	None	8	6.419
284	5	9	None	7	5.526	326	5	5	None	16	6.427
285	5	8	Linear	8	5.533	327	11	7	Exponential	11	6.451
286	7	7	None	7	5.535	328	5	7	Exponential	4	6.462
287	15	15	Linear	7	5.539	329	5	8	Exponential	21	6.527
288	7	7	Linear	6	5.544	330	14	5	Exponential	29	6.586
289	6	13	Exponential	4	5.555	331	9	5	Linear	8	6.642
290	9	15	None	8	5.577	332	8	6	Exponential	10	6.653
291	9	15	Linear	8	5.577	333	10	7	Exponential	5	6.673
292	10	15	Linear	8	5.577	334	12	5	Exponential	13	6.690
293	9	7	Exponential	8	5.611	335	6	15	Linear	6	6.702
294	11	6	Exponential	8	5.621	336	13	5	Exponential	20	6.702
295	7	6	Linear	7	5.669	337	15	5	Exponential	13	6.732
296	14	7	Exponential	10	5.673	338	8	15	Exponential	5	6.821
297	13	7	Exponential	14	5.683	339	14	15	Exponential	4	6.847
298	9	5	None	10	5.690	340	5	9	Linear	7	6.895
299	5	11	Exponential	5	5.700	341	6	15	Exponential	6	6.927
300	12	6	Exponential	10	5.718	342	7	5	None	12	7.014
301	5	12	Exponential	6	5.749	343	5	15	None	9	7.109
302	10	6	Exponential	6	5.753	344	10	15	Exponential	5	7.151
303	7	7	Exponential	12	5.761	345	11	15	Exponential	5	7.151
304	15	15	Exponential	8	5.762	346	9	15	Exponential	5	7.151
305	10	5	Linear	5	5.814	347	5	9	Exponential	6	7.154
306	9	6	Exponential	8	5.830	348	11	5	Exponential	31	7.157
307	6	9	Exponential	7	5.902	349	10	5	Exponential	14	7.211
308	8	15	Linear	6	5.982	350	6	5	None	10	7.253
309	6	6	None	6	6.003	351	7	15	Exponential	3	7.401
310	5	7	Linear	10	6.037	352	5	15	Linear	6	7.645
311	11	5	Linear	6	6.043	353	6	5	Linear	9	7.682
312	12	15	Exponential	7	6.104	354	5	6	Linear	8	7.794
313	13	15	Exponential	7	6.104	355	8	5	Exponential	27	8.104
314	6	6	Linear	6	6.127	356	7	5	Linear	15	8.117
315	10	15	None	6	6.177	357	5	6	Exponential	11	8.129
316	11	15	None	6	6.177	358	9	5	Exponential	5	8.191
317	11	15	Linear	6	6.177	359	5	15	Exponential	6	8.320
318	12	15	Linear	6	6.177	360	5	5	Linear	10	8.422
319	6	7	Exponential	9	6.244	361	5	5	Exponential	15	8.506
320	7	6	Exponential	14	6.281	362	7	5	Exponential	10	8.632
321	7	9	Exponential	16	6.284	363	6	5	Exponential	13	8.712
322	8	6	Linear	5	6.296						

Table 19: Sydney results (1 of 3).

Rank	Search Distance	Grid Spacing	Cooling	Iterations	Fitness	Rank	Search Distance	Grid Spacing	Cooling	Iterations	Fitness
1	11	11	None	7	19.359	71	6	7	None	9	20.249
2	6	9	None	9	19.362	72	9	6	Linear	4	20.283
3	15	6	None	8	19.435	73	9	7	None	7	20.295
4	11	6	None	13	19.437	74	12	9	None	8	20.320
5	12	6	Linear	11	19.467	75	5	7	None	8	20.329
6	5	9	None	7	19.535	76	11	9	None	9	20.332
7	8	9	None	9	19.542	77	8	9	Exponential	7	20.367
8	10	6	None	8	19.549	78	11	9	Linear	9	20.412
9	11	7	None	9	19.574	79	10	9	None	8	20.419
10	13	11	None	7	19.592	80	12	9	Linear	9	20.447
11	11	6	Linear	6	19.594	81	9	7	Linear	7	20.448
12	10	11	Linear	10	19.595	82	6	8	None	8	20.486
13	8	6	None	11	19.626	83	12	9	Exponential	11	20.495
14	10	11	None	6	19.643	84	6	8	Linear	9	20.505
15	11	11	Linear	6	19.643	85	8	6	Linear	8	20.536
16	15	11	None	6	19.652	86	5	8	None	7	20.550
17	14	8	None	7	19.660	87	7	7	None	8	20.589
18	14	8	Linear	7	19.660	88	12	5	None	9	20.613
19	11	8	None	4	19.662	89	15	9	Exponential	7	20.628
20	12	8	None	4	19.662	90	9	5	None	11	20.663
21	11	8	Linear	4	19.662	91	7	7	Linear	9	20.683
22	12	8	Linear	4	19.662	92	11	5	None	7	20.684
23	13	8	None	7	19.663	93	12	11	Exponential	7	20.685
24	13	8	Linear	7	19.663	94	13	11	Exponential	7	20.685
25	14	8	Exponential	6	19.665	95	14	7	None	8	20.690
26	14	11	None	7	19.675	96	15	7	None	8	20.690
27	12	6	None	9	19.679	97	15	7	Linear	7	20.691
28	10	6	Linear	12	19.703	98	12	7	Linear	7	20.692
29	11	7	Linear	9	19.710	99	14	6	Exponential	12	20.695
30	7	6	None	12	19.711	100	12	7	None	8	20.697
31	15	8	None	6	19.730	101	13	7	None	8	20.697
32	15	8	Linear	6	19.730	102	13	7	Linear	8	20.697
33	15	8	Exponential	6	19.738	103	14	7	Linear	8	20.697
34	14	6	Linear	9	19.740	104	10	9	Linear	8	20.756
35	11	8	Exponential	7	19.746	105	13	9	Exponential	9	20.760
36	14	6	None	7	19.747	106	14	9	Exponential	9	20.769
37	15	6	Linear	7	19.747	107	11	5	Linear	7	20.771
38	8	7	None	8	19.769	108	14	10	None	7	20.808
39	13	8	Exponential	9	19.829	109	8	7	Exponential	12	20.816
40	9	6	None	9	19.829	110	7	8	Linear	7	20.837
41	10	7	None	9	19.850	111	7	9	None	9	20.856
42	8	7	Linear	7	19.914	112	7	9	Linear	12	20.859
43	8	8	Linear	5	19.945	113	9	9	None	8	20.864
44	10	7	Linear	9	19.961	114	10	10	None	8	20.868
45	12	11	None	6	19.975	115	10	9	Exponential	5	20.875
46	12	11	Linear	6	19.975	116	7	8	None	8	20.879
47	13	11	Linear	6	19.975	117	8	5	None	14	20.885
48	14	11	Linear	6	19.975	118	10	5	None	9	20.889
49	15	11	Linear	6	19.975	119	11	9	Exponential	8	20.941
50	8	8	None	3	19.978	120	5	9	Linear	5	20.954
51	9	8	Linear	3	19.978	121	15	10	None	8	20.981
52	9	8	None	3	19.985	122	12	5	Linear	9	20.988
53	10	8	None	3	19.985	123	6	7	Linear	9	20.990
54	10	8	Linear	3	19.985	124	5	8	Linear	8	21.046
55	12	8	Exponential	5	20.021	125	12	10	None	7	21.052
56	10	8	Exponential	6	20.093	126	13	10	None	7	21.052
57	9	8	Exponential	4	20.103	127	12	10	Linear	7	21.052
58	14	11	Exponential	7	20.120	128	13	10	Linear	7	21.052
59	15	11	Exponential	7	20.120	129	14	10	Linear	7	21.102
60	13	9	Linear	9	20.121	130	15	10	Linear	8	21.151
61	8	8	Exponential	4	20.189	131	9	9	Linear	5	21.153
62	13	6	None	4	20.191	132	7	6	Linear	5	21.196
63	8	9	Linear	6	20.208	133	6	8	Exponential	7	21.196
64	6	9	Linear	6	20.215	134	12	6	Exponential	13	21.202
65	15	6	Exponential	7	20.215	135	14	7	Exponential	9	21.211
66	13	6	Linear	4	20.220	136	15	7	Exponential	9	21.211
67	13	9	None	9	20.242	137	15	9	None	4	21.216
68	14	9	None	9	20.242	138	5	7	Linear	11	21.271
69	14	9	Linear	9	20.242	139	13	5	None	9	21.274
70	15	9	Linear	9	20.242	140	10	10	Linear	9	21.295

Table 20: Sydney results (2 of 3).

Rank	Search Distance	Grid Spacing	Cooling	Iterations	Fitness	Rank	Search Distance	Grid Spacing	Cooling	Iterations	Fitness
141	7	8	Exponential	14	21.299	211	10	6	Exponential	4	22.275
142	11	10	None	7	21.314	212	9	9	Exponential	5	22.289
143	11	10	Linear	7	21.314	213	9	13	Linear	8	22.364
144	5	6	None	5	21.377	214	5	10	None	9	22.376
145	14	10	Exponential	5	21.401	215	9	15	Linear	7	22.391
146	14	5	None	11	21.416	216	6	5	Linear	12	22.402
147	15	5	Linear	12	21.422	217	9	10	Linear	7	22.416
148	12	15	None	9	21.432	218	9	10	Exponential	7	22.430
149	13	15	None	9	21.432	219	9	6	Exponential	4	22.434
150	14	15	None	9	21.432	220	8	12	Linear	8	22.441
151	15	15	None	9	21.432	221	6	10	Exponential	8	22.482
152	14	15	Linear	9	21.432	222	8	11	None	11	22.519
153	15	15	Linear	9	21.432	223	15	14	None	9	22.573
154	7	10	None	11	21.438	224	9	11	Linear	10	22.590
155	5	6	Linear	12	21.476	225	7	15	None	6	22.603
156	12	10	Exponential	9	21.483	226	8	15	None	6	22.603
157	13	10	Exponential	9	21.483	227	8	15	Linear	6	22.603
158	11	10	Exponential	7	21.557	228	13	14	None	8	22.615
159	13	6	Exponential	10	21.558	229	14	14	None	8	22.615
160	9	10	None	7	21.558	230	15	14	Linear	8	22.615
161	9	12	None	8	21.588	231	10	13	Linear	9	22.632
162	15	10	Exponential	7	21.601	232	6	6	Linear	14	22.633
163	7	12	None	7	21.604	233	5	11	None	7	22.638
164	6	6	None	12	21.649	234	7	15	Linear	6	22.653
165	5	8	Exponential	13	21.651	235	5	10	Linear	9	22.658
166	7	5	Linear	8	21.654	236	8	5	Linear	4	22.689
167	8	12	None	7	21.688	237	10	12	Linear	7	22.750
168	9	11	None	8	21.720	238	10	12	None	9	22.751
169	15	5	None	8	21.747	239	8	11	Linear	8	22.764
170	7	5	None	9	21.824	240	10	5	Linear	10	22.803
171	12	15	Linear	7	21.853	241	10	13	None	5	22.816
172	13	15	Linear	7	21.853	242	5	11	Linear	8	22.833
173	12	13	None	10	21.859	243	6	13	None	9	22.838
174	10	11	Exponential	6	21.861	244	15	15	Exponential	6	22.842
175	11	11	Exponential	6	21.861	245	7	10	Exponential	13	22.880
176	11	13	None	7	21.884	246	8	10	Exponential	10	22.889
177	12	13	Linear	7	21.884	247	5	6	Exponential	4	22.992
178	9	7	Exponential	10	21.890	248	12	15	Exponential	7	23.020
179	10	10	Exponential	7	21.905	249	13	15	Exponential	7	23.020
180	12	7	Exponential	10	21.944	250	14	15	Exponential	7	23.020
181	13	7	Exponential	10	21.944	251	6	11	None	8	23.026
182	10	15	None	7	21.949	252	11	13	Linear	7	23.047
183	11	15	None	7	21.949	253	12	5	Exponential	13	23.053
184	14	5	Linear	11	21.957	254	8	14	None	7	23.066
185	8	10	None	8	21.977	255	5	11	Exponential	15	23.069
186	13	13	None	7	21.982	256	7	11	Linear	7	23.142
187	14	13	None	7	21.982	257	8	13	None	9	23.144
188	14	13	Linear	7	21.982	258	6	13	Linear	13	23.170
189	15	13	Linear	7	21.982	259	8	5	Exponential	16	23.171
190	7	9	Exponential	5	21.998	260	7	13	None	9	23.207
191	10	14	None	11	22.018	261	9	5	Linear	11	23.210
192	15	13	None	6	22.020	262	14	13	Exponential	8	23.228
193	13	13	Linear	7	22.028	263	15	13	Exponential	8	23.228
194	6	9	Exponential	6	22.036	264	11	14	None	9	23.232
195	11	6	Exponential	5	22.062	265	12	14	None	9	23.232
196	9	15	None	9	22.098	266	11	14	Linear	9	23.232
197	9	13	None	8	22.108	267	12	14	Linear	9	23.232
198	6	10	None	6	22.140	268	13	14	Linear	9	23.232
199	11	15	Linear	7	22.146	269	14	14	Linear	9	23.232
200	8	6	Exponential	4	22.147	270	12	13	Exponential	7	23.241
201	13	5	Linear	11	22.156	271	13	13	Exponential	7	23.241
202	8	10	Linear	9	22.162	272	9	12	Exponential	10	23.262
203	5	5	None	8	22.167	273	10	14	Linear	10	23.267
204	6	5	None	7	22.179	274	10	7	Exponential	15	23.287
205	7	10	Linear	12	22.184	275	5	9	Exponential	6	23.350
206	6	10	Linear	6	22.215	276	5	10	Exponential	13	23.369
207	10	15	Linear	7	22.228	277	5	5	Linear	8	23.400
208	7	11	None	6	22.249	278	11	13	Exponential	6	23.420
209	9	12	Linear	7	22.261	279	11	7	Exponential	5	23.437
210	7	6	Exponential	4	22.268	280	10	15	Exponential	13	23.494

Table 21: Sydney results (3 of 3).

Rank	Search Distance	Grid Spacing	Cooling	Iterations	Fitness	Rank	Search Distance	Grid Spacing	Cooling	Iterations	Fitness
281	11	15	Exponential	13	23.494	323	6	5	Exponential	10	24.222
282	8	12	Exponential	7	23.495	324	8	14	Linear	10	24.254
283	9	14	Exponential	7	23.518	325	11	12	Linear	7	24.255
284	7	12	Linear	7	23.519	326	10	14	Exponential	6	24.267
285	8	13	Linear	9	23.603	327	11	12	None	5	24.332
286	7	13	Linear	9	23.627	328	12	12	Linear	5	24.332
287	15	14	Exponential	8	23.629	329	13	12	Linear	5	24.332
288	13	12	None	7	23.666	330	11	14	Exponential	6	24.356
289	7	11	Exponential	8	23.675	331	6	15	None	7	24.395
290	15	12	Linear	6	23.678	332	14	5	Exponential	5	24.397
291	7	7	Exponential	15	23.693	333	5	14	None	9	24.491
292	9	13	Exponential	8	23.705	334	8	14	Exponential	6	24.506
293	7	12	Exponential	8	23.712	335	6	6	Exponential	4	24.561
294	14	12	None	5	23.723	336	6	14	None	7	24.568
295	15	12	None	5	23.723	337	9	5	Exponential	7	24.610
296	9	14	None	7	23.729	338	11	5	Exponential	5	24.677
297	6	12	None	8	23.759	339	5	13	Linear	8	24.683
298	7	13	Exponential	8	23.766	340	5	12	Exponential	8	24.741
299	10	13	Exponential	7	23.775	341	8	15	Exponential	5	24.759
300	7	5	Exponential	12	23.776	342	5	15	Linear	5	24.761
301	5	7	Exponential	3	23.829	343	6	11	Exponential	4	24.764
302	15	5	Exponential	9	23.858	344	7	14	Linear	7	24.787
303	8	13	Exponential	8	23.901	345	7	15	Exponential	6	24.806
304	6	12	Linear	6	23.904	346	14	12	Exponential	5	24.919
305	14	12	Linear	7	23.908	347	15	12	Exponential	5	24.919
306	9	15	Exponential	6	23.910	348	6	12	Exponential	7	24.926
307	6	13	Exponential	7	23.933	349	6	14	Linear	6	24.931
308	5	13	None	6	23.960	350	6	15	Linear	6	25.009
309	6	11	Linear	7	23.966	351	5	14	Linear	8	25.092
310	12	14	Exponential	8	23.974	352	5	5	Exponential	9	25.107
311	13	14	Exponential	8	23.974	353	10	12	Exponential	4	25.122
312	9	14	Linear	6	23.980	354	11	12	Exponential	8	25.337
313	5	12	None	7	23.999	355	12	12	Exponential	6	25.361
314	7	14	None	9	24.010	356	13	12	Exponential	6	25.361
315	14	14	Exponential	8	24.043	357	10	5	Exponential	5	25.483
316	8	11	Exponential	11	24.075	358	5	13	Exponential	7	25.719
317	6	7	Exponential	15	24.091	359	6	15	Exponential	9	25.797
318	13	5	Exponential	12	24.101	360	7	14	Exponential	11	25.804
319	12	12	None	6	24.118	361	6	14	Exponential	7	25.973
320	9	11	Exponential	4	24.160	362	5	14	Exponential	9	26.286
321	5	12	Linear	8	24.204	363	5	15	Exponential	8	26.964
322	5	15	None	4	24.220						

Table 22: Vienna results (1 of 3).

Rank	Search Distance	Grid Spacing	Cooling	Iterations	Fitness	Rank	Search Distance	Grid Spacing	Cooling	Iterations	Fitness
1	15	10	None	9	0.850	71	13	13	None	7	1.819
2	13	10	None	9	0.972	72	14	13	None	7	1.819
3	14	10	None	9	0.986	73	15	13	None	7	1.819
4	15	10	Linear	9	1.037	74	6	10	None	8	1.827
5	14	5	None	15	1.153	75	15	8	None	11	1.831
6	15	5	None	13	1.184	76	13	6	Linear	9	1.836
7	15	7	None	14	1.212	77	14	8	Linear	8	1.838
8	11	10	None	7	1.233	78	7	10	None	9	1.858
9	9	10	None	17	1.276	79	15	9	Linear	7	1.872
10	12	9	None	8	1.284	80	13	8	Linear	8	1.898
11	10	15	Exponential	7	1.341	81	5	9	None	13	1.905
12	11	15	Exponential	7	1.341	82	10	6	None	5	1.911
13	11	10	Linear	6	1.353	83	6	6	None	11	1.914
14	12	10	Linear	7	1.361	84	10	10	Linear	11	1.916
15	13	10	Linear	7	1.380	85	9	8	None	9	1.946
16	14	10	Linear	7	1.380	86	5	15	None	10	1.951
17	15	7	Linear	10	1.411	87	10	8	Linear	10	1.956
18	13	8	None	15	1.427	88	5	12	Linear	9	1.963
19	15	8	Linear	13	1.439	89	10	8	None	8	1.966
20	9	6	None	11	1.458	90	11	5	None	10	1.974
21	14	5	Linear	13	1.467	91	7	9	None	7	1.987
22	8	14	None	7	1.468	92	12	15	Exponential	5	1.999
23	12	5	None	10	1.475	93	13	15	Exponential	5	1.999
24	13	7	None	12	1.477	94	15	11	None	7	2.006
25	13	5	None	10	1.481	95	15	12	None	7	2.015
26	10	10	None	8	1.485	96	8	10	Linear	9	2.025
27	8	6	None	14	1.505	97	14	11	None	8	2.031
28	15	5	Linear	12	1.506	98	12	8	Linear	8	2.033
29	14	7	Linear	10	1.506	99	11	8	Linear	7	2.042
30	8	10	None	9	1.508	100	7	15	None	8	2.059
31	13	5	Linear	12	1.530	101	11	5	Linear	10	2.061
32	12	10	None	8	1.542	102	9	10	Linear	9	2.069
33	11	9	None	10	1.547	103	7	6	None	10	2.072
34	15	9	None	12	1.575	104	14	6	None	8	2.075
35	8	14	Linear	7	1.587	105	8	13	Linear	7	2.077
36	13	9	None	11	1.625	106	12	13	Exponential	7	2.078
37	14	9	None	10	1.625	107	13	13	Exponential	7	2.078
38	10	6	Linear	8	1.646	108	14	13	Exponential	7	2.078
39	11	7	None	13	1.666	109	15	13	Exponential	6	2.078
40	12	5	Linear	13	1.683	110	13	6	None	10	2.086
41	15	6	None	10	1.683	111	8	7	None	11	2.091
42	14	7	None	10	1.689	112	10	5	None	13	2.099
43	7	15	Linear	7	1.699	113	11	9	Linear	9	2.103
44	11	8	None	8	1.705	114	10	9	None	9	2.107
45	8	13	None	6	1.716	115	11	11	None	9	2.113
46	10	7	None	12	1.718	116	7	13	Linear	7	2.115
47	14	6	Linear	11	1.720	117	7	13	None	7	2.115
48	5	12	None	7	1.724	118	10	13	Exponential	6	2.126
49	9	7	None	12	1.735	119	11	13	Exponential	6	2.126
50	12	7	None	11	1.762	120	15	11	Linear	8	2.146
51	15	6	Linear	8	1.765	121	12	11	None	8	2.146
52	9	9	None	8	1.770	122	13	11	None	8	2.157
53	12	8	None	8	1.776	123	6	12	Linear	8	2.161
54	12	7	Linear	9	1.779	124	6	8	None	12	2.162
55	12	9	Linear	7	1.785	125	9	13	Exponential	7	2.165
56	14	12	None	7	1.787	126	8	11	None	7	2.177
57	14	8	None	8	1.792	127	8	8	Linear	9	2.179
58	14	9	Linear	7	1.811	128	15	14	None	8	2.185
59	8	9	None	8	1.814	129	9	14	None	9	2.189
60	9	13	Linear	7	1.819	130	9	15	Linear	7	2.189
61	10	13	Linear	7	1.819	131	10	15	Linear	7	2.189
62	11	13	Linear	7	1.819	132	11	15	Linear	7	2.189
63	12	13	Linear	7	1.819	133	12	15	Linear	7	2.189
64	13	13	Linear	7	1.819	134	13	15	Linear	7	2.189
65	14	13	Linear	7	1.819	135	14	15	Linear	7	2.189
66	15	13	Linear	7	1.819	136	15	15	Linear	7	2.189
67	9	13	None	7	1.819	137	8	15	None	7	2.189
68	10	13	None	7	1.819	138	9	15	None	7	2.189
69	11	13	None	7	1.819	139	10	15	None	7	2.189
70	12	13	None	7	1.819	140	11	15	None	7	2.189

Table 23: Vienna results (2 of 3).

Rank	Search Distance	Grid Spacing	Cooling	Iterations	Fitness	Rank	Search Distance	Grid Spacing	Cooling	Iterations	Fitness
141	12	15	None	7	2.189	211	15	7	Exponential	9	2.505
142	13	15	None	7	2.189	212	10	5	Linear	14	2.534
143	14	15	None	7	2.189	213	14	10	Exponential	7	2.536
144	15	15	None	7	2.189	214	15	10	Exponential	7	2.536
145	7	13	Exponential	8	2.189	215	10	7	Linear	10	2.544
146	14	15	Exponential	8	2.195	216	9	5	None	10	2.553
147	15	15	Exponential	8	2.195	217	12	6	Linear	9	2.556
148	8	15	Linear	8	2.195	218	10	10	Exponential	7	2.595
149	13	9	Linear	8	2.216	219	11	10	Exponential	7	2.595
150	6	9	None	11	2.219	220	11	7	Linear	9	2.601
151	8	13	Exponential	7	2.219	221	6	8	Linear	13	2.607
152	8	12	Linear	8	2.220	222	15	8	Exponential	9	2.607
153	9	12	Linear	8	2.220	223	7	10	Exponential	12	2.627
154	10	12	Linear	8	2.220	224	15	9	Exponential	8	2.635
155	6	12	None	8	2.220	225	10	8	Exponential	14	2.638
156	8	12	None	8	2.220	226	11	6	Linear	13	2.675
157	9	12	None	8	2.220	227	9	10	Exponential	5	2.679
158	10	12	None	8	2.220	228	8	6	Linear	10	2.679
159	11	12	Linear	8	2.221	229	6	15	None	5	2.690
160	12	12	Linear	8	2.221	230	6	10	Linear	6	2.717
161	13	12	Linear	8	2.221	231	12	9	Exponential	13	2.805
162	14	12	Linear	8	2.221	232	6	15	Linear	5	2.813
163	15	12	Linear	8	2.221	233	8	12	Exponential	9	2.815
164	11	12	None	8	2.221	234	12	10	Exponential	6	2.816
165	12	12	None	8	2.221	235	13	10	Exponential	6	2.816
166	13	12	None	8	2.221	236	7	10	Linear	6	2.816
167	7	8	Linear	10	2.222	237	9	14	Exponential	9	2.818
168	10	14	Linear	11	2.231	238	7	7	Linear	10	2.820
169	15	12	Exponential	7	2.235	239	8	10	Exponential	4	2.828
170	7	12	Linear	7	2.250	240	8	7	Linear	8	2.852
171	7	12	None	6	2.250	241	6	10	Exponential	6	2.866
172	12	12	Exponential	8	2.251	242	10	11	Exponential	9	2.868
173	13	12	Exponential	8	2.251	243	11	11	Exponential	9	2.868
174	14	12	Exponential	8	2.251	244	11	7	Exponential	17	2.873
175	6	13	None	6	2.254	245	11	6	None	9	2.883
176	12	11	Linear	7	2.258	246	9	11	Exponential	12	2.892
177	13	11	Linear	7	2.258	247	12	8	Exponential	9	2.899
178	14	11	Linear	7	2.258	248	13	8	Exponential	9	2.899
179	7	9	Linear	10	2.259	249	14	8	Exponential	9	2.899
180	5	8	None	11	2.260	250	12	11	Exponential	6	2.918
181	10	9	Linear	7	2.270	251	13	11	Exponential	6	2.918
182	15	11	Exponential	8	2.281	252	6	9	Linear	10	2.948
183	5	13	None	7	2.302	253	11	8	Exponential	7	2.960
184	10	12	Exponential	9	2.305	254	5	11	None	7	2.980
185	11	12	Exponential	8	2.307	255	9	9	Exponential	14	2.985
186	9	8	Linear	9	2.312	256	7	11	None	6	2.996
187	7	8	None	6	2.312	257	7	7	Exponential	13	3.010
188	9	14	Linear	6	2.315	258	7	12	Exponential	9	3.019
189	9	11	None	6	2.328	259	14	9	Exponential	10	3.022
190	10	11	None	6	2.328	260	5	9	Linear	11	3.024
191	8	9	Linear	8	2.329	261	6	12	Exponential	9	3.036
192	9	11	Linear	5	2.339	262	10	7	Exponential	18	3.086
193	14	11	Exponential	7	2.341	263	6	13	Linear	6	3.090
194	10	11	Linear	5	2.342	264	6	6	Linear	10	3.092
195	11	11	Linear	5	2.342	265	15	5	Exponential	18	3.093
196	9	7	Linear	8	2.345	266	5	8	Linear	4	3.099
197	8	11	Linear	7	2.361	267	5	13	Linear	10	3.099
198	8	8	None	8	2.362	268	8	8	Exponential	8	3.100
199	9	6	Linear	9	2.363	269	9	8	Exponential	9	3.100
200	9	12	Exponential	10	2.365	270	9	7	Exponential	13	3.116
201	12	7	Exponential	19	2.397	271	13	9	Exponential	7	3.118
202	13	7	Exponential	19	2.397	272	6	11	None	6	3.143
203	7	7	None	9	2.401	273	6	11	Linear	7	3.144
204	14	7	Exponential	10	2.431	274	7	11	Linear	5	3.145
205	9	9	Linear	9	2.439	275	8	7	Exponential	15	3.148
206	13	7	Linear	9	2.440	276	8	5	None	10	3.152
207	10	9	Exponential	11	2.461	277	5	10	Linear	9	3.199
208	11	9	Exponential	11	2.461	278	14	6	Exponential	17	3.236
209	12	6	None	11	2.463	279	15	6	Exponential	16	3.236
210	5	10	None	9	2.496	280	14	5	Exponential	14	3.240

Table 24: Vienna results (3 of 3).

Rank	Search Distance	Grid Spacing	Cooling	Iterations	Fitness	Rank	Search Distance	Grid Spacing	Cooling	Iterations	Fitness
281	5	8	Exponential	15	3.304	323	6	9	Exponential	9	3.782
282	10	6	Exponential	13	3.307	324	6	7	Linear	10	3.874
283	14	14	Exponential	6	3.320	325	6	7	Exponential	16	3.904
284	15	14	Exponential	6	3.320	326	8	6	Exponential	8	3.908
285	11	6	Exponential	11	3.331	327	10	14	Exponential	7	3.912
286	6	14	None	6	3.338	328	11	14	Exponential	7	3.912
287	6	15	Exponential	11	3.349	329	6	14	Linear	7	3.943
288	11	14	Linear	8	3.352	330	9	15	Exponential	6	4.048
289	10	14	None	8	3.352	331	6	11	Exponential	6	4.061
290	5	5	None	18	3.353	332	7	11	Exponential	7	4.096
291	5	15	Linear	6	3.358	333	5	9	Exponential	8	4.101
292	6	8	Exponential	7	3.364	334	11	5	Exponential	23	4.198
293	5	13	Exponential	9	3.377	335	8	5	Linear	4	4.233
294	7	14	None	6	3.381	336	12	14	Linear	8	4.233
295	8	9	Exponential	8	3.384	337	13	14	Linear	8	4.233
296	5	14	None	10	3.408	338	14	14	Linear	8	4.233
297	8	14	Exponential	8	3.412	339	15	14	Linear	8	4.233
298	6	13	Exponential	8	3.415	340	11	14	None	8	4.233
299	12	6	Exponential	8	3.416	341	12	14	None	8	4.233
300	5	12	Exponential	9	3.418	342	13	14	None	8	4.233
301	7	8	Exponential	4	3.426	343	14	14	None	8	4.233
302	6	5	None	14	3.462	344	6	5	Linear	30	4.336
303	5	10	Exponential	9	3.469	345	10	5	Exponential	17	4.344
304	5	11	Linear	5	3.474	346	5	7	Linear	11	4.367
305	7	6	Linear	4	3.479	347	7	15	Exponential	5	4.456
306	5	7	None	7	3.479	348	5	11	Exponential	8	4.464
307	12	14	Exponential	7	3.492	349	8	5	Exponential	21	4.501
308	13	14	Exponential	7	3.492	350	7	6	Exponential	5	4.546
309	5	14	Linear	7	3.502	351	6	6	Exponential	4	4.583
310	8	11	Exponential	7	3.518	352	9	5	Exponential	20	4.642
311	6	7	None	13	3.549	353	5	5	Linear	12	4.699
312	12	5	Exponential	22	3.577	354	5	7	Exponential	29	4.787
313	13	5	Exponential	22	3.577	355	5	6	Linear	12	5.269
314	7	5	None	15	3.608	356	6	5	Exponential	19	5.369
315	7	9	Exponential	11	3.623	357	6	14	Exponential	6	5.423
316	9	6	Exponential	16	3.624	358	7	5	Linear	11	5.465
317	9	5	Linear	8	3.626	359	5	14	Exponential	7	5.582
318	5	15	Exponential	7	3.664	360	5	5	Exponential	5	5.658
319	5	6	None	13	3.674	361	7	14	Exponential	4	5.673
320	13	6	Exponential	9	3.704	362	7	5	Exponential	24	5.761
321	7	14	Linear	4	3.728	363	5	6	Exponential	3	6.694
322	8	15	Exponential	12	3.764						

Table 25: Washington results (1 of 3).

Rank	Search Distance	Grid Spacing	Cooling	Iterations	Fitness	Rank	Search Distance	Grid Spacing	Cooling	Iterations	Fitness
1	14	11	None	11	3.844	71	13	8	Linear	9	4.510
2	15	11	Linear	8	3.844	72	12	8	None	7	4.520
3	13	11	None	9	4.021	73	13	8	None	10	4.538
4	12	6	None	13	4.023	74	10	7	None	7	4.548
5	14	6	None	10	4.034	75	9	8	Linear	17	4.549
6	11	12	Linear	9	4.070	76	6	15	None	9	4.562
7	10	8	None	7	4.075	77	12	9	Exponential	12	4.575
8	11	8	None	8	4.084	78	13	9	Exponential	12	4.575
9	11	12	None	7	4.102	79	10	7	Linear	8	4.583
10	12	12	None	7	4.102	80	9	11	Linear	7	4.584
11	12	12	Linear	7	4.102	81	11	14	Linear	7	4.586
12	15	10	Linear	8	4.116	82	14	11	Exponential	14	4.592
13	14	8	None	10	4.122	83	7	11	Linear	10	4.623
14	15	8	Linear	10	4.122	84	15	11	Exponential	9	4.623
15	13	12	None	10	4.132	85	10	14	None	6	4.634
16	11	10	None	8	4.133	86	11	14	None	6	4.634
17	12	10	None	8	4.144	87	12	14	None	6	4.634
18	13	10	None	8	4.144	88	13	14	None	6	4.634
19	13	10	Linear	8	4.144	89	14	14	None	6	4.634
20	14	10	Linear	8	4.144	90	15	14	None	6	4.634
21	15	8	None	10	4.146	91	12	14	Linear	6	4.634
22	14	12	None	8	4.162	92	13	14	Linear	6	4.634
23	14	7	Linear	9	4.163	93	14	14	Linear	6	4.634
24	14	10	None	7	4.168	94	15	14	Linear	6	4.634
25	15	10	None	7	4.168	95	11	6	None	17	4.635
26	14	11	Linear	11	4.175	96	8	7	None	11	4.636
27	13	12	Linear	9	4.175	97	7	7	None	12	4.637
28	12	9	None	6	4.190	98	8	14	None	6	4.639
29	13	9	None	6	4.190	99	14	8	Linear	6	4.641
30	14	9	None	6	4.190	100	6	7	None	12	4.642
31	15	9	None	6	4.190	101	9	7	None	9	4.643
32	13	9	Linear	6	4.190	102	5	7	None	17	4.644
33	14	9	Linear	6	4.190	103	9	14	Linear	7	4.655
34	15	9	Linear	6	4.190	104	13	7	Linear	7	4.658
35	11	11	None	10	4.196	105	7	8	None	9	4.661
36	15	6	Linear	8	4.200	106	9	8	None	8	4.662
37	12	7	None	10	4.206	107	5	10	None	9	4.662
38	15	6	None	11	4.218	108	12	7	Linear	7	4.665
39	10	12	None	9	4.240	109	7	15	None	7	4.667
40	12	9	Linear	8	4.249	110	9	15	None	7	4.667
41	14	12	Linear	9	4.258	111	9	15	Linear	7	4.667
42	15	12	Linear	9	4.258	112	10	15	Linear	7	4.667
43	11	9	None	7	4.269	113	7	14	None	7	4.672
44	12	11	None	7	4.270	114	12	11	Exponential	8	4.686
45	6	11	None	12	4.278	115	13	11	Exponential	8	4.686
46	11	7	None	11	4.289	116	6	10	None	7	4.689
47	13	6	None	9	4.295	117	7	10	None	7	4.689
48	12	10	Linear	7	4.297	118	8	10	Linear	7	4.689
49	11	8	Linear	9	4.317	119	8	15	None	7	4.691
50	10	11	Linear	8	4.327	120	9	14	None	5	4.692
51	14	7	None	9	4.327	121	10	14	Linear	5	4.692
52	15	7	None	9	4.327	122	6	12	None	9	4.699
53	15	7	Linear	9	4.327	123	8	7	Linear	9	4.706
54	9	11	None	7	4.334	124	5	10	Linear	22	4.708
55	10	11	None	7	4.336	125	13	15	None	8	4.712
56	12	11	Linear	7	4.336	126	14	15	None	8	4.712
57	13	11	Linear	8	4.337	127	15	15	None	8	4.712
58	15	12	None	7	4.341	128	14	7	Exponential	13	4.714
59	7	11	None	7	4.346	129	15	7	Exponential	11	4.714
60	12	8	Linear	11	4.348	130	10	15	None	7	4.722
61	10	10	None	7	4.354	131	11	15	None	7	4.722
62	11	11	Linear	8	4.355	132	12	15	None	7	4.722
63	15	11	None	6	4.359	133	11	15	Linear	7	4.722
64	13	7	None	8	4.401	134	12	15	Linear	7	4.722
65	8	10	None	9	4.415	135	13	15	Linear	7	4.722
66	15	9	Exponential	6	4.417	136	13	6	Linear	10	4.724
67	11	7	Linear	10	4.447	137	8	11	Linear	6	4.730
68	8	11	None	7	4.456	138	8	6	None	10	4.739
69	14	6	Linear	8	4.498	139	14	15	Linear	7	4.742
70	14	9	Exponential	8	4.500	140	15	15	Linear	7	4.742

Table 26: Washington results (2 of 3).

Rank	Search Distance	Grid Spacing	Cooling	Iterations	Fitness	Rank	Search Distance	Grid Spacing	Cooling	Iterations	Fitness
141	14	5	Linear	9	4.753	211	13	12	Exponential	6	5.210
142	7	10	Linear	9	4.755	212	12	12	Exponential	8	5.216
143	9	12	None	7	4.756	213	7	13	None	8	5.220
144	12	10	Exponential	12	4.759	214	14	13	None	8	5.226
145	13	10	Exponential	12	4.759	215	15	13	None	7	5.226
146	9	10	None	6	4.759	216	10	10	Exponential	8	5.227
147	9	10	Linear	6	4.759	217	11	10	Exponential	8	5.227
148	10	10	Linear	6	4.759	218	6	9	Linear	4	5.229
149	11	10	Linear	6	4.759	219	13	5	Linear	9	5.243
150	14	10	Exponential	6	4.759	220	8	5	None	13	5.244
151	15	10	Exponential	6	4.759	221	6	6	None	15	5.256
152	9	7	Linear	9	4.794	222	12	8	Exponential	7	5.265
153	7	7	Linear	15	4.803	223	13	8	Exponential	7	5.265
154	7	8	Linear	14	4.813	224	10	9	Exponential	4	5.265
155	6	10	Linear	6	4.821	225	11	9	Exponential	4	5.265
156	5	15	None	8	4.827	226	6	15	Linear	9	5.281
157	10	8	Linear	11	4.833	227	14	5	None	9	5.285
158	6	14	None	9	4.842	228	11	5	None	10	5.302
159	14	8	Exponential	21	4.856	229	9	6	Linear	9	5.322
160	8	14	Linear	7	4.867	230	14	12	Exponential	6	5.336
161	15	8	Exponential	12	4.869	231	15	12	Exponential	6	5.336
162	10	9	None	3	4.887	232	11	5	Linear	10	5.369
163	11	9	Linear	3	4.887	233	8	14	Exponential	5	5.380
164	12	7	Exponential	8	4.903	234	9	14	Exponential	5	5.380
165	13	7	Exponential	8	4.903	235	10	14	Exponential	5	5.380
166	15	14	Exponential	7	4.903	236	11	14	Exponential	5	5.380
167	14	14	Exponential	7	4.911	237	9	12	Linear	7	5.394
168	9	6	None	9	4.914	238	9	10	Exponential	6	5.398
169	10	6	None	9	4.924	239	14	13	Linear	5	5.410
170	13	5	None	10	4.925	240	15	15	Exponential	6	5.414
171	9	9	None	3	4.927	241	5	9	None	4	5.418
172	10	9	Linear	3	4.927	242	5	14	Linear	6	5.423
173	11	13	None	9	4.938	243	12	15	Exponential	7	5.437
174	12	13	None	9	4.938	244	13	15	Exponential	7	5.437
175	12	13	Linear	9	4.938	245	10	8	Exponential	16	5.455
176	13	13	Linear	9	4.938	246	15	13	Linear	5	5.462
177	12	6	Linear	9	4.943	247	12	5	Linear	11	5.471
178	8	9	None	3	4.973	248	7	12	None	5	5.474
179	7	9	None	3	4.975	249	7	6	None	10	5.486
180	9	9	Linear	3	4.975	250	12	13	Exponential	8	5.487
181	5	12	None	14	4.976	251	13	13	Exponential	8	5.487
182	5	8	None	11	4.983	252	6	11	Exponential	5	5.496
183	7	15	Linear	8	4.984	253	7	11	Exponential	5	5.496
184	10	12	Linear	8	4.996	254	6	13	Linear	11	5.496
185	6	9	None	3	5.000	255	11	6	Linear	8	5.504
186	7	9	Linear	3	5.000	256	6	8	Linear	7	5.505
187	8	9	Linear	3	5.000	257	5	8	Linear	21	5.507
188	12	14	Exponential	6	5.010	258	6	13	None	7	5.522
189	13	14	Exponential	6	5.010	259	7	13	Linear	7	5.522
190	7	12	Linear	6	5.013	260	8	13	Linear	7	5.522
191	9	5	None	10	5.067	261	15	13	Exponential	7	5.522
192	10	13	None	7	5.090	262	8	15	Exponential	8	5.525
193	11	13	Linear	7	5.090	263	10	13	Exponential	8	5.533
194	8	15	Linear	7	5.097	264	11	13	Exponential	8	5.533
195	6	11	Linear	8	5.104	265	5	11	Exponential	6	5.536
196	8	12	None	7	5.122	266	7	14	Linear	6	5.541
197	8	12	Linear	9	5.127	267	14	13	Exponential	6	5.542
198	10	11	Exponential	7	5.137	268	10	15	Exponential	6	5.543
199	11	11	Exponential	7	5.137	269	11	15	Exponential	6	5.543
200	10	6	Linear	9	5.145	270	5	15	Linear	7	5.553
201	8	8	None	5	5.157	271	9	15	Exponential	7	5.553
202	8	8	Linear	5	5.157	272	5	6	None	11	5.556
203	6	8	None	4	5.166	273	6	14	Exponential	7	5.559
204	8	6	Linear	9	5.166	274	7	14	Exponential	7	5.559
205	5	11	None	7	5.170	275	10	13	Linear	7	5.595
206	8	11	Exponential	13	5.177	276	5	12	Linear	12	5.618
207	13	13	None	8	5.183	277	9	13	Linear	7	5.620
208	5	11	Linear	9	5.186	278	8	13	None	7	5.625
209	9	11	Exponential	9	5.186	279	8	10	Exponential	10	5.629
210	5	14	None	6	5.202	280	6	12	Linear	9	5.638

Table 27: Washington results (3 of 3).

Rank	Search Distance	Grid Spacing	Cooling	Iterations	Fitness	Rank	Search Distance	Grid Spacing	Cooling	Iterations	Fitness
281	9	13	None	5	5.652	323	14	6	Exponential	3	6.264
282	6	10	Exponential	13	5.653	324	10	6	Exponential	9	6.266
283	7	10	Exponential	13	5.653	325	11	6	Exponential	9	6.270
284	5	9	Linear	13	5.660	326	6	9	Exponential	3	6.272
285	6	15	Exponential	10	5.675	327	7	9	Exponential	3	6.272
286	7	15	Exponential	10	5.675	328	6	6	Linear	7	6.299
287	6	14	Linear	7	5.676	329	5	6	Linear	11	6.314
288	6	5	None	17	5.680	330	15	5	None	9	6.333
289	8	9	Exponential	9	5.688	331	7	5	Linear	16	6.343
290	9	9	Exponential	7	5.688	332	10	12	Exponential	11	6.343
291	6	7	Linear	11	5.692	333	13	5	Exponential	11	6.376
292	7	6	Linear	8	5.717	334	5	6	Exponential	25	6.417
293	11	8	Exponential	10	5.719	335	6	5	Linear	22	6.421
294	9	8	Exponential	13	5.732	336	6	6	Exponential	18	6.509
295	9	7	Exponential	13	5.758	337	5	9	Exponential	3	6.568
296	7	5	None	13	5.781	338	12	5	Exponential	11	6.630
297	11	12	Exponential	6	5.801	339	5	7	Exponential	19	6.645
298	15	5	Linear	7	5.804	340	9	6	Exponential	11	6.649
299	10	7	Exponential	7	5.829	341	5	10	Exponential	3	6.661
300	11	7	Exponential	7	5.829	342	8	5	Exponential	19	6.692
301	7	7	Exponential	8	5.842	343	9	5	Exponential	31	6.716
302	12	5	None	5	5.894	344	5	7	Linear	4	6.724
303	6	12	Exponential	6	5.930	345	5	5	Linear	24	6.772
304	14	15	Exponential	6	5.954	346	15	5	Exponential	4	6.797
305	9	5	Linear	14	5.955	347	5	15	Exponential	6	6.862
306	8	8	Exponential	11	5.971	348	8	12	Exponential	3	6.893
307	9	12	Exponential	6	6.001	349	7	12	Exponential	9	6.897
308	5	12	Exponential	6	6.012	350	11	5	Exponential	11	6.935
309	5	13	None	6	6.033	351	8	6	Exponential	3	6.981
310	6	8	Exponential	16	6.050	352	7	6	Exponential	3	6.986
311	5	8	Exponential	18	6.061	353	14	5	Exponential	3	6.991
312	7	8	Exponential	9	6.088	354	10	5	Exponential	11	6.993
313	10	5	Linear	11	6.150	355	7	5	Exponential	17	7.071
314	5	5	None	17	6.155	356	5	13	Linear	6	7.077
315	10	5	None	9	6.181	357	6	5	Exponential	16	7.153
316	12	6	Exponential	8	6.181	358	8	13	Exponential	6	7.349
317	13	6	Exponential	8	6.181	359	9	13	Exponential	6	7.349
318	8	5	Linear	9	6.211	360	5	13	Exponential	6	7.499
319	8	7	Exponential	4	6.216	361	6	13	Exponential	7	7.520
320	6	7	Exponential	9	6.222	362	7	13	Exponential	7	7.520
321	5	14	Exponential	3	6.226	363	5	5	Exponential	14	7.547
322	15	6	Exponential	3	6.262						

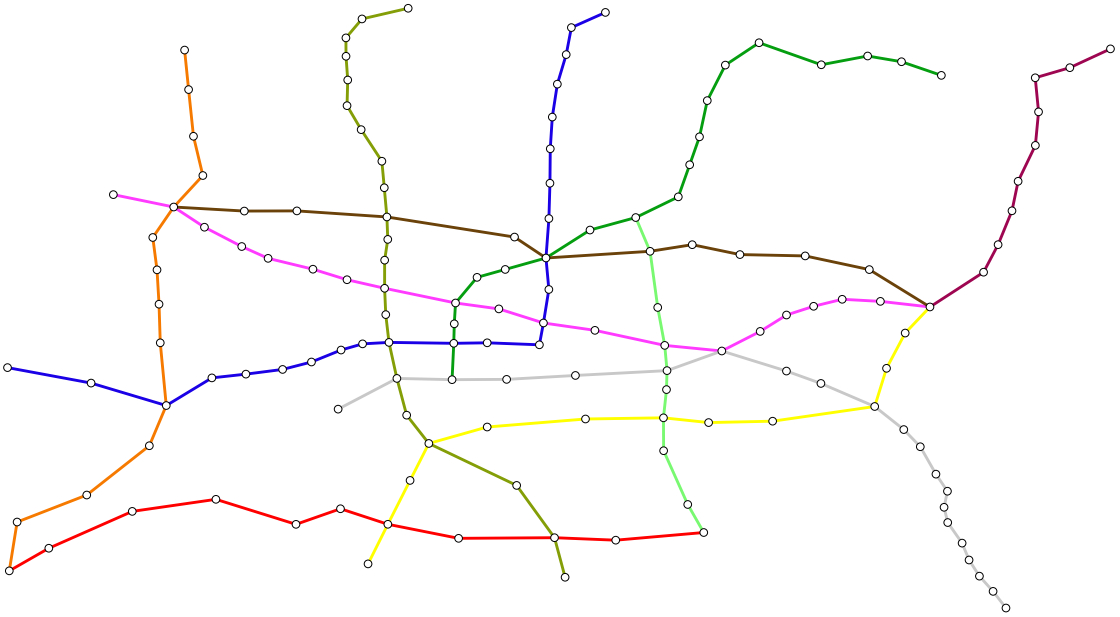


Figure 121: Mexico – Geographic.

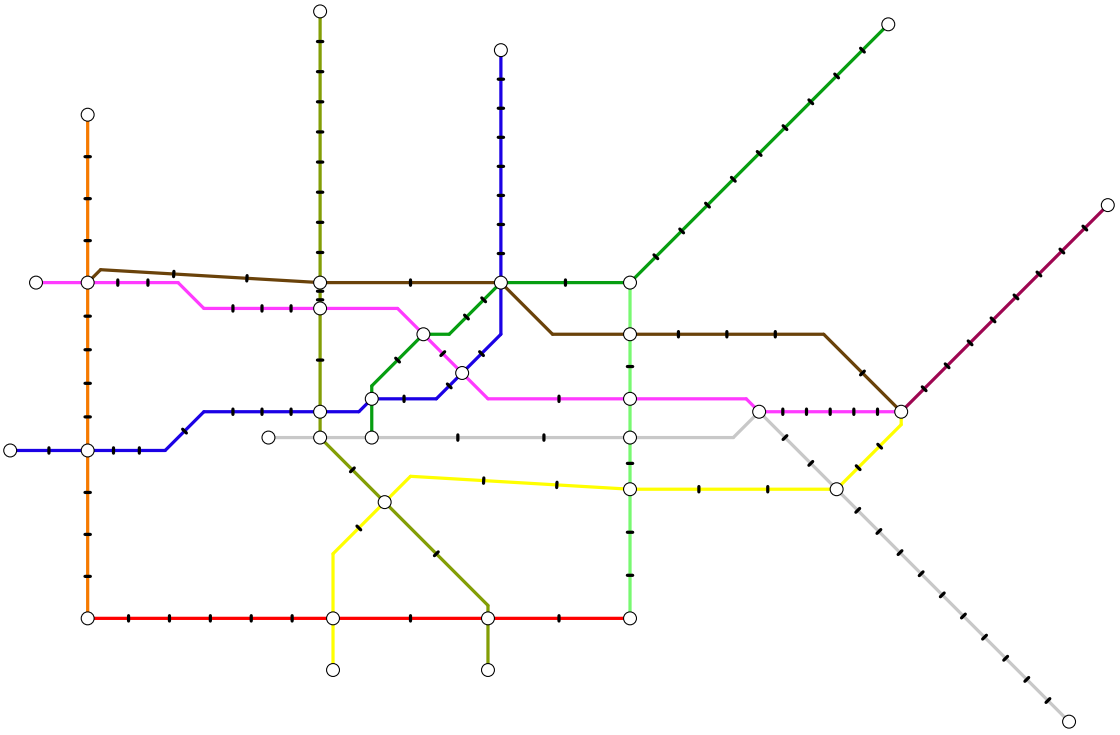


Figure 122: Mexico – Rank 1 (Fitness = 2.653).

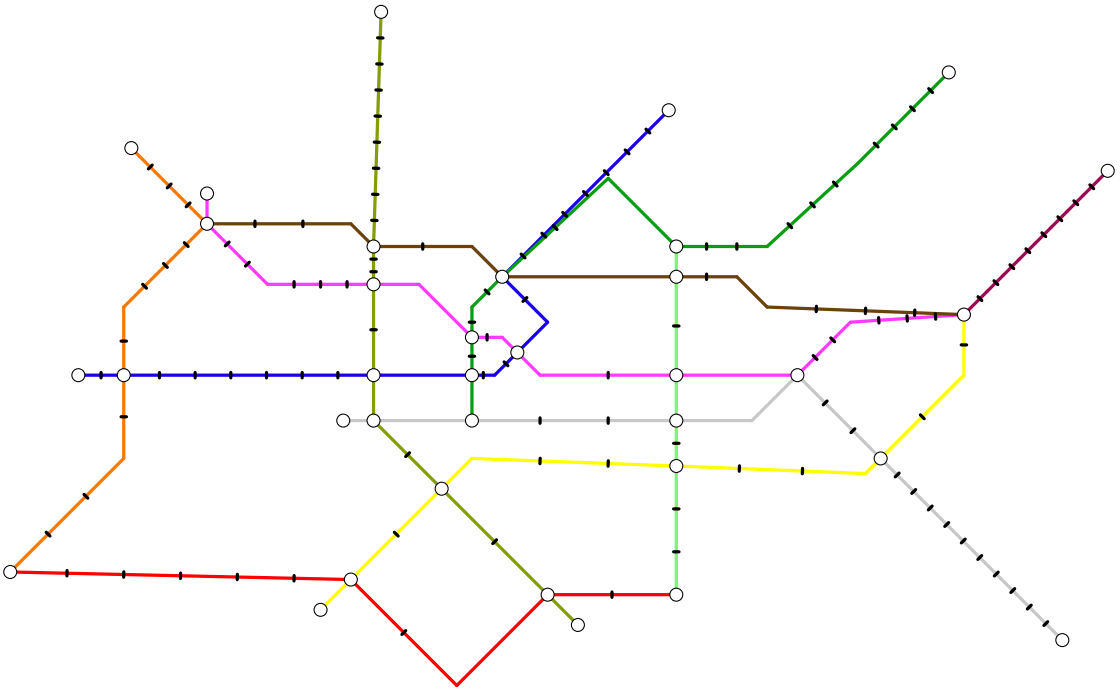


Figure 123: Mexico – Rank 181 (Fitness = 4.379)

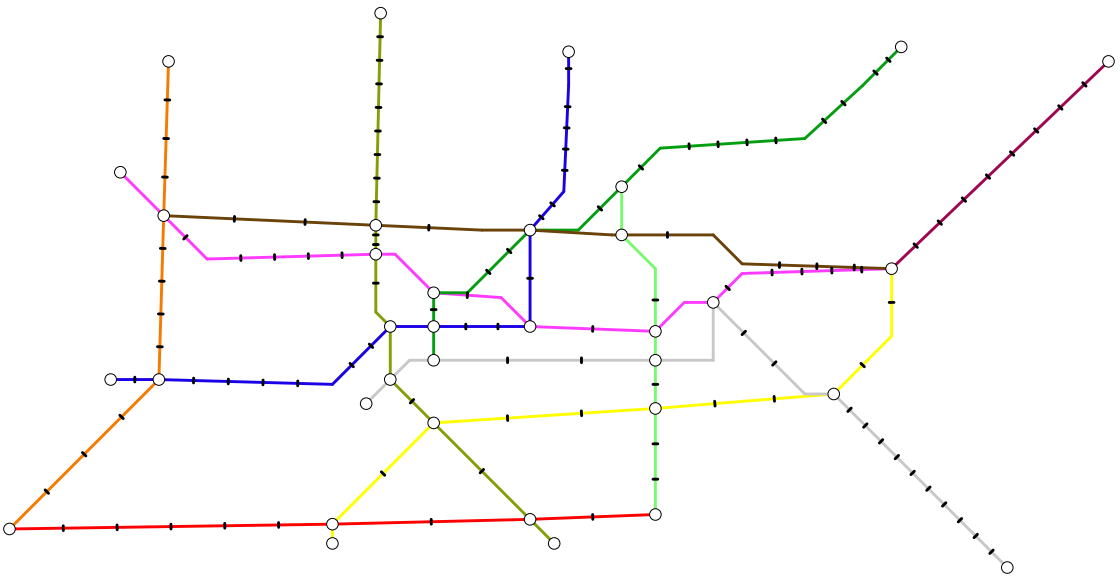


Figure 124: Mexico – Rank 363 (Fitness = 8.712)

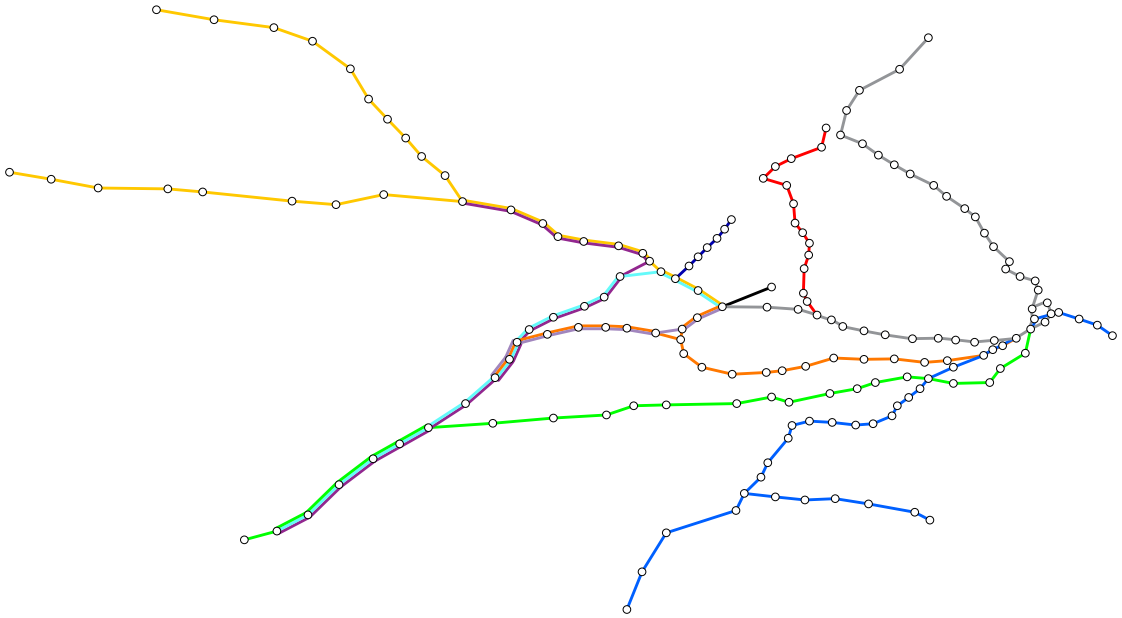


Figure 125: Sydney – Geographic.

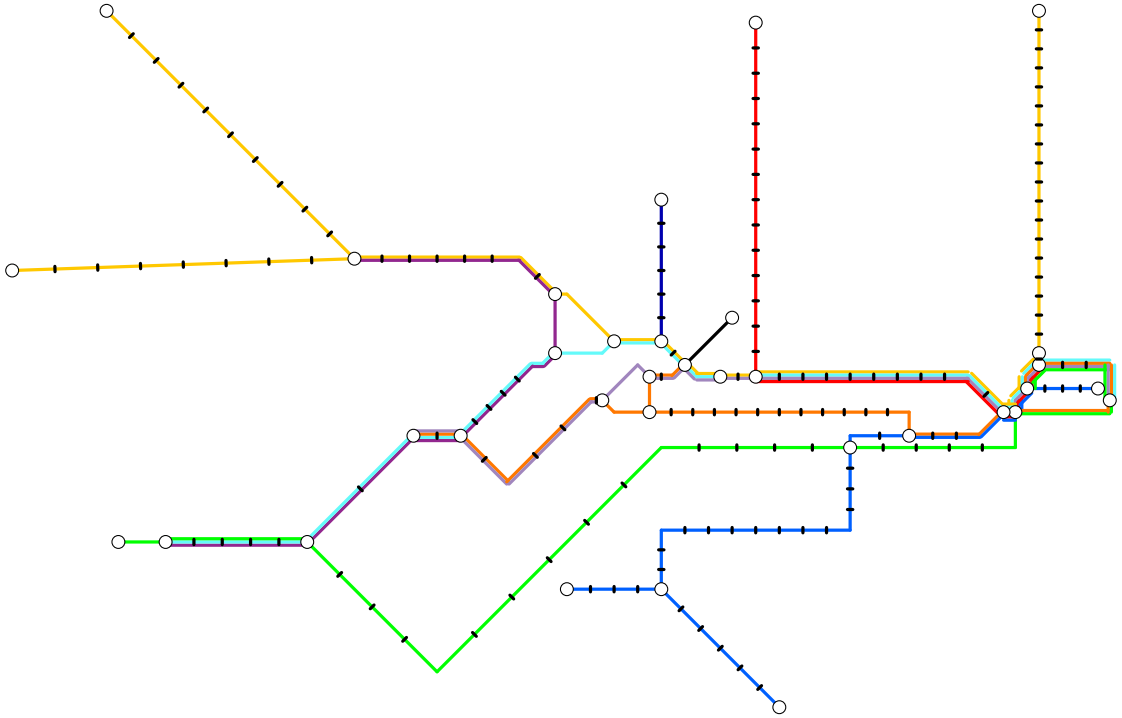


Figure 126: Sydney – Rank 1 (Fitness = 19.359).

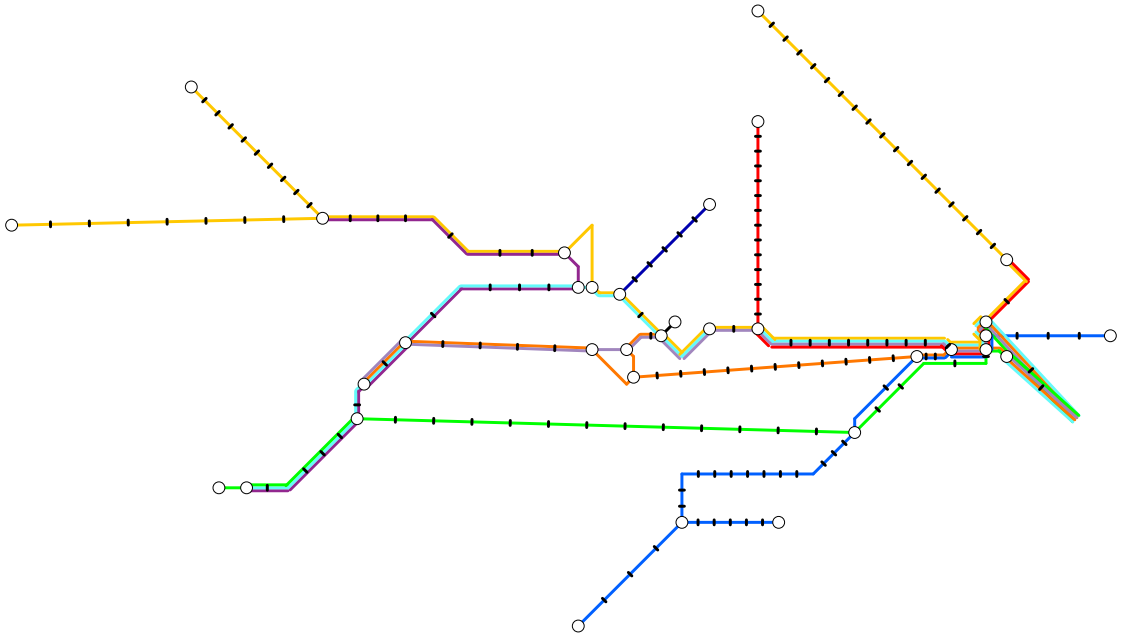


Figure 127: Sydney – Rank 181 (Fitness = 21.944).

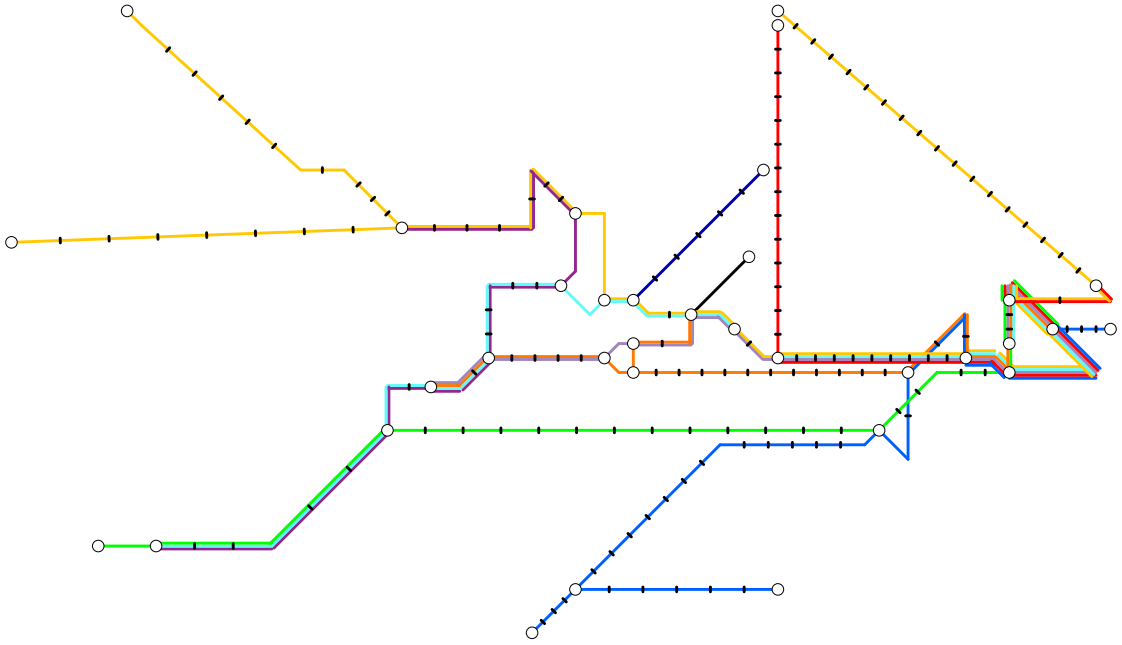


Figure 128: Sydney – Rank 363 (Fitness = 26.964).

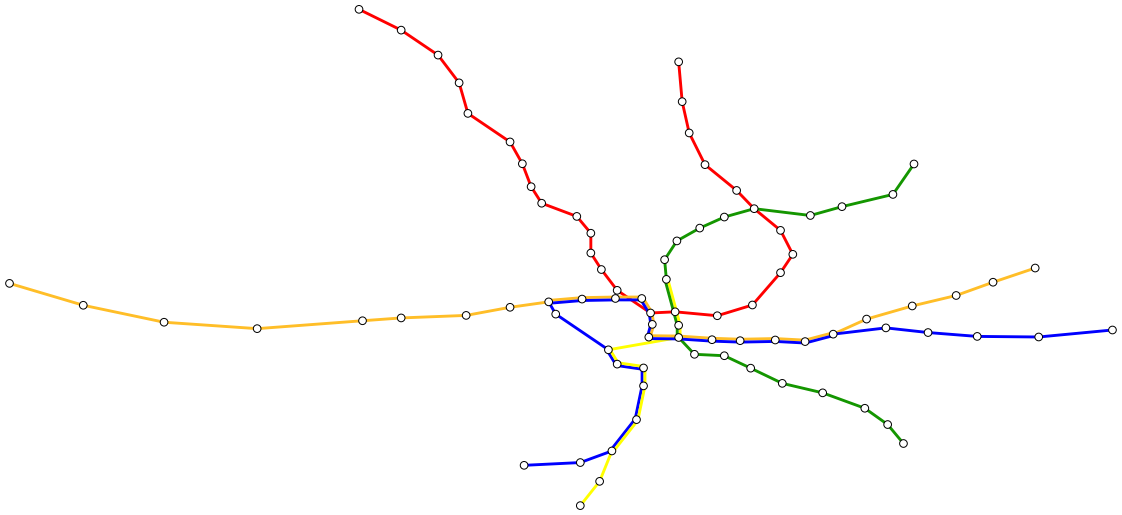


Figure 129: Washington – Geographic.

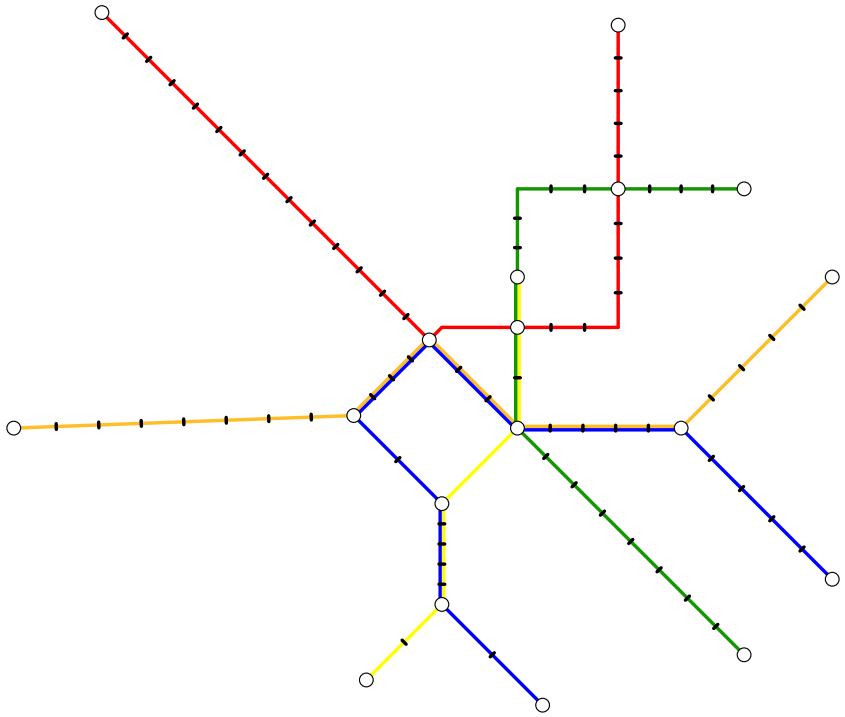


Figure 130: Washington – Rank 1 (Fitness = 3.844).

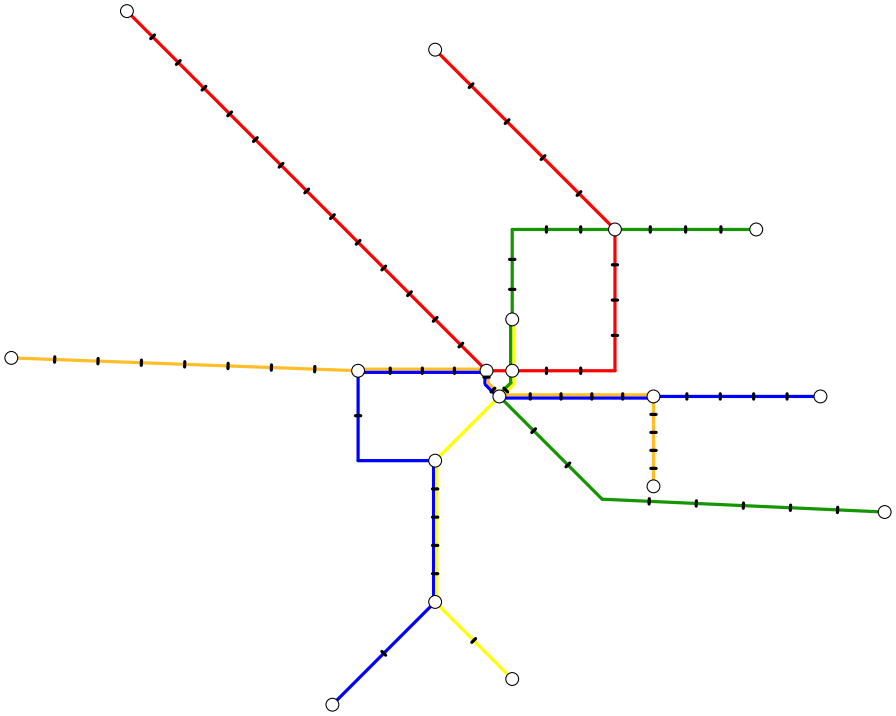


Figure 131: Washington – Rank 181 (Fitness = 4.976).

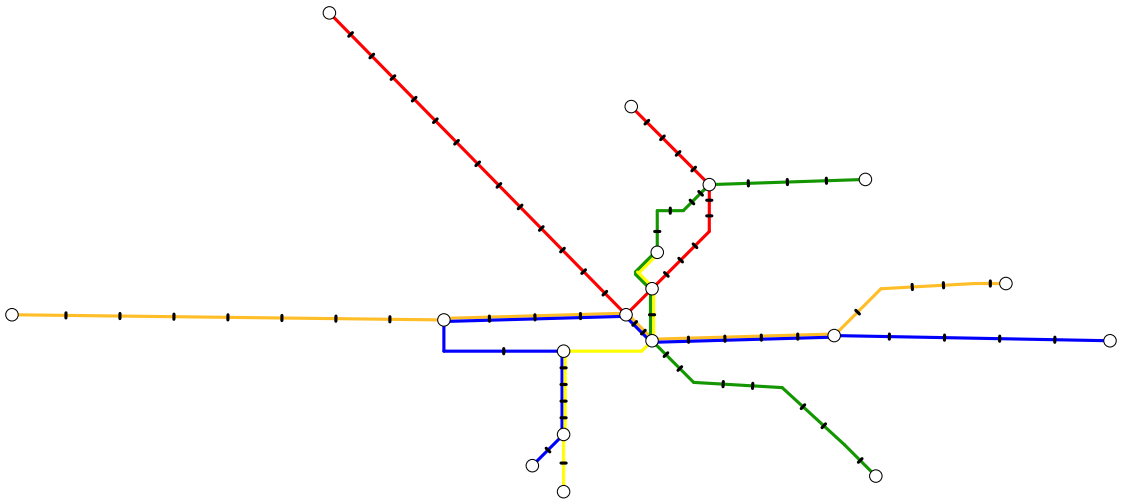


Figure 132: Washington – Rank 363 (Fitness = 7.547).

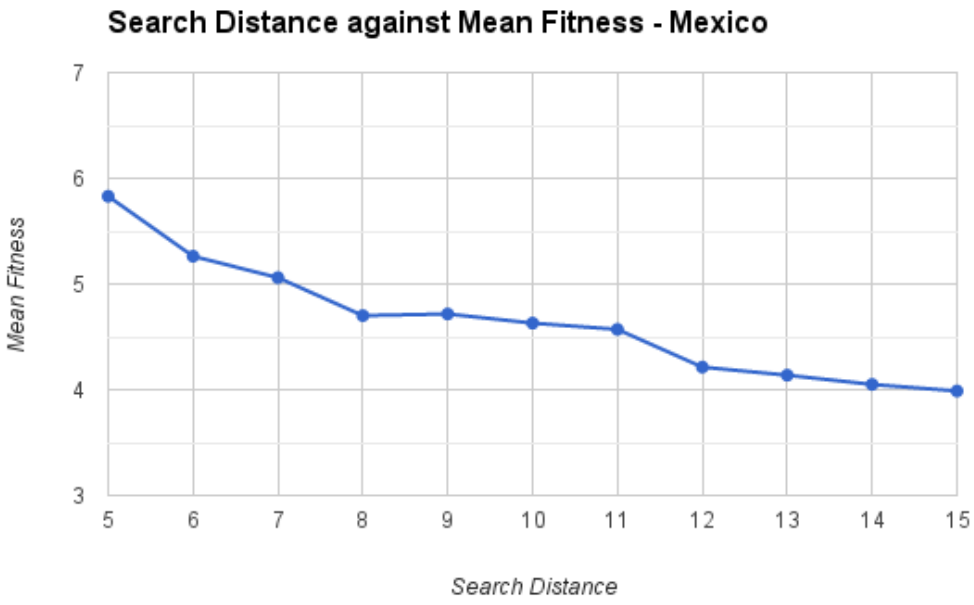


Figure 133: Search distance against mean fitness – Mexico.

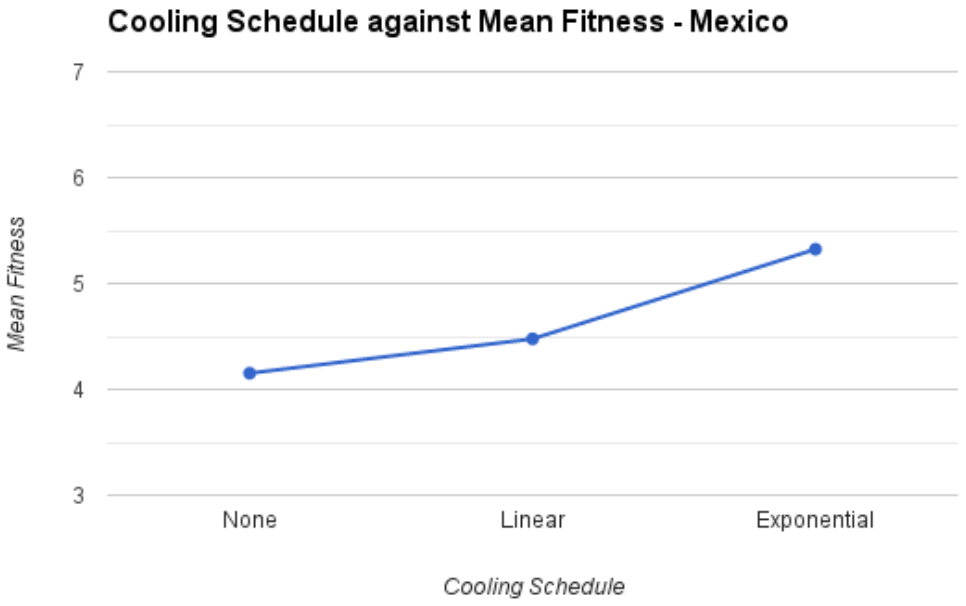


Figure 134: Cooling schedule against mean fitness – Mexico.

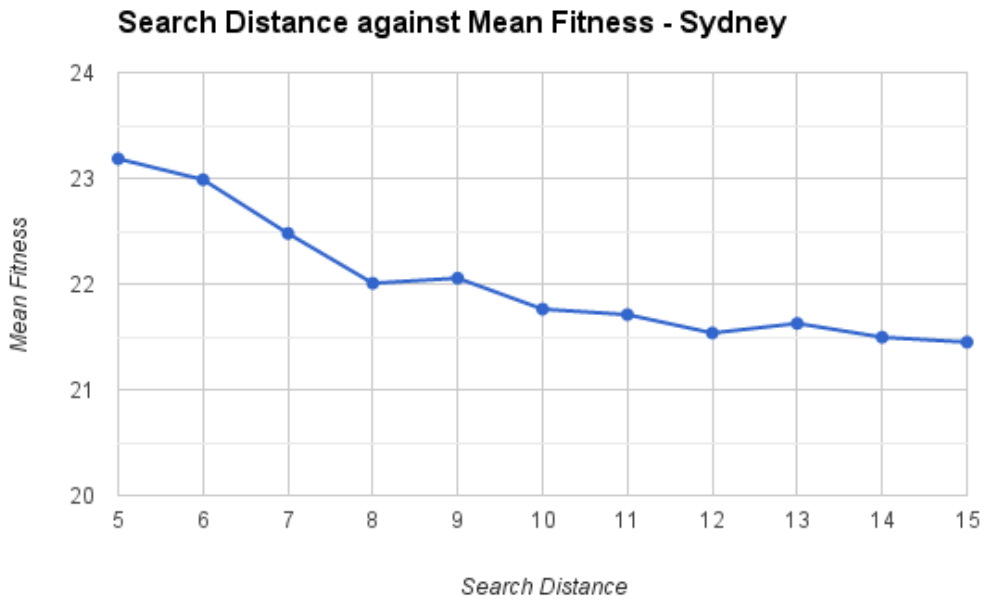


Figure 135: Search distance against mean fitness – Sydney.

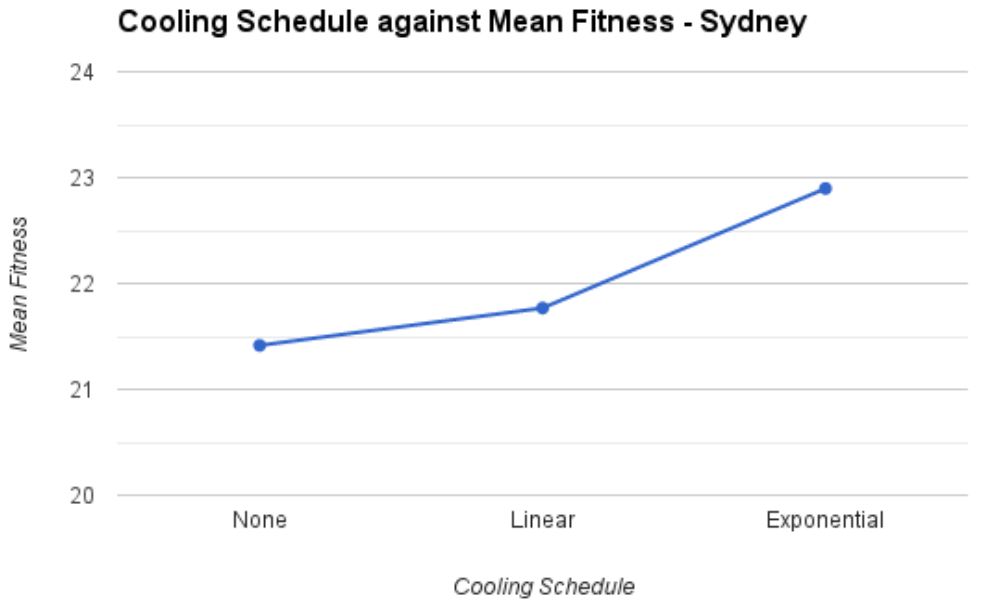


Figure 136: Cooling schedule against mean fitness – Sydney.

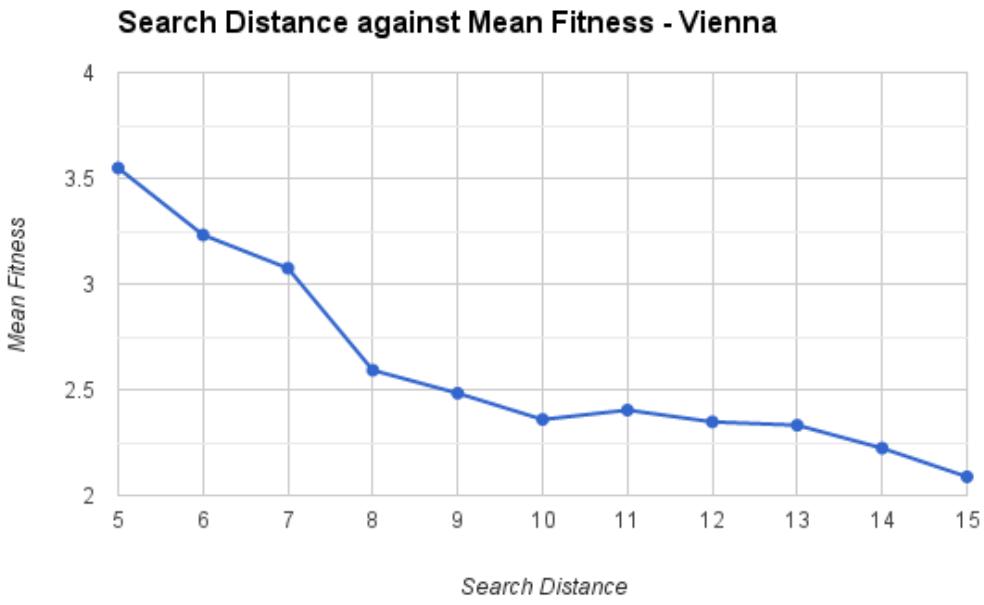


Figure 137: Search distance against mean fitness – Vienna.

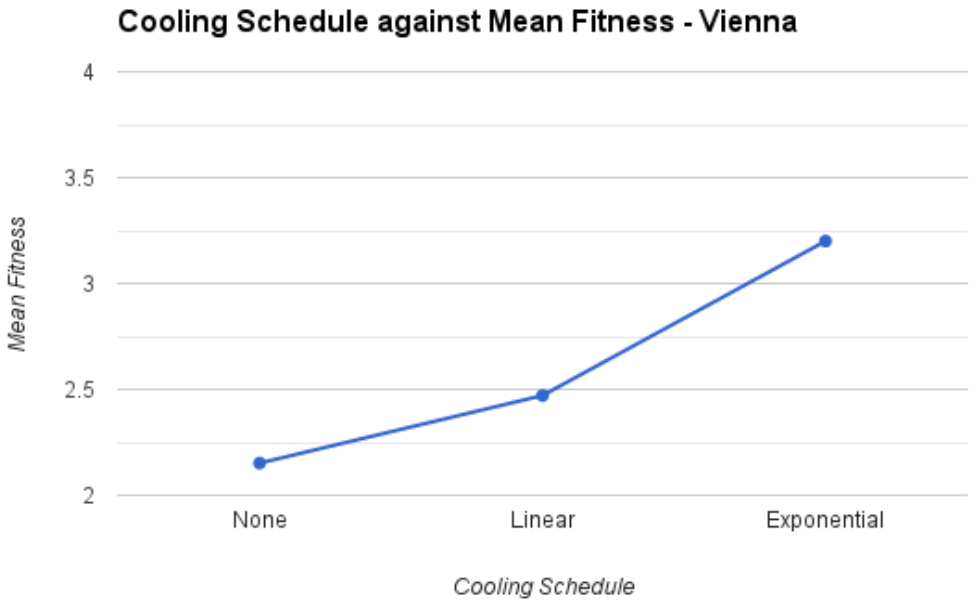


Figure 138: Cooling schedule against mean fitness – Vienna.

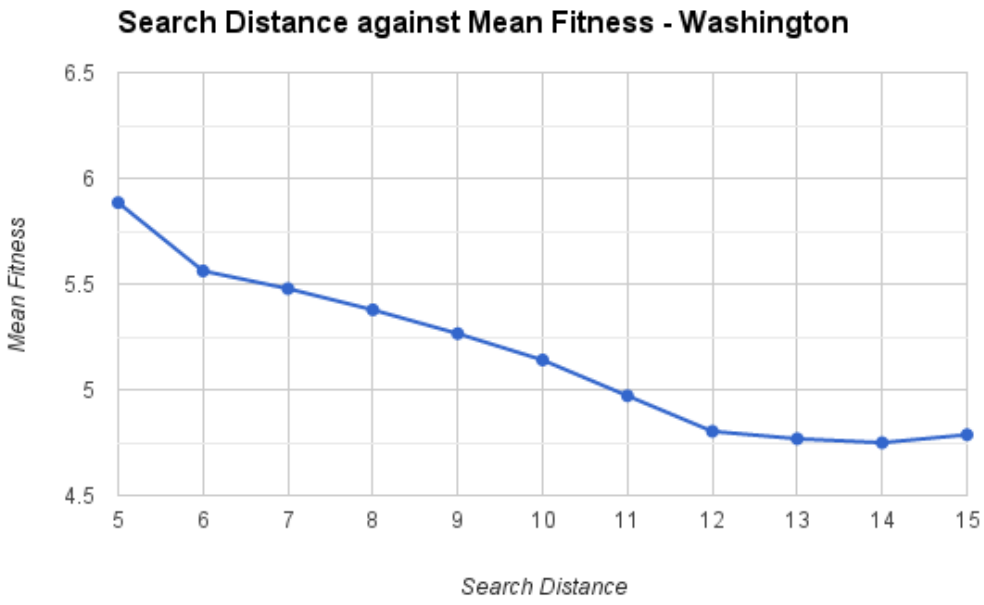


Figure 139: Search distance against mean fitness – Washington.

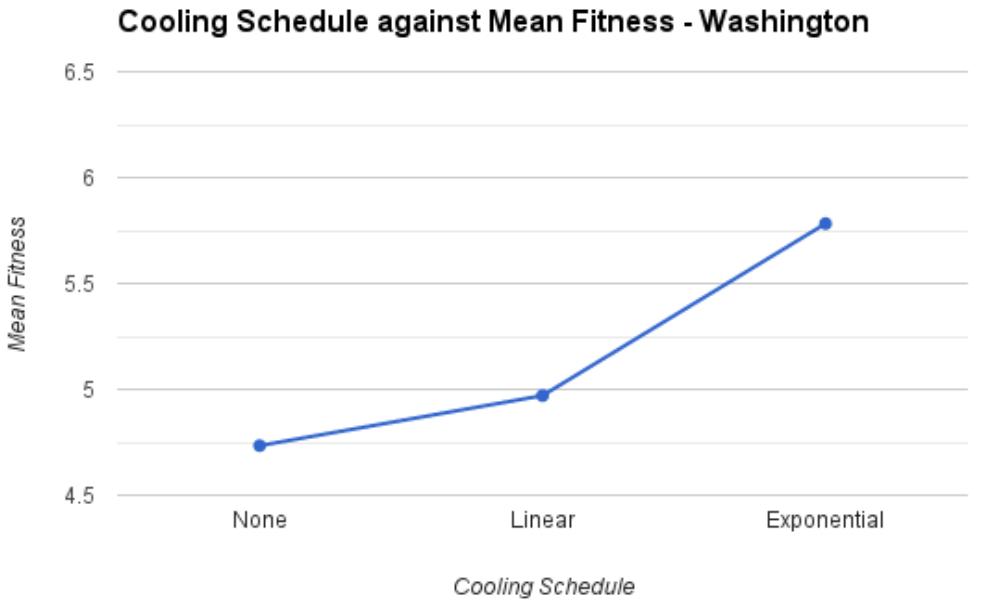


Figure 140: Cooling schedule against mean fitness – Washington.