Metro Map Layout Using Multicriteria Optimization

Jonathan M. Stott, Peter Rodgers
University of Kent, UK
{jms30@kent.ac.uk, P.J.Rodgers@kent.ac.uk}

Abstract

We describe a system to automatically generate metro maps using a multicriteria approach. We have implemented a hill climbing optimizer which uses a fitness score generated from a sum of several aesthetic metrics. This is used to move from the initial geographic layout of the map to a schematic layout that is intended to aid travellers' navigation. We describe the software and show its application to a number of real world metro maps.

Keywords--- **metro map layout problem, public transport schematics, multicriteria optimization, graph drawing**.

1. Introduction

Metro maps, or public transport schematics, are familiar to most people. Many cities have underground metros or above ground tram networks which are usually represented by a schematic map. The map simplifies the true geographic layout of the network by straightening lines and evenly distributing stations along the lines [8]. These maps take inspiration from what is considered the first such schematic, developed by Harry Beck for the London Underground [3]. Travellers find it easier to use such a simplified map to plan routes between start and destination stations.

However, these schematics are typically created by hand taking a large amount of effort. The goal of our research project is to address the metro map layout problem, where we attempt to automatically generate a schematic for a metro map.

Such maps are not only used in public transport. Closely related schematics can be found in circuit diagrams (from which Harry Beck drew his inspiration) and public utility pipeline diagrams. The metro map metaphor has also been considered to be a useful visualization technique for non-spatial data [10].

We have approached the problem by using a hill climbing multicriteria optimization technique. Multicriteria systems have been seen before in other graph layout applications [2] and a number of metrics have been investigated [1][6]. In our case we represent a metro map as a graph, with stations represented by nodes and lines between stations represented by edges, including multiple edges where there is more than one line between stations. We start with the geographic layout of the network. A number of aesthetic metrics are calculated in the graph and summed to produce a total fitness value. The nodes are tested in turn to see if a position can be found that improves the score. This process continues for a specified number of iterations.

The implemented metrics are: edge crossings; edge length, which attempts to position nodes evenly; angular resolution, which attempts to avoid very narrow angles for edges attached to the same node; line straightness, which attempts to avoid a change in direction when a metro line goes through a node; and 4-gonal, which tries to make lines horizontal, vertical or 45° diagonal.

In addition to the basic multicriteria system, we have implemented other features to improve the final layout including a simple labelling strategy that tests a number of possible label orientations. We also implemented a useful technique to contract long lines. Here we replace a chain of degree two nodes with one weighted edge. The layout is then generated, after which the edge is expanded, so that the nodes in the contracted line reappear. This avoids the computational expense and difficulties inherent in optimising a long line in the map. A further technique deals with over-length edges. Here, two clusters of stations are separated by a edge that is too long. It is difficult to deal with this problem directly as an aesthetic criteria, so tests for such edges are made periodically during the optimising process. When such edges are found, the clusters are moved together. We also experimented with a restriction on the movement of nodes to maintain some physical relationship between neighbours, so that for example if one node is north of another, it remains north in the final layout. We do not use this restriction in the examples in the paper, because the degree of movement allowed in the current implementation is too small, and because in any case, without this restriction in place the nodes do not actually wander very far from the desired relationship to each other.

There is some closely related work, one paper describes a force directed attempt to solve the closely

related problem of laying out schematic cable plans [7]. We know of one previous attempt at automatically laying out public transport schematics by Hong et. al. [4][5]. Their work introduces the "Metro Map Layout Problem" and describes an implemented solution to it. They take a force directed approach, and use a very similar graph model to the one presented in this paper, and as with our work they have a contraction preprocessing step. Rather than begin with the geographic position of stations as a starting layout, their work begins with a random layout. They consider similar aesthetics to those we have chosen, although we have not produced metrics for dealing with overlapping labels or methods for drawing lines with unique colours. We have additional metrics: the edge length and angular resolution metrics are not explicitly present in their work. We have also experimented with enforcing a geographic relationship rule, not present in their work.

Our approach is relatively slow compared to force directed methods. However, there are significant benefits. The requirements for the layout can be specified directly, and adjusted to user preference by altering the relative weights of each metric. Moreover, further metrics can easily be implemented and to deal with specific difficulties in the layout or user requirements. For instance, a metric could be implemented to ensure that stations appear in suitable positions relative to other geographic features, such as rivers or roads.

The remainder of this paper is organized as follows: Section 2 gives a detailed description of our method applied to real metro maps; Section 3 shows some examples; and Section 4 gives our conclusions, discusses some problems with the method and outlines further work.

2. Method

A metro map can be represented as a graph where stations are nodes and lines between stations are edges. As some metro maps feature multiple lines between two stations, we have to take into account multiple edges between nodes. We use the term 'line' to talk about the subset of edges and nodes that form a line in the map and are normally distinguished by colour (for example, the Central Line on the London Underground map).

We embed the graph on an integer grid. This minimizes the number of points to consider when moving nodes and therefore reduces the overall running time of the algorithm. It also encourages edges to be more orthogonal.

The method that we have developed involves using a multicriteria optimization technique with a hill climbing approach. A number of metrics are calculated in order to determine an aesthetic quality for the graph. Nodes are moved such that the total metric value never increases. A preprocessing step is also included which simplifies the graph by contracting nodes of degree two. A software tool is used to visualize and manipulate the graphs and algorithm parameters.

2.1. Schematics Software Tool

In order to experiment with various metrics and their settings, we decided to implement our own software tool in Java. The tool (Figure 1) consists of a graphical interface where nodes and edges can be created and manipulated. Whole graphs can be saved in text files so that the tool can display graphs that have been worked on previously.

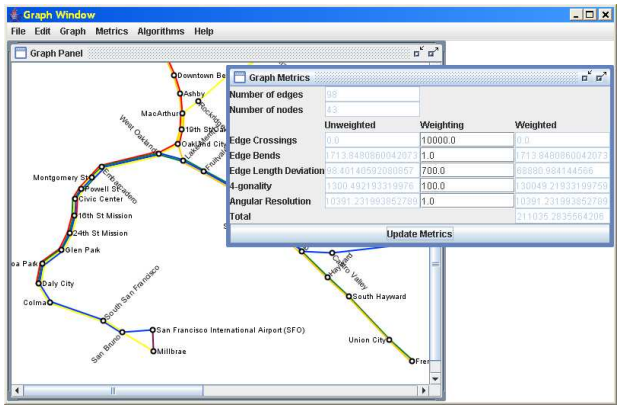

Figure 1. Screenshot of the schematics software tool

2.2. Preprocessing – Graph Contraction

Metro maps tend to have a certain characteristic of long lines of stations radiating from a central area. These lines can usually be drawn as a single straight line. Replacing them by a single edge means the optimizer does not have to attempt to generate the straight line iteratively (which does not always occur) and means that the process has improved performance, because the number of nodes that are moved during each iteration is reduced.

The preprocessing step involves contracting the graph such that all nodes of degree two are removed from the graph and replaced by a single weighted edge. The removed nodes and edges are not considered during calculation of the aesthetic metrics. At the end of the algorithm, the contracted edges are expanded and nodes are replaced at regular intervals between the endpoints of the edge.

However, it might not be desirable in every case to contract the graph in this manner. Once an edge is contracted it merely becomes a single straight edge, losing any bends between nodes in between the end points of the contracted edge.

2.3. The Hill Climbing Optimizer

The hill climber is an iterative process. During each iteration, an attempt is made to move each node in the graph. Any nodes which satisfy the conditions for movement are moved to their new positions.

There are various ways in which moves can be made when optimizing a graph. We have experimented with random movement of nodes, however it can take a considerable time for the random positioning to find the

ideal location for a particular node. Instead, we constrain nodes to the grid and calculate the metrics for each possible location in a square around the start point of the node. The size of this square area is specified at the start of the algorithm. For example, specifying a size as one will allow the node to move to any of the eight immediately adjacent grid points. The node is moved to the point showing the largest improvement to the total weighted metrics. A move is also allowed if the total weighted metric value is the same as the start point. If all the possible moves are worse than the current value for the node, no movement is made.

A cooling option is provided in order to allow the maximum distance that a node can move to decrease linearly with each iteration. The reasoning behind this is that the graph initially requires relatively large node movements to form a overall structure for the layout. In later iterations, only minor movements need to be made to node positions in order to refine the layout within the structure developed in the first iterations.

We experimented with enforcing a geographical relationship rule whereby nodes that were to the north of those nodes stayed to the north of other nodes, nodes to the east stayed to the east, etc. However, we decided that enforcing these rules was not flexible enough as there are many situations where the rule has to be broken to achieve a better layout (particularly in the case of highly connected regions of graphs).

Finally, before a movement is made, the graph is checked to make sure that the movement does not introduce edge crossings or that the node and its adjacent edges do not occlude any other nodes or edges.

2.4. Aesthetic Metrics

We implemented a total of five different metrics based on various geometric measurements that we believe affect the quality of the graph. A metric involves iterating through either all the nodes or all the edges in the graph and calculating a value for each item. These are then summed to provide the value for the metric. In order to overcome some of the problems with metric values being disproportionate, each metric value is multiplied by a weighting. Altering the weighting also allows the user to emphasize or de-emphasize particular criteria. All the metrics we implemented are invariant under scaling, so that if the graph (and underlying grid) are scaled, the value for the metric remains the same.

The five metrics that we implemented are:

Edge Crossings Metric. The edge crossings metric is defined as the total number of edge intersections. As this is typically a low number, the weighting for this particular metric tends to be significant compared to the weightings for other metrics. It can be argued that edge crossings have meaning in metro maps, representing a line that crosses another. However, as edges are represented as straight lines between stations and not as their true route, unwanted edge crossings may be inadvertently introduced into the initial layout of the graph. This metric will only remove edge crossings that are in the initial layout because node movement is constrained to never adding an edge crossing.

4-gonality Metric. The 4-gonality metric is a measure of how close edges are to being horizontal, vertical or at 45° diagonal [12]. The intention of this metric is to favour edges which are orthogonal or at a 45° diagonal. Other edges as penalized with respect to by how much an angle they differ from being orthogonal or at 45° diagonal.

The metric for a graph containing edges E is:

$$\sum_{(u,v)\in E}\left|\sin 4\left(\tan^{-1}\frac{|y(u)-y(v)|}{|x(u)-x(v)|}\right)\right| \qquad (1)$$

where (u, v) is an edge between nodes u and v, and $y(v)$ and $x(v)$ are the y- and x-coordinate of node v respectively.

Edge Length Metric. In order to make sure that nodes are spaced evenly along lines, it is necessary to try to minimize the length of edges in the graph. The smallest edge length will be no less than the spacing of one grid point, so edge lengths are considered as being multiples of the grid spacing.

The edge length metric for a graph containing edges E is:

$$\sum_{e\in E}\frac{|e|}{g} \qquad (2)$$

where $|e|$ is the length of an edge and g is the grid spacing. For contracted edges, the number of hidden nodes must be taken into account. The metric is therefore modified:

$$\sum_{e\in E}\left|\frac{|e|}{g}-(d+1)\right| \qquad (3)$$

where d is the number nodes of degree two that e represents. This will cause contracted edges which are too short as well as too long to contribute to the metric.

Angular Resolution Metric. This metric attempts to ensure that all the edges incident to a node are evenly spaced. For example, if the node has four incident edges, the ideal angle between each pair of adjacent edges will be 90°. Maximising the angular resolution in this way reduces the possible confusion between edges which would otherwise be drawn very close together.

To calculate the metric, the absolute value of the difference between the ideal angle and the angle between each pair of adjacent incident edges is found and summed for all the nodes v in the graph:

$$\sum_{v\in V}\left(\sum_{\substack{e1,e2\in E \text{ where}\\ e1\,\&\,e2 \text{ are incident}\\ \text{to } v \text{ and adjacent}}}\left|\frac{360°}{\deg(v)}-\theta(e1,e2)\right|\right) \qquad (4)$$

where deg(v) is the degree of node v and $\theta(e1,e2)$ is the angle between the adjacent edges *e1* and *e2*.

Line Straightness Metric. An obvious feature of public transport schematics is that lines should appear to pass straight through nodes much the same way as the metro line passes through the station.

In order to measure this metric, a subgraph for each line in the graph is considered and all the nodes of degree two are then found. For each of these nodes, the difference in angle between the two incident edges is found:

$$\sum_{v \in V} \left(\sum_{\substack{e1, e2 \in E \text{ where } e1 \& e2 \\ \text{are the only two edges} \\ \text{of the same line incident to } v}} \theta(e1, e2) \right) \quad (5)$$

where $\theta(e1, e2)$ is the angle between the adjacent edges *e1* and *e2*.

Figure 2 shows a simple example of how the line straightness metric is calculated. The angles θ_1 and θ_2 show the angles which are measured as part of the metric.

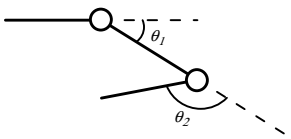

Figure 2. Line straightness metric example

Therefore, if the two edges are parallel, the metric evaluates to 0. If the two edges are at right angles, the metric evaluates to 90. This will penalize edges which double back on themselves more than edges which bend only slightly.

If a node in the subgraph for a line has a degree of more than two (for example, a line that branches), the node is not included in the line straightness metric. This is because it is not obvious as to which of the edges is the main line (and should therefore be straightened) and which is the branch line.

Total Metrics. The hill climber uses the sum of the weighted metrics to calculate a value for the total aesthetic metric for any particular layout of the graph.

2.5. Dealing with Over-length Edges

One of the problems that we discovered while developing this work was that the graph would tend to form small clusters of nodes, particularly at the ends of lines or where a line branched. This is because with only moving one node at a time, the nodes at the edge of each cluster can't move towards the rest of the graph. A feature of these clusters is that the they are connected to the rest of the graph by one (or more) long edge.

We initially tried to find the clusters using the k-means clustering algorithm. However, this method is not used because we discovered clusters are not easy to find, particularly where the number of clusters and the number of nodes in each cluster varies.

Our next approach was to limit ourselves to clusters that formed at the end of lines. These are relatively easy to find by using an algorithm to find over-length bridge edges. These are edges which are longer than one grid spacing and whose removal would disconnect the graph.

To find bridge edges, a recursive algorithm is used. Starting with an over-length edge, the graph is recursively explored starting from the two endpoint nodes of the over-length edge. If at any point the algorithm comes upon the node at the other end of the over-length edge, it can be assumed that the edge is not a bridge edge (there must have been some other route between the two endpoint nodes rather than directly along the over-length edge). If the over-length edge is a bridge edge, then the algorithm returns two sets of nodes, representing the subgraph formed at each end of the over-length edge.

Once all the over-length bridge edges are found, all of the nodes that are on the far end of the bridge edge (the smaller of the two sets of nodes) are moved so they are closer to the rest of the graph. If both sets of nodes are the same size, then one set is chosen arbitrarily. Figure 3 shows an example of a bridge edge where the edge *b* is over-length. Neither of the two end nodes of *b* can move towards each other without increasing the length of two or more edges. In this case, the three nodes to the right of *b* will be moved closer to the other four nodes as it is the smallest subset of nodes. This process is implemented as an optional step to be performed after each iteration.

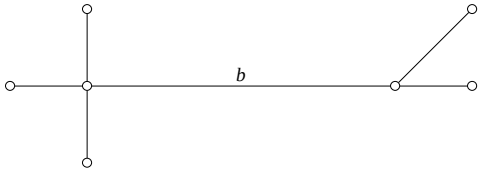

Figure 3. Bridge edge example

2.6 Labelling

We have implemented a simple labelling strategy for station names. Station labels can be oriented in up to eight different directions, listed in order of preference: horizontally to the right of the node, horizontally to the left of the node, 45° diagonally above-right, below-right, above-left and below-left of the node, vertically above the node or vertically below the node. To determine in which orientation to place a label, each possible orientation is checked in order of preference and the first location which is clear of incident edges is used for the label. An example of labelling a map is shown in Figure 8. No checks are made to make sure that the label does not obscure other nodes, edges or labels.

3. Examples

To illustrate the performance of our algorithm, we show how it performs on three metro maps of increasing size and complexity. The Atlanta MARTA Rail map [9]

is an example of a small metro map with only two lines and 38 nodes. The Washington Metro [13] is an example of a medium sized map with 86 nodes and five lines. For an example of a large metro map, we use the Sydney Suburban CityRail map [11] which has 172 nodes and seven lines. The layout of the Atlanta map is dealt with in more detail than the other two maps to show intermediate stages during the running of the algorithm.

Examples were run on a 2.4GHz Pentium 4 machine with 512MB of RAM using Java 2 v1.4.2. The running time in seconds for each map using both contracted and uncontracted edges with the metrics weightings as shown in the following sections and with ten iterations is shown in Table 1. As the complexity of the map increases, the time taken to layout the map increases. Using the preprocessing step to contract edges significantly reduces the running time of the algorithm.

Map	Uncontracted Graph - Time	Contracted Graph - Time
Atlanta	10.665	0.260
Washington	161.394	16.865
Sydney	1690.228	241.590

Table 1. Running times for the three examples

On most of the examples, labels have been omitted for clarity.

3.1. Atlanta MARTA Rail Map

The Atlanta MARTA Rail Map is used as an example of a small metro map. It's structure is that of a tree as it has no cycles. The initial layout of the map is shown in Figure 4, and like all our examples, the starting layout is the geographic position of the stations. The metric weightings used to generate this map are: edge crossings, 10000; 4-gonality 8.0; edge length 7.0; angular resolution 0.0; line straightness 1.0. Angular resolution has a value of 0.0 because it has little effect in this particular example and therefore discounting it speed up finding the final layout.

Figure 5 and Figure 6 show the graph during and after the first iteration. It is evident that only one iteration is required to remove any edges that are not 4-gonal. It can also be seen in Figure 6 that all the over-length edges are shortened to their shortest length. However, owing to not enforcing geographical relationships between nodes, the single-node branch at the western end of the darker east-west line has been moved to the wrong side of the line.

Figure 7 shows the final layout of the Atlanta map after 10 iterations. Figure 8 shows the same final layout, this time with labels to illustrate the labelling of station names. It is clearly a much improved layout than the initial layout shown in Figure 4. The overall topology of the map has been preserved. However, the darker east-west line seems to have skewed. This is partially due to the order in which the nodes are processed by the algorithm. Also, neither of the two nodes either side of the intersection with the other line can move without introducing an extra bend in the line. This is a problem that could be solved by using contracted edges.

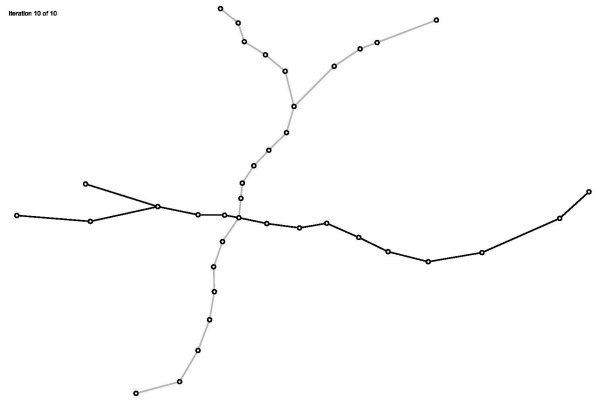

Figure 4. Initial layout of the Atlanta MARTA Rail map

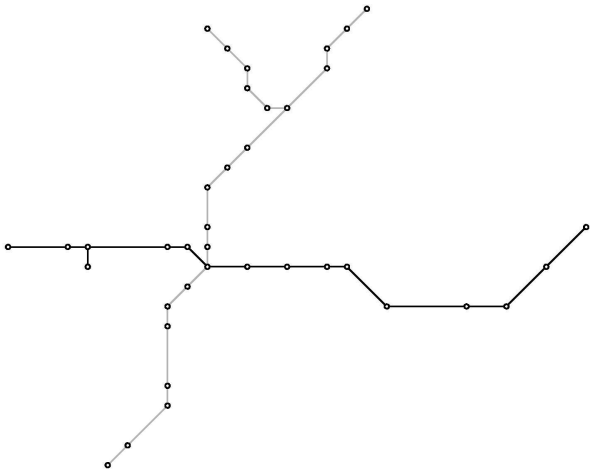

Figure 5. During the first iteration, before over-length edges are processed

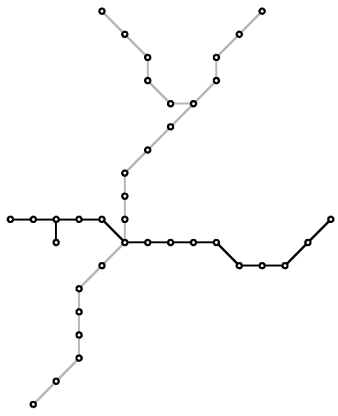

Figure 6. After the first iteration

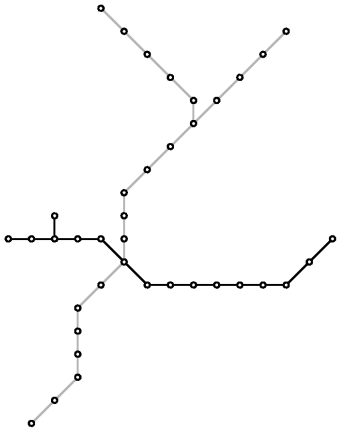

Figure 7. Final layout of the Atlanta MARTA Rail map

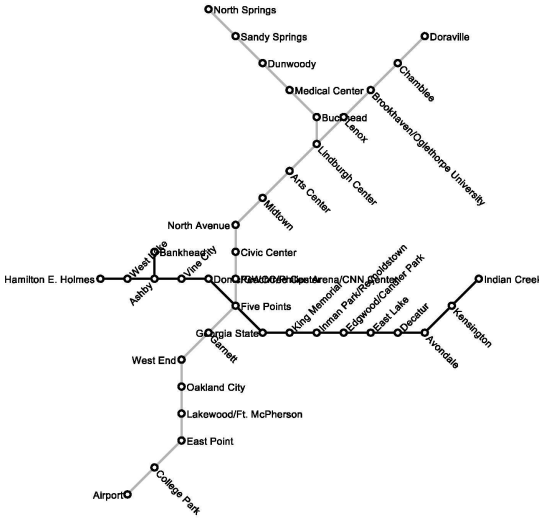

Figure 8. Final layout of the Atlanta MARTA Rail map with labelling

3.2. Washington Metro

The Washington Metro map is more complex than the Atlanta map. The initial layout of the Washington Metro map is shown in Figure 9. Of note are the introduction of loops of nodes, particularly in the centre of the map and on the two north-eastern lines. The map also has multiple edges between certain stations. For this map, the algorithm used the edge contraction preprocessing step, the result of which can be seen in Figure 10. The metric weightings used to generate this map are: edge crossings, 10000; 4-gonality 5.0; edge length 4.0; angular resolution 0.05; line straightness 2.0.

Figure 11 shows the final layout of the Washington map after four iterations. The finished graph is greatly improved from the initial graph, with lines significantly straightened and in the majority of cases each line can be followed easily enough. The resulting graph shows a problem with the loop of stations at the north-east where the loop has been significantly squashed. This problem is caused by the edge contraction stage. A possible solution to this problem would be to use one or two intermediate nodes as 'anchor-node' so that both halves of the loop are not totally contracted. Another problem also arises in the centre of the graph where some of the edges are drawn with a disproportionate length. However, this is also a problem for the designers of the real map, so perhaps it is unreasonable to draw each edge with the same length.

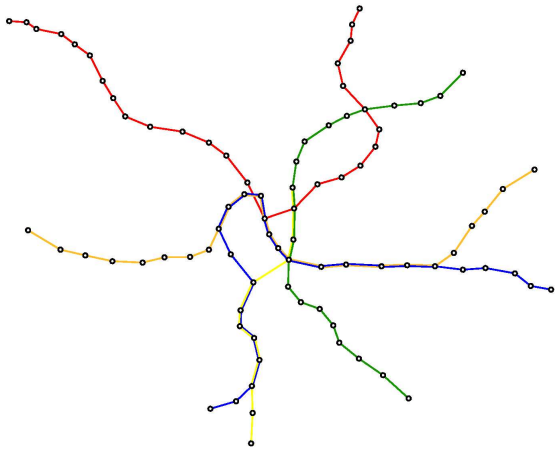

Figure 9. Initial layout of the Washington Metro Map

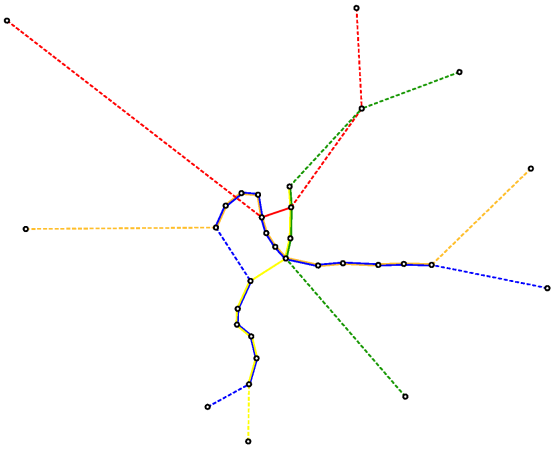

Figure 10. Initial layout with contracted edges (dotted edges represent contracted edges)

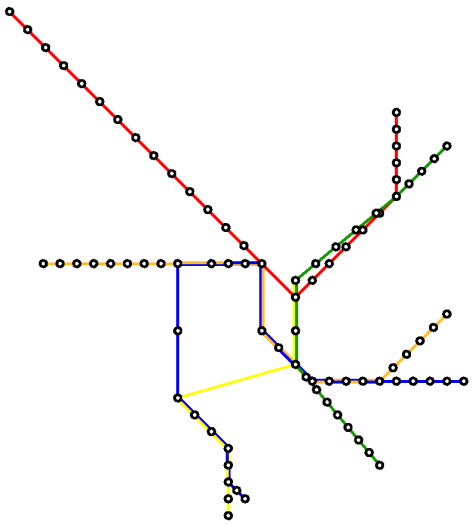

Figure 11. Final layout of the Washington map

3.3. Sydney Suburban CityRail

The Sydney Suburban CityRail map (shown as part of Figure 12) is used as an example of using our algorithm to lay out a large metro map. We only use the suburban part of the map and leave out the intercity lines such as the northernmost line. This makes it difficult to compare our results against those in [4], where a larger map is used. Figure 13 shows the initial layout of the CityRail map while Figure 14 and Figure 15 show two final layouts after ten iterations, the first a layout generated using contracted edges, the second without. The metric weightings used to generate this map are: edge crossings, 10000; 4-gonality 8.0; edge length 7.0; angular resolution 0.0; line straightness 1.0.

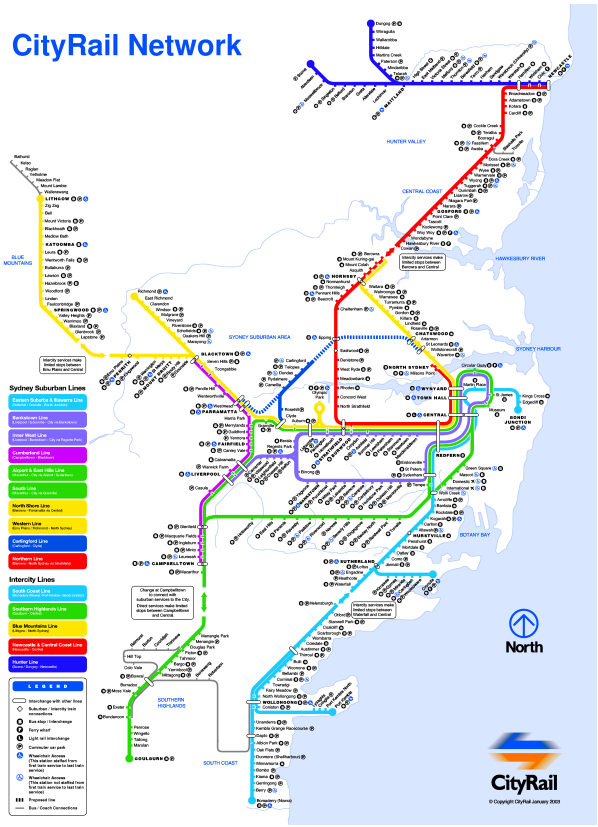

Figure 12. Sydney CityRail map

Both the final graphs show significant straightening of all the lines. The graph drawn without edge contraction is closer to the initial layout but at the cost of more time required to create the layout. We believe that both of the final graphs are preferable to the approximation when it comes to using them for navigation round the network.

In Figure 14, some contracted edges are clearly not 4-gonal. There are two possible reasons for this. Firstly, that it is not possible to move the nodes at the end of the contracted to improve the metrics without moving more than one node. Secondly, that in the case of diagonal edges, because the integer grid restricts the possible positions for nodes, which means that the best position for the endpoints of the edge is slightly offset from the 45° diagonal in order to satisfy the edge length metric. Some sections of the graph also suffer from irregular node spacing, mainly where more than one bridge edge partitions the graph.

Figure 15 suffers from similar problems to Figure 14, mainly with respect to irregular spacing of nodes along lines. Both graphs struggle to cope with the small loop in the middle on the right; the loop is excessively large in order to accommodate the small branch line inside the loop.

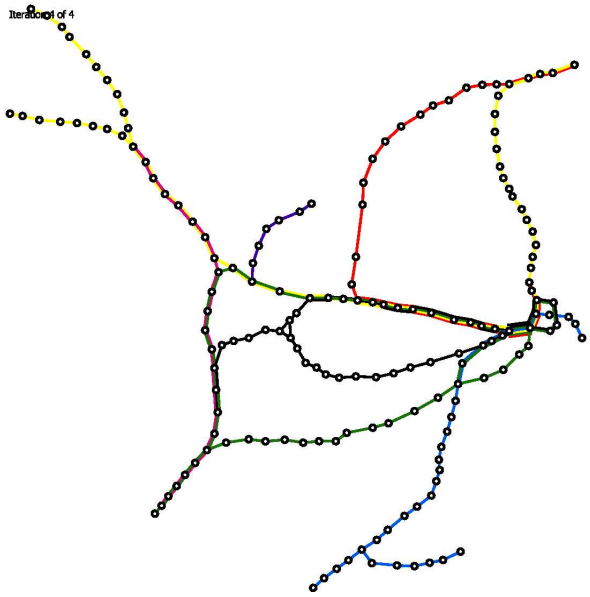

Figure 13. Initial layout of the Sydney CityRail map

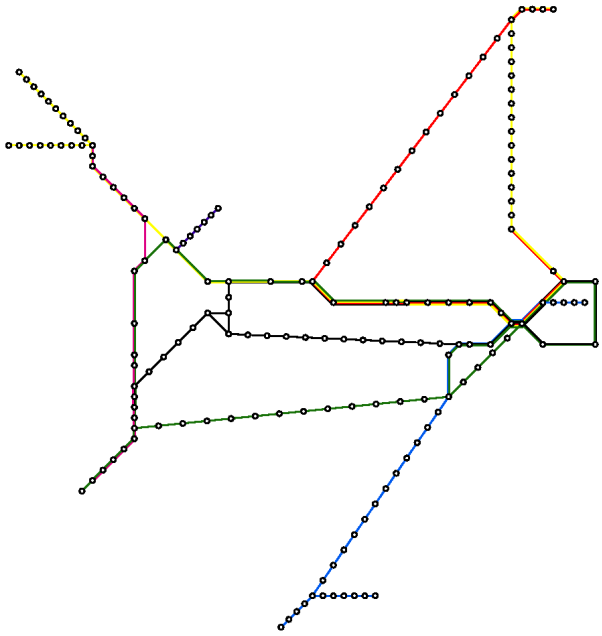

Figure 14. Final layout of the Sydney CityRail map using contracted edges

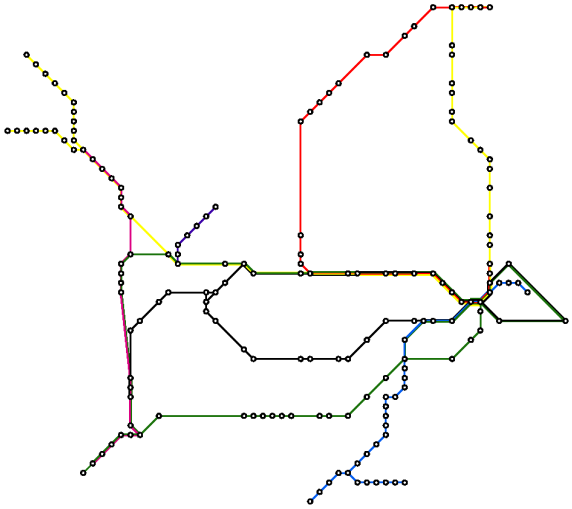

Figure 15. Final layout of the Sydney CityRail map without using contracted edges

4. Conclusions and Future Work

In this paper we presented an algorithm using a hill climbing multicriteria optimisation technique to automatically generate good layouts of metro maps. We implemented a preprocessing step to contract nodes of degree two and an additional step at each iteration of the algorithm to deal with over-length edges. The final layouts of the metro maps show a significant improvement on the original geographic layout and make the maps much easier to navigate.

We believe that out multicriteria optimisation technique can be extended relatively easily with other metrics in order improve further on the quality of the final map layouts. In particular, metrics can be introduced so that station labels are positioned with regard to the rest of the map – for example, it is generally desirable when labelling metro maps that labels never obscure edges, nodes or other labels. Other metrics might include constraining the graph to a certain area or to reduce the chance that a node could move close to another edge. Another improvement would be to use polylines to draw edges that cannot be drawn either orthogonally or diagonally.

There are also other problems that need to be considered. These include dealing with highly connected maps, contracting multiedges and reducing over-length edges when there is more than one over-length bridge edge separating the graph.

As the size of the network increases, the speed of our system degrades significantly. There is a great deal of optimisation that could be implemented, for example the metrics are completely recalculated for each node movement. This could be improved by the system reusing many of the calculations in subsequent iterations. The calculations of the metrics could be speeded up by integrating the calculation of multiple metrics.

5. References

[1] G. Battista, P. Eades, R. Tamassia and I. Tollis. Graph Drawing: Algorithms for the Visualisation of Graphs. Prentice Hall. 1999.

[2] R. Davidson, D. Harel. Drawing Graphs Nicely Using Simulated Annealing. ACM Trans. Graphics, 15(4):301-331, 1996.

[3] K. Garland. *Mr. Beck's Underground Map*. Capital Transport Publishing. England. 1994.

[4] S. H. Hong, D. Merrick, H. A. D do Nascimento. *The Metromap Layout Problem*. Technical Report IT-IVG-2003-03, School of IT, University of Sydney, 2003.

[5] S. H. Hong, D. Merrick, H. A. D do Nascimento. The metro map layout problem, in N. Churcher and C. Churcher, eds, Information Visualisation 2004, Vol. 35 of Conferences in Research and Practice in Information Technology, ACS, pp. 91-100. 2004.

[6] M. Kaufmann and D. Wagner. Drawing Graphs: Methods and Models, LNCS 2025. 2001.

[7] Ulrich Lauther, Andreas Stübinger. Generating Schematic Cable Plans Using Springembedder Methods. Proc. Graph Drawing 2001. LNCS 2265. pp. 465-466. Springer. 2001.

[8] M. Ovenden, *Metro Maps of the World*, Capital Transport Publishing, England, 2003.

[9] Metropolitan Atlanta Rapid Transport Authority network map, http://www.itsmarta.com/getthere/schedules/index-rail.htm. Accessed on 5 March 2004.

[10] E.S. Sandvad, K. Grønbæk, L. Sloth, J.L. and Knudsen. *A Metro Map Metaphor for Guided Tours on the Web: the Webvise Guided Tour System*. Proc. 10th International World Wide Web Conference, Hong Kong, May 1-5, 2001. ACM: New York, 2001. pp. 326-333.

[11] Sydney CityRail network map, http://www.cityrail.info/networkmaps/selection.jsp. Accessed on 5 March 2004.

[12] R. Tamassia. On Embedding a Graph in the Grid with the Minimum Number of Bends. SIAM Journal of Computing. Vol. 16. No. 3. pp. 421-444. June 1987.

[13] Washington Metro network map, http://www.wmata.com/metrorail/systemmap.cfm. Accessed on 5 March 2004.