The Metro Map Layout Problem *

Seok-Hee Hong **Damian Merrick** **Hugo A. D. do Nascimento**

School of Information Technologies
University of Sydney,
NSW 2006, Australia
Email: shhong,dmerrick@it.usyd.edu.au
Instituto de Informatica-UFG, Brazil
Email: hadn@inf.ufg.br

Abstract

We address a new problem of *automatic* metro map layout. In general, a metro map consists of a set of lines which have intersections or overlaps. We present a method to produce a good layout of a metro map automatically. Our method uses a variation of the spring algorithm with a suitable preprocessing step. The experimental results with real world data sets show that our approach is promising.

Keywords: metro map layout, spring algorithm.

1 Introduction

A metro map is a simple example of a geometric network that appears in our daily life. For example, see the Sydney Cityrail NSW network in Figure 1 [14].

Futhermore, the *metro map metaphor* has been used successfully for visualisation of *abstract* information, such as visualisation of websites [9] or taxonomy [8]. For example, see the visualisation of relations of taxonomy which uses a metro map metaphor in Figure 2 [8].

Note that a metro map can be modeled as a graph, and automatic visualisation of graphs has received a great deal of interest from visualisation researchers over the past 10 years.

However, automatic visualisation of the metro map is a very challenging problem, as already observed in [6, 12]. Note that existing metro maps are produced manually. Hence, it would be interesting to know how far automatic visualisation methods can achieve the quality of the hand drawn pictures.

In this paper, we address this new problem of metro map layout. We present a method to produce a good layout of metro map automatically. Our method uses a variation of the spring algorithm with a suitable preprocessing step. The experimental results with real world data sets show that our approach is promising.

In the next section, we define the problem. We present our metro map layout methods in Section 3 and discuss metro map labeling methods in Section 4. In Section 5, we present the experimental results and Section 6 concludes.

*This is a *Work in Progress* paper. The first author was supported by a SESQUI grant from the University of Sydney. The last author was supported by UFG and CAPES-Brazil, Ph.D Scholarship of CAPES.

Figure 1: *Example of a metro map: Sydney Cityrail network.*

2 The Metro Map Layout Problem

A *metro map graph* consists of a graph G and a set of paths that cover all the vertices and edges of G. Some vertices and edges may appear in more than one path, but each occurs in at least one path. For example, see Figure 3.

A *layout* of a metro map consists of a drawing of the graph. Thus, the main problem of this paper can be formally defined as follows.

The Metro Map Layout Problem
Input: a metro map graph G.
Output: a good layout L of G.

Here, we need to define what is a *good* layout of a metro map graph. For this purpose, we have studied existing *hand-drawn* metro maps from all over the world [14, 15, 16, 17]. For example, a detailed study of the London metro map can be found in [6]. From these manually produced layouts, we derive the following criteria for a good metro map layout.

- *C1*: each line (or path) drawn as straight as possible.

Figure 2: *Example of a metro map metaphor.*

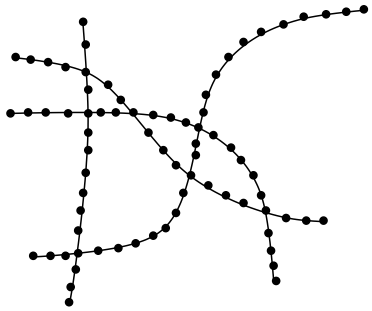

Figure 3: *Example of a metro map graph.*

- *C2*: no edge crossings.
- *C3*: no overlapping of labels.
- *C4*: paths are mostly drawn horizontally or vertically, with some at 45 degrees.
- *C5*: each line drawn with unique color.

Based on these criteria, we designed layout methods. It should be noted that we do not pursue exact geometry or topology here. This is partly because, in general, the metro map metaphor can be used for visualisation of abstract information which has no fixed geometry. Another reason is that, the most common usage of the metro map is mainly for navigation, that is, to find out how to get a destination. For example, consider the situation where a visitor to London (who does not know the exact geometry of London) uses the metro map for navigation. Hence, achieving exact geometry or topology is not the primary aim of our project. Rather, we try to achieve $C1, C2, C3, C4$ and $C5$.

3 The Layout Methods

We have tried five different layout methods using various combinations of spring algorithms. The tools that we use are GEM [4], PrEd [1] and a magnetic spring algorithm [11]. In summary, each method can be briefly described as follows.

1. `Method 1`: The GEM algorithm.
2. `Method 2`: We simplify the metro map graph using a preprocessing step described in Section 3.2 and use the GEM algorithm *with* edge weight (details are explained later).
3. `Method 3`: We simplify the metro map graph and use the GEM algorithm *without* edge weight. Then we use the PrEd algorithm *with* edge weight.
4. `Method 4`: We simplify the metro map graph and use the GEM algorithm without edge weight. Then we use the PrEd algorithm with edge weight, plus *orthogonal* magnetic spring algorithm.
5. `Method 5`: We simplify the metro map graph and use the GEM algorithm without edge weight. Then we use the PrEd algorithm with edge weight, orthogonal magnetic spring algorithm, plus *45 degree* magnetic field forces.

We now describe each method in detail.

3.1 Method 1

GEM, used in Method 1, is a generic spring embedder. We use GEM mainly as a baseline for comparison.

3.2 Method 2

First, we explain the preprocessing step. Note that there are many vertices of degree two in the metro map graph G. However, they do not contribute to the *embedding*, that is the overall topology or layout, of the metro map graph. This motivates us to remove these vertices and define a simplified metro map graph G'. The resulting graph has only intersection vertices and vertices with degree 1. For example, the metro map graph in Figure 3 can be simplified as in Figure 4. Note that special care is needed to handle a self loop or multiple edges.

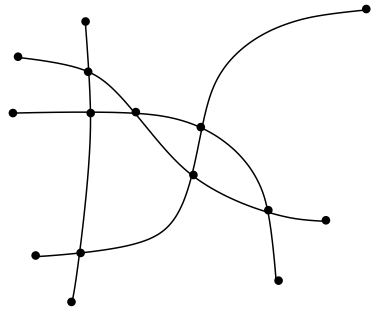

Figure 4: *Example of a simplified metro map graph.*

After drawing the simplified graph G', we need to reinsert those removed vertices to get a layout of G. This requires space; hence we define edge weights according to the number of removed vertices to edges of G' and produce a layout of G' which reflects those edge weights. Thus, Method 2 can be described as follows.

`Method 2`

1. Compute a simplified metro map graph G' by removing degree two vertices from the metro map graph G.
2. Produce a layout L' of G' using the GEM algorithm with edges weighted according to the number of vertices removed at Step 1.
3. Produce a layout L of G by reinserting the removed vertices, spaced evenly along the edges in the layout L'.

3.3 Method 3

Method 3 is to use the preprocessing step, the GEM algorithm and the PrEd algorithm [1]. The PrEd algorithm is a special force directed method that preserves the topology of its input layout. For our purpose, we modified PrEd algorithm to include edge weights.

In our modification, the x-components of the attraction force $F^a(u,v)$ and the repulsion force $F^r(u,v)$ between two vertices u and v are defined as follows:

$$F_x^a(u,v) = \frac{d(u,v)}{\delta(u,v)}(x(v) - x(u)) \quad (1)$$

$$F_x^r(u,v) = \frac{-\delta(u,v)^2}{d(u,v)^2}(x(v) - x(u)) \quad (2)$$

where $x(u)$ and $x(v)$ are the x coordinates of vertices u and v respectively, $d(u,v)$ is the distance between vertices u and v, and $\delta(u,v)$ is an ideal distance between the two vertices defined as $\delta(u,v) = L \times min(W, weight(u,v))^2$, where L, W are positive constants and $weight(u,v)$ is the weight of the edge between u and v. In the case of multiple edges between u and v, the maximum weight of that set of edges is used.

The y-components $F^a y(u,v)$ and $F_y^r(u,v)$ of the force vectors are computed similarly.

Node-edge repulsion forces are computed identically as in the original PrEd algorithm, that is:

$$F_x^e(v,(a,b)) = -\frac{(\gamma - d(v,i_v))^2}{d(v,i_v)}(x(i_v) - x(v)) \quad (3)$$

if $i_v \in (a,b)$, $d(v,i_v) < \gamma$, otherwise $F_x^e(v,(a,b)) = 0$.

As in the PrEd algorithm, the total force acting on a vertex v is calculated by summing all attraction and repulsion forces on that vertex, that is:

$$\begin{aligned} F_x(v) \quad = \quad & \sum_{(u,v)\in E} F_x^a(u,v) + \sum_{u\in V} F_x^r(u,v) \\ & + \sum_{(a,b)\in E} F_x^e(v,(a,b)) \\ & - \sum_{u,w\in V,(v,w)\in E} F_x^e(u,(v,w)) \end{aligned} \quad (4)$$

We now describe Method 3.

`Method 3`

1. Compute a simplified metro map graph G' by removing degree two vertices from the metro map graph G.
2. Produce an initial layout L' of G' using the GEM algorithm (with no edge weights).
3. Produce a better layout L'' of G' using the PrEd algorithm, modified to include edges weights.
4. Produce a layout L of G by reinserting the removed vertices, spaced evenly along the edges in the layout L''.

3.4 Method 4

Method 4 is to use the preprocessing step and the GEM algorithm and the PrEd algorithm with orthogonal magnetic springs, that is, with horizontal and vertical forces.

Forces are calculated as for Method 3, but with the addition of magnetic field forces acting on each edge. Equal and opposite forces are applied to each vertex of an edge to attempt to align that edge with a horizontally or vertically directed vector [11].

The magnitude of a magnetic force from an individual force field vector on the edge connecting vertices u and v is determined by a similar calculation to that of a magnetic spring [11]:

$$F^m(u,v) = c_m b d(u,v)^\alpha \theta^\beta \quad (5)$$

where b represents the strength of the magnetic field, θ is the angle between the edge (u,v) and the magnetic force vector, and $c_m, \alpha, \beta > 0$ are model-tuning constants.

Four force field vectors are used - left, right, up and down directed vectors. At any instant only a single magnetic force is applied to a given edge. The magnitude of the force applied is calculated according to the above equation for the force field vector to which the edge has the lowest angle θ. The direction of the force applied is perpendicular to the direction of the edge. To effect the desired rotational force on the edge, a force of magnitude $F^m(u,v)$ is applied to one vertex of the edge and a force of magnitude $-F^m(u,v)$ is applied to the other.

With the addition of the magnetic field force, the total force applied to a vertex v becomes:

$$\begin{aligned} F_x(v) \quad = \quad & \sum_{(u,v)\in E} F_x^a(u,v) + \sum_{u\in V} F_x^r(u,v) \\ & + \sum_{(a,b)\in E} F_x^e(v,(a,b)) \\ & - \sum_{u,w\in V,(v,w)\in E} F_x^e(u,(v,w)) \\ & + \sum_{(u,v)\in E} F_x^m(u,v) \end{aligned} \quad (6)$$

We now describe Method 4.

`Method 4`

1. Compute a simplified metro map graph G' by removing degree two vertices from the metro map graph G.
2. Produce an initial layout L' of G' using the GEM algorithm (with no edge weights).
3. Produce a better layout L'' of G' using the PrEd algorithm, modified to include edge weights and orthogonal magnetic field forces.
4. Produce a layout L of G by reinserting the removed vertices, spaced evenly along the edges in the layout L''.

3.5 Method 5

Method 5 is to use the preprocessing step and the GEM algorithm and the PrEd algorithm with orthogonal magnetic springs and 45 degree magnetic forces. Forces are calculated as for Method 4, but with the addition of four diagonal magnetic force field vectors.

Vectors running bottom-left to top right, bottom-right to top-left, top-right to bottom-left and top-left to bottom-right are added to the set of magnetic field forces used. Magnetic field forces are calculated as described for Method 4, and the equation for the total force acting on a vertex v does not change. We now describe Method 5.

Method 5

1. Compute a simplified metro map graph G' by removing degree two vertices from the metro map graph G.
2. Produce an initial layout L' of G' using the GEM algorithm (with no edge weights).
3. Produce a better layout L'' of G' using the PrEd algorithm, modified to include edge weights and orthogonal magnetic field forces and 45 degree magnetic forces.
4. Produce a layout L of G by reinserting the removed vertices, spaced evenly along the edges in the layout L''.

4 Metro Map Labeling

The second part of this project is to produce a good labeling for metro map layout. We use a well known combinatorial approach for labeling map features [3, 13, 10], where a predefined set of label positions is assigned to every feature and a subset of these positions is chosen for producing an overlap-free label placement.

The first step of the approach is to specify the predefined label positions. Figure 5 illustrates sets with eight label positions for two map features (in our case, the features are metro stations).

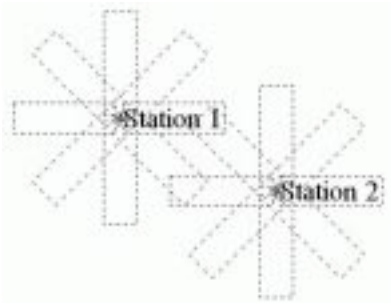

Figure 5: *Label positions for two map features. The labels were placed on the right-side position.*

The next step is to construct a conflict graph that describes all overlaps between label positions. The conflict graph has a vertex for every label position, and an edge linking every pair of label positions that overlap in the map. Moreover, the set of label positions for a feature forms a clique in the graph.

The labeling problem is then formulated as computing the maximum independent set with minimum cost [10] of the conflict graph. Such an independent set defines the maximum number of label positions that can be chosen for labeling the features without overlapping. Features whose label positions do not appear in the maximum independent are left unlabeled. We note that generating a good labeling solution even for a well defined situation is a difficult computational task, as the Independent Set Problem is NP-hard. However, good heuristics are available [7].

We apply this automatic process for producing a good initial labeling solution. After that, we use an interactive system for manually improving the labeling.

A manual adjustment is necessary whenever the automatic approach does not consider some important domain knowledge for map making. For instance, if one of two features with the same general importance has to stay unlabeled in order to allow an overlap-free labeling solution, then the choice of which feature to label may be made arbitrarily by the labeling algorithm. However, a map maker may prefer to choose a particular label based on some subjective domain knowledge not included in the combinatorial model. The map maker may also prefer to specify a different set of label positions for the problematic features, or to reduce the font size of the labels to avoid overlaps.

We have labeled the metro stations of the maps in this paper using `LabelHints` system [7]. This offers several tools for exploring map labeling solutions by manipulating conflict graphs. In experiments done with the system and in studies of other metro maps we observed that labels with diagonal orientation (45 degrees) are visually more pleasing. We have, therefore, decided to use mostly this orientation in our metro maps.

5 Experimental Results

The layout algorithms were implemented as a plugin to `jjGraph` [5], an extensible graph editing package written in Java. The package includes existing implementations of the GEM and the PrEd algorithms, which were modified for the metro map layout algorithms.

Metro map data is stored in a custom text file format describing the sequence of stations along each line in the network. These files are read by the metro map plugin, which then lays out the network and displays the resulting graph layout in `jjGraph`. `jjGraph` permits the user to navigate and modify this graph layout, as well as providing save and image export functionality. The metro map plugin was later made to export a complete layout to another format to be loaded into the `LabelHints` software [7].

The tests were executed on a single processor 1.5GHz Pentium 4 machine with 512MB of RAM, and the code was run under the Sun Microsystems Java(TM) 2 Runtime Environment, Standard Edition.

We used four data sets: the Sydney Cityrail NSW network and the London metro map as typical large data set, the Barcelona city metro map as a medium size data set, and the Auckland network as a small data set. Details of the data sets are below. Let $G = (V, E)$ be the original metro map graph and $G' = (V', E')$ be the reduced metro map graph.

Sydney: $|V| = 319, |E| = 897, |V'| = 41, |E'| = 178$
London: $|V| = 271, |E| = 745, |V'| = 92, |E'| = 317$.
Barcelona: $|V| = 101, |E| = 111, |V'| = 22, |E'| = 32$.
Auckland: $|V| = 39, |E| = 50, |V'| = 7, |E'| = 9$.

We now explain our experience with these data sets. First, we present results of Sydney Cityrail network. In summary, the methods gradually improve both in terms of the running time and the quality of the layout. Details of the running time of each method is:

Method 1: 120417 ms.
Method 2: 656 ms.
Method 3: 1937 ms.
Method 4: 2129 ms.
Method 5: 2306 ms.

Note that Method 2 significantly reduces the running time over Method 1. Methods 3, 4 and 5 take

slightly longer than Method 2, due to use of two spring algorithms; however, they are still significantly faster than Method 1.

The results for Sydney Cityrail produced by each method are illustrated in Figures 6, 7, 8, 9 and 10 respectively. Note that each successive Figure improves the previous one; and Method 5 is clearly the best. It satisfies most of the criteria $C1, C2, C3$ and $C4$ that we wanted to achieve.

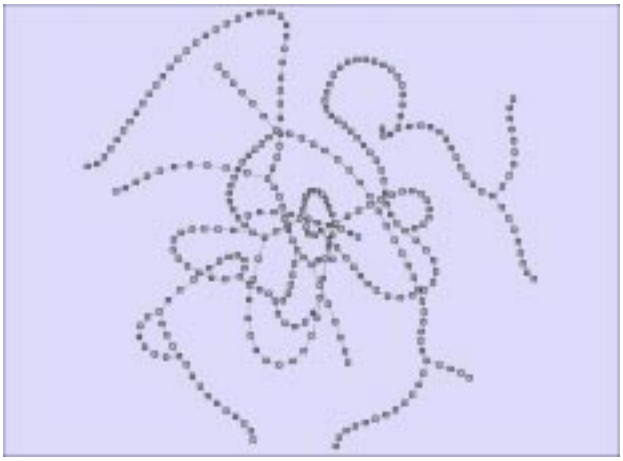

Figure 6: *Sydney Cityrail produced by Method 1.*

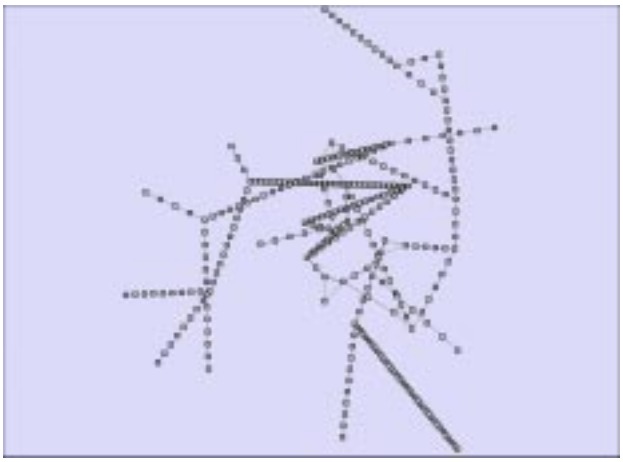

Figure 7: *Sydney Cityrail produced by Method 2.*

Since Method 5 produces the best result, we chose this layout for labeling. The results for the Sydney, Barcelona, Auckland and London metro maps with layouts produced by Method 5 and labels added using `LabelHints`, are shown in Figures 11, 12, 13 and 14 respectively.

The running times for Method 5 are:

Sydney: 2306 ms.
London: 22035 ms.
Barcelona: 275 ms.
Auckland: 124 ms.

6 Conclusion and Future Work

From our experiments, we have learned that carefully designed spring algorithms can be useful for the automatic layout of metro maps. Using a suitable preprocessing and magnetic springs, we obtain the results

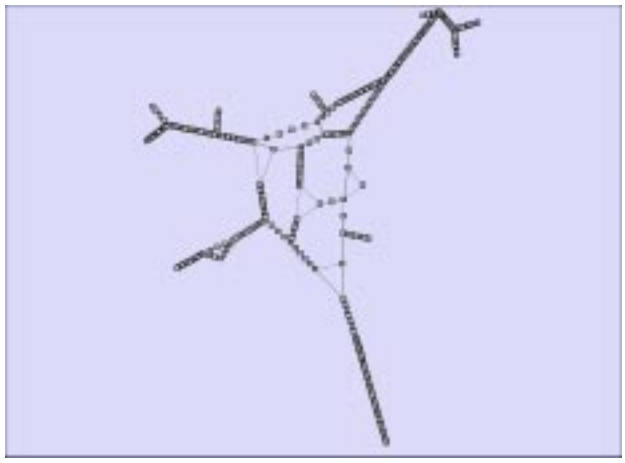

Figure 8: *Sydney Cityrail produced by Method 3.*

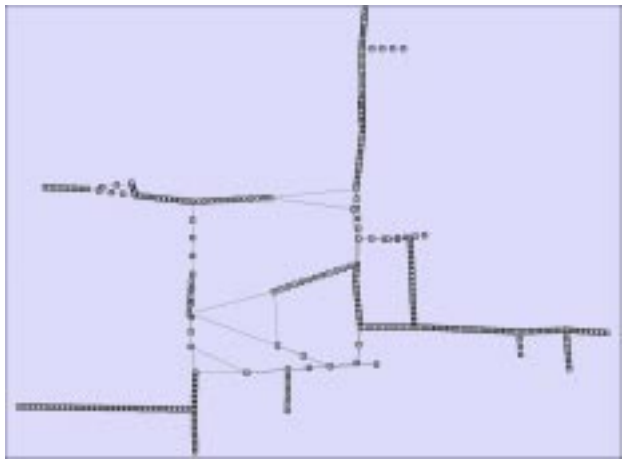

Figure 9: *Sydney Cityrail produced by Method 4.*

that satisfy criteria $C1, C2, C3$ and $C4$. These results are comparable (except in topology) to hand drawn metro maps.

Nevertheless, there is room for improvement. Our current system does not display correct geometry and multiple lines. In addition, it does not clearly support colors ($C5$). We plan to implement these features.

We also plan to do more experiments using other large data sets such as the Tokyo metro map, the Paris metro map, the Berlin metro map and the New York metro map. Furthermore, we want to apply other graph drawing approaches.

Finally, we would like to extend this method to visualize more complex geometric networks, such as the European railway networks.

We believe that automatic visualisation can not achieve significantly better quality than hand drawn visualisation by expert. However, automatic tools can certainly be used as a fast preprocessing step for producing a good quality visualisation very quickly.

7 Acknowledgement

We thank Peter Eades, who originally proposed this problem, for his interests and valuable feedback on this project.

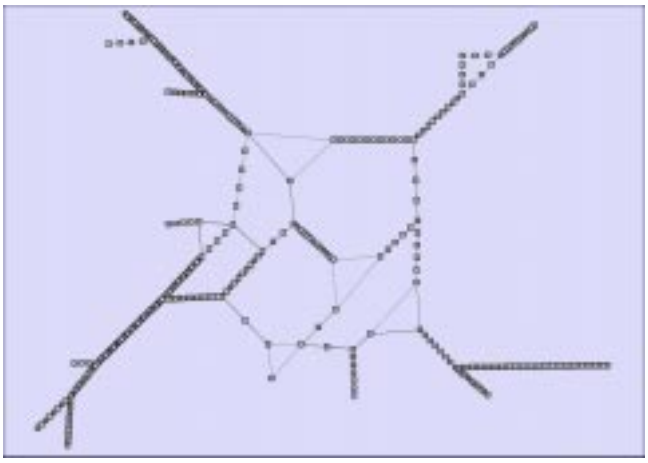

Figure 10: *Sydney Cityrail produced by Method 5.*

References

[1] F. Bertault, A Force-Directed Algorithm that Preserves Edge Crossing Properties, *Proc. of Graph Drawing 99*, LNCS 1731, pp. 351-358, Springer Verlag, 1999.

[2] G. Di Battista, P. Eades, R. Tamassia and I. G. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*, Prentice-Hall, 1998.

[3] S. Edmondson, J. Christensen, J. Marks, and S. Shieber, A General Cartographic Labeling Algorithm, *Cartographica*, Vol. 33, No. 4, pp.13-23, 1997.

[4] A. Frick, A. Ludwig and H. Mehldau, A Fast Adaptive Layout Algorithm for Undirected Graphs, Proc. of Graph Drawing 94, LNCS 894, pp. 388403, Springer Verlag, 1995.

[5] C. Friedrich, `jjgraph`, *personal communication.*

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

[7] H. A. D. do Nascimento and P. Eades, User Hints for Map Labelling, *Proc. of Australasian Computer Science Conference (ACSC2003)*, Vol. 16 of *Conferences in Research and Practice in Information Technology*, ACS, Adelaide, Australia, pp. 339-347, 2003.

[8] K. Nesbitt, Multi-sensory Display of Abstract Data, *PhD. Thesis*, University of Sydney, 2003.

[9] E. S. Sandvad, K. Gronbak, L. Sloth and J. L. Knudsen, Metro Map Metaphor for Guided Tours on the Web: the Webvise guided Tour system, *Proc. of International Conference on World Wide Web*, pp. 326-333, 2001.

[10] T. Strijk, B. Verweij and K. Aardal, Algorithms for Maximum Independent Set Applied to Map Labelling, Tech. Rep. UU-CS-2000-22, Department of Computer Science, Utrecht University, 2000.

[11] K. Sugiyama and K. Misue, Graph drawing by Magnetic Spring Model, *Journal of Visual Languages and Computing*, Vol.6, No.3, pp. 217-231, 1995.

[12] E. R. Tufte, *Visual Explanations*, Graphics Press, Cheshire, 1997.

[13] S. Zoraster, The Solution of Large 0-1 Integer Programming Problems Encountered in Automated Cartography, *Operations Research*, Vol. 38, No. 5, pp. 752-759, 1990.

[14] Sydney Cityrail NSW Netwrok Map, `http://www.cityrail.info/networkmaps/mainmap.jsp`.

[15] Auckland Tranz Metro, `http://www.tranzmetro.co.nz/auckland/default.asp`.

[16] Barcelona TMB metromap network, `http://www.tmb.net/eng/metro/metroplanol.jsp`.

[17] London Metro Map, `http://www.cityrail.info/networkmaps/mainmap.jsp`.

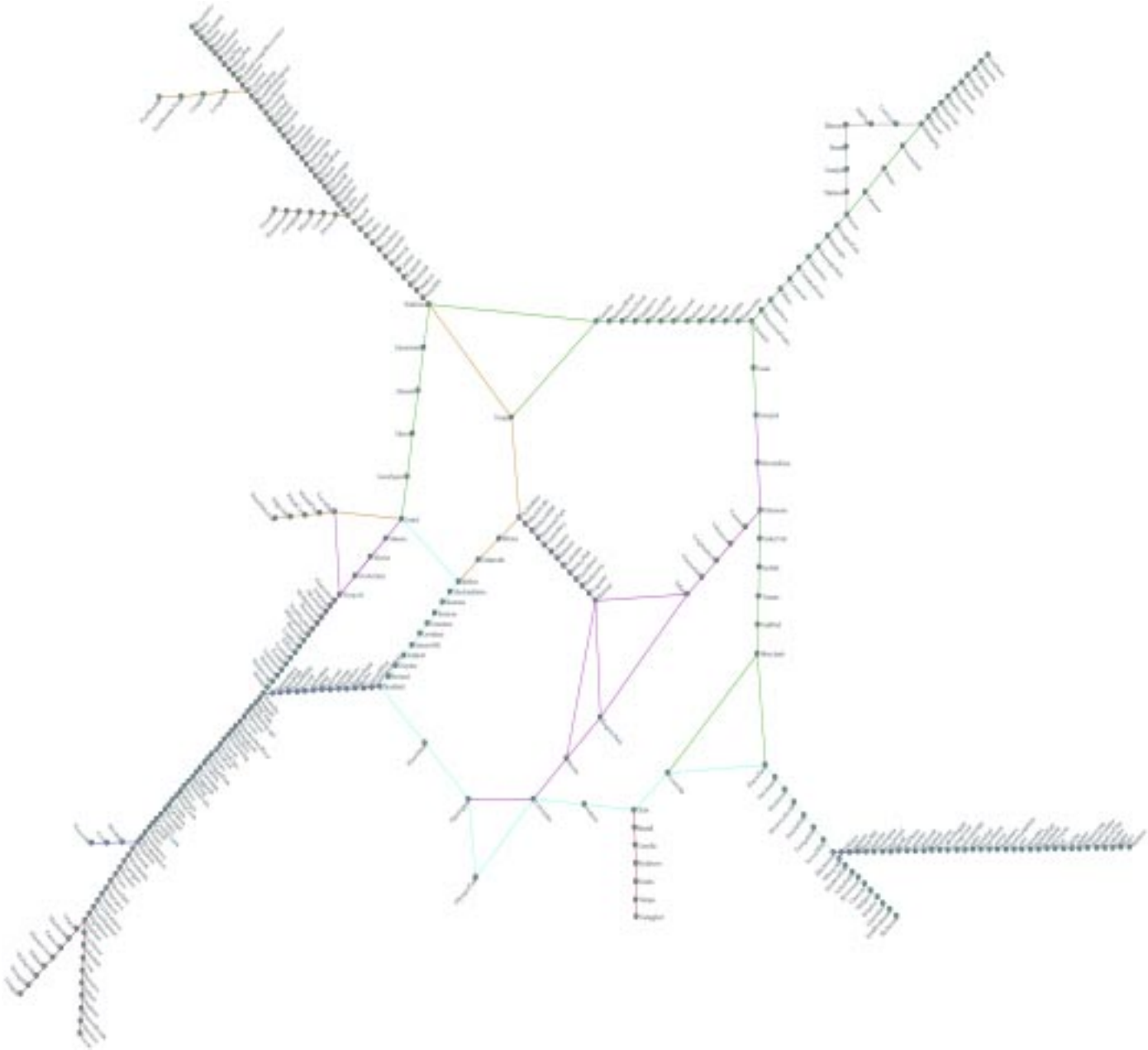

Figure 11: *Sydney Cityrail metro map with labeling.*

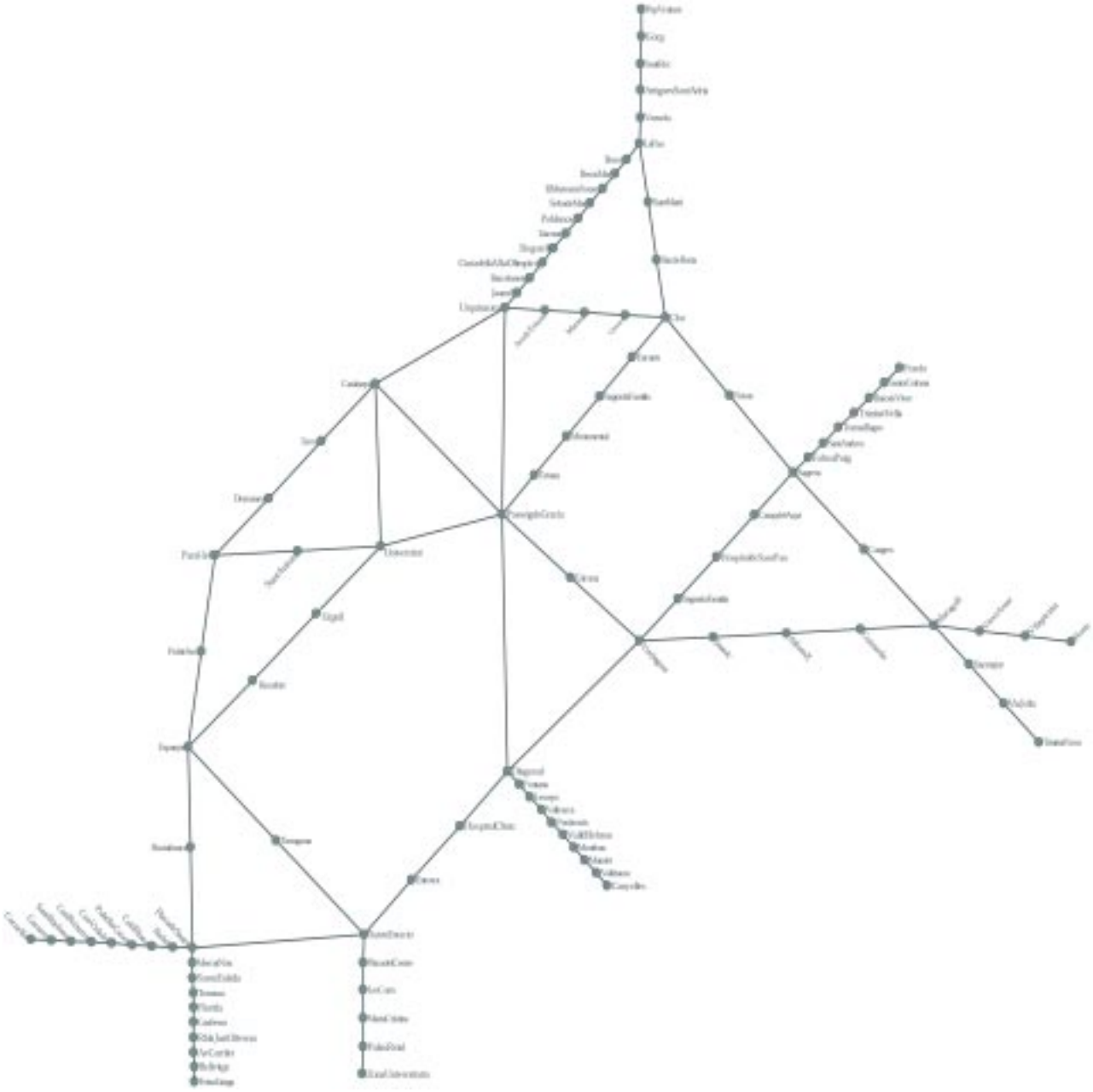

Figure 12: *Barcelona city metro map with labeling.*

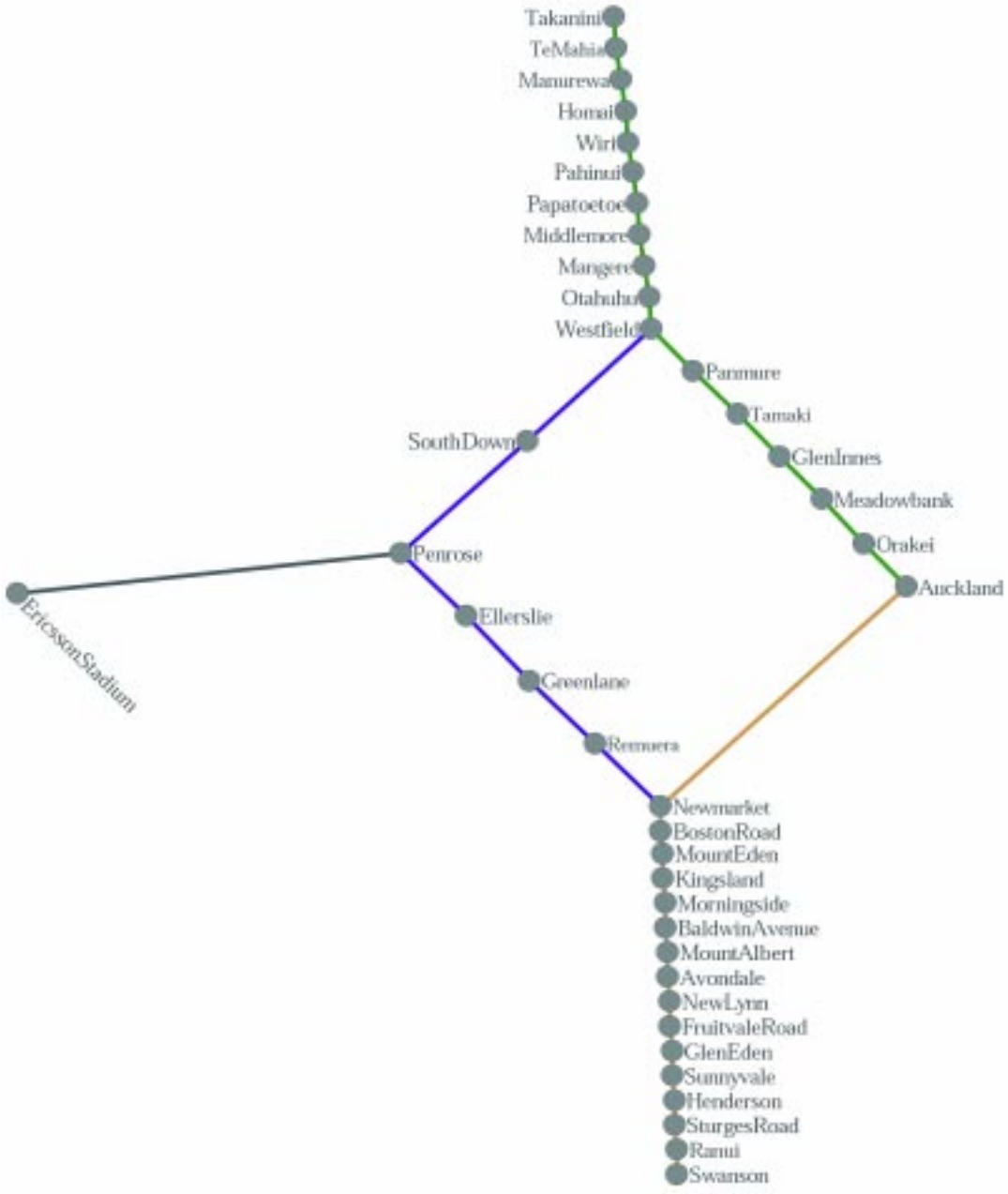

Figure 13: *Auckland metro map with labeling.*

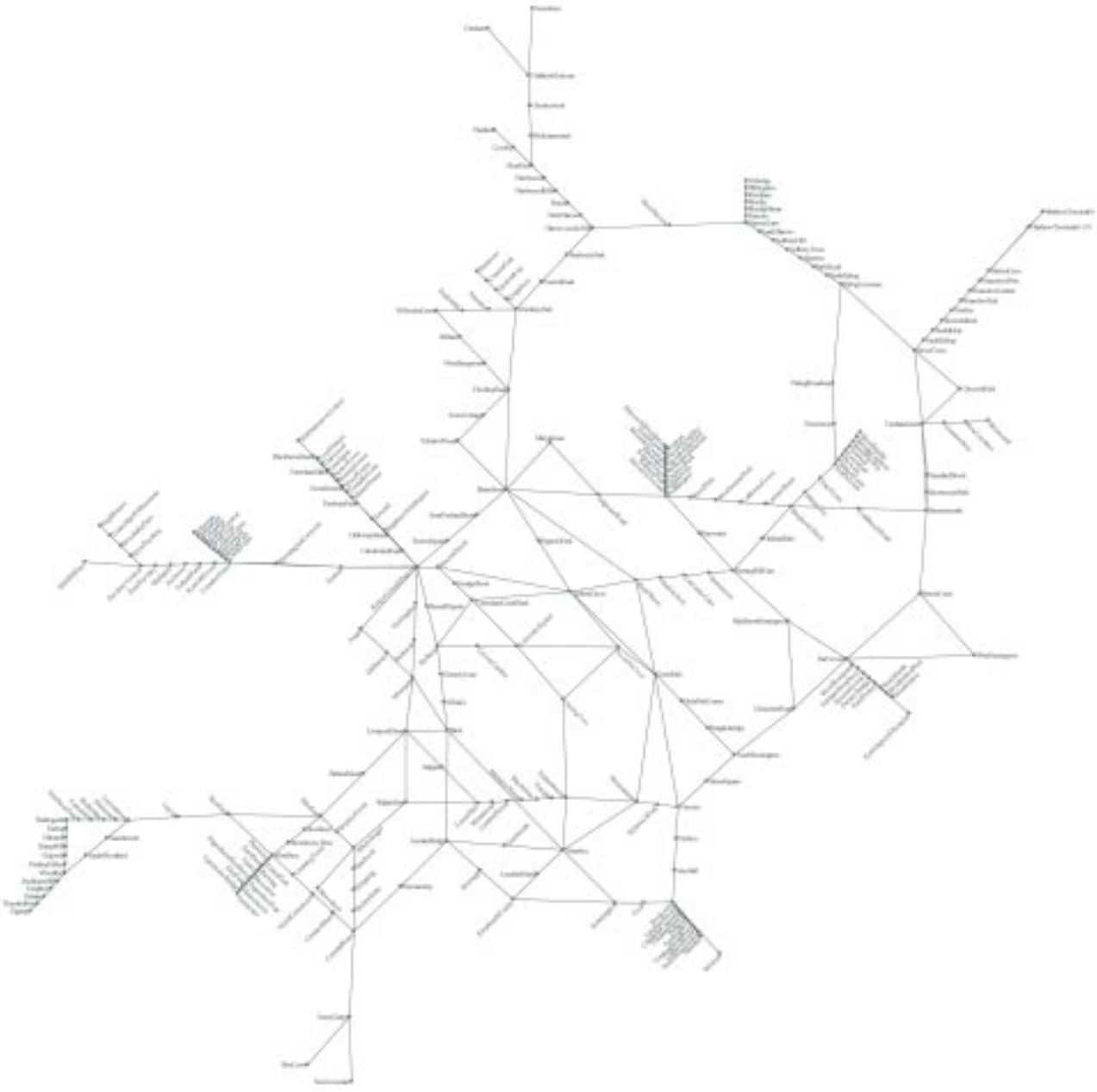

Figure 14: *London metro map with labeling.*