The Metro Map Layout Problem*

Seok-Hee Hong[1], Damian Merrick[1], and Hugo A. D. do Nascimento[2]

[1] National ICT Australia**; School of Information Technologies,
University of Sydney, Australia,
{shhong,dmerrick}@it.usyd.edu.au
[2] Instituto de Informatica-UFG, Brazil
hadn@inf.ufg.br

Abstract. We initiate 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 define a set of aesthetic criteria for good metro map layouts and present a method to produce such layouts 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 method produces good metro map layouts quickly.

1 Introduction

A metro map is a simple example of a geometric network that appears in our daily life. An example of such, the Sydney Cityrail NSW train network, is shown in Figure 1 (a) [13]. Furthermore, the *metro map metaphor* has been used successfully for visualising *abstract* information, such as the "train of thought" network in Figure 1 (b) [8], website networks [9], and networks of related books in Figure 5 (a) [14].

In general, 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 metro maps is a very challenging problem, as already observed by Beck [4] and Tufte [12]. Note that existing metro maps are produced manually. Hence, it would be interesting to know how far automatic visualisation methods can go towards achieving the quality of the hand drawn pictures.

In this paper, we address this new problem of metro map layout. We define a set of aesthetic criteria for good metro map layouts and present a method to produce such layouts 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 method produces good metro map layouts quickly.

---

*A preliminary version of this paper was published in [5]. For a version of this paper with full-size colour images, see [6].

**National ICT Australia is funded by the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council.

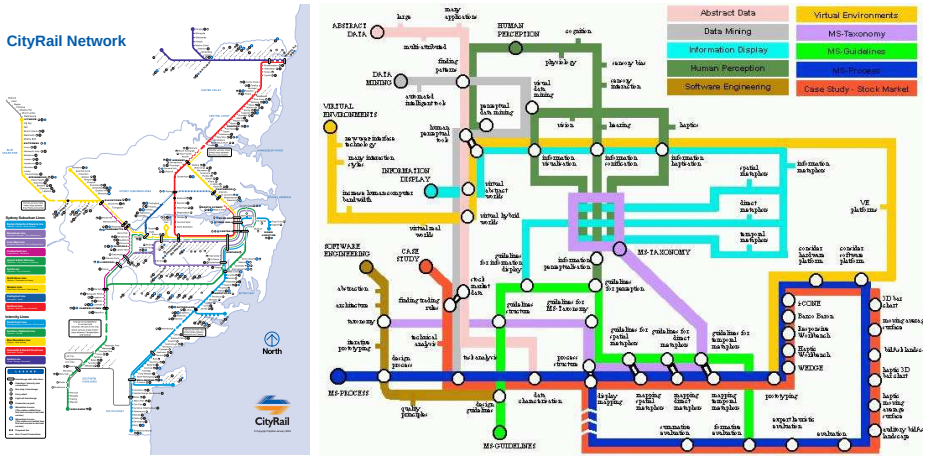

Fig. 1. *(a) Sydney Cityrail NSW network (b) "Train of thought" network.*

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.

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. See Figure 2 (a) for an example.

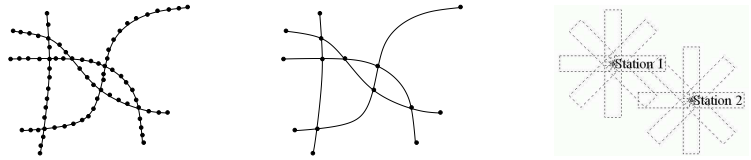

Fig. 2. *(a) Example of a metro map graph (b) Simplified metro map graph (c) Eight label positions for each vertex.*

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 with a set of lines, each being a sequence

of stations.
Output: a good layout L of G.

We now need a definition of a *good* layout of a metro map graph. For this purpose, we have studied existing *hand-drawn* metro maps from all over the world. For example, a detailed study of the London metro map by Beck can be found in [4]. From these manually produced layouts, we derive the following criteria for a good metro map layout.

C1: Each line drawn as straight as possible.
C2: No edge crossings.
C3: No overlapping of labels.
C4: Lines mostly drawn horizontally or vertically, with some at 45 degrees.
C5: Each line drawn with unique color.

We have designed layout methods based on these criteria. It should be noted that producing layouts conforming to exact geometry or topology is not the primary aim of our project. 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 for *navigation*, that is, to find out how to get to a specific 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.

3 The Layout Methods

We have tried five different layout methods using various combinations of spring algorithms. The tools that we use are GEM [2], a *modified version* of 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`: Simplify the metro map graph using a preprocessing step described in Section 3.1 and use the GEM algorithm *with* edge weight (details are explained later).
3. `Method 3`: Simplify the metro map graph and use the GEM algorithm *without* edge weight. Then we use the modified PrEd algorithm *with* edge weight.
4. `Method 4`: Simplify the metro map graph and use the GEM algorithm without edge weight. Then we use the modified PrEd algorithm with edge weight, plus *orthogonal* magnetic spring algorithm.
5. `Method 5`: Simplify the metro map graph and use the GEM algorithm without edge weight. Then we use the modified PrEd algorithm with edge weight, orthogonal magnetic spring algorithm, plus *45 degree* magnetic field forces.

We now describe each method in detail. *Method 1* simply uses GEM, a generic spring embedder. We use *Method 1* mainly as a baseline for comparison.

3.1 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, of the graph. This motivates us to remove these vertices and define a simplified graph G'. The resulting graph only contains intersection vertices and vertices with degree one. For example, the metro map graph in Figure 2 (a) can be simplified as in Figure 2 (b). Note that special care is needed to handle self loop and multiple edge cases.

After drawing the simplified graph G', we need to reinsert those removed vertices to get a layout of G. This requires space; hence we assign 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'.

3.2 Method 3

Method 3 uses 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 the PrEd algorithm to take into account 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:

$$F_x^a(u,v) = \frac{d(u,v)}{\delta(u,v)}(x(v) - x(u)),\ \ F_x^r(u,v) = \frac{-\delta(u,v)^2}{d(u,v)^2}(x(v) - x(u)) \quad (1)$$

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 the 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 in an identical manner to the original PrEd algorithm, that is:

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

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:

$$F_x(v) = \sum_{(u,v)\in E} F_x^a(u,v) + \sum_{u\in V} F_x^r(u,v) + \sum_{(a,b)\in E} F_x^e(v,(a,b)) - \sum_{u,w\in V,(v,w)\in E} F_x^e(u,(v,w)) \quad (3)$$

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 edge weights.
4. Produce a layout L of G by reinserting the removed vertices, spaced evenly along the edges in the layout L''.

3.3 Method 4

Method 4 uses the preprocessing step, GEM and PrEd with orthogonal magnetic springs, that is, with horizontal and vertical aligning 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 force from an individual force field vector on the edge connecting vertices u and v is determined by a similar calculation to that for a magnetic spring:

$$F^m(u,v) = c_m b d(u,v)^\alpha \theta^\beta \quad (4)$$

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, the total force applied to a vertex v becomes:

$$F_x(v) = \sum_{(u,v)\in E} F_x^a(u,v) + \sum_{u\in V} F_x^r(u,v) + \sum_{(a,b)\in E} F_x^e(v,(a,b)) - \sum_{u,w\in V,(v,w)\in E} F_x^e(u,(v,w)) + \sum_{(u,v)\in E} F_x^m(u,v) \quad (5)$$

Method 4 can be described similarly to Method 3, except at Step 3: Produce a better layout L'' of G' using the PrEd algorithm, modified to include edge weights and orthogonal magnetic field forces.

3.4 Method 5

Method 5 uses the preprocessing step, 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.

Method 5 can be described similarly to Method 3, except at Step 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.

Note that the production of a metro map layout which preserves geographical constraints can be achieved with a small modification. Instead of using GEM at Step 2, we can assign real geographical coordinates to vertices according to the real world latitude and longitude of their associated train stations. This ensures that the geographical embedding of the graph remains unchanged throughout the layout process. As a result, the relative ordering of edges and their crossings are preserved.

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. In this approach, 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. We define an *eight position model*, orthogonal and diagonal, for metro map labels as illustrated in Figure 2 (c). Note that the labeling of other types of maps typically uses only a four position model.

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 assigned to each feature forms a clique in the graph. Each vertex can also have a cost value indicating the preference of using its associated label position on the map.

A labeling solution is then generated by computing the *maximum independent set with minimum cost* of the conflict graph. If a vertex is included in the independent set then its associated label position is used for label placement. Features which have no label positions appearing in the set are left unlabeled. Note that the maximum independent set represents label positions that do no overlap.

We have labeled the metro maps using the `LabelHints` system with eight position model. `LabelHints` implements the automatic approach described above, and offers several other tools for interactively exploring map labeling solutions [7]. It uses a simulated annealing algorithm and a greedy heuristic for producing an initial labeling. Whenever the users are not satisfied with the computer-generated result, they can improve the solution by directly changing the conflict graph, re-executing the algorithms, and/or modifying the pre-computed independent set. Such interactions allow the users to include important domain knowledge that was not considered in the automatic process.

Through 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 layouts.

5 Implementation and Experimental Results

The layout algorithms were implemented as a plugin to `jjGraph` [3]. The tests were executed on a single processor 3.0GHz Pentium 4 machine with 1GB of RAM, and the code was run under the Sun Microsystems Java(TM) 2 Runtime Environment, Standard Edition.

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` allows 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 `LabelHints` [7].

We used real world data sets with several hundred vertices. Let $G = (V, E)$ be the original metro map graph and $G' = (V', E')$ be the reduced metro map graph. Details of the data sets are as follows. *Sydney*: $|V| = 319, |E| = 897, |V'| = 41, |E'| = 178$, *Barcelona*: $|V| = 101, |E| = 111, |V'| = 22, |E'| = 32$, *Tokyo*: $|V| = 224, |E| = 292, |V'| = 62, |E'| = 122$, *London*: $|V| = 271, |E| = 745, |V'| =$

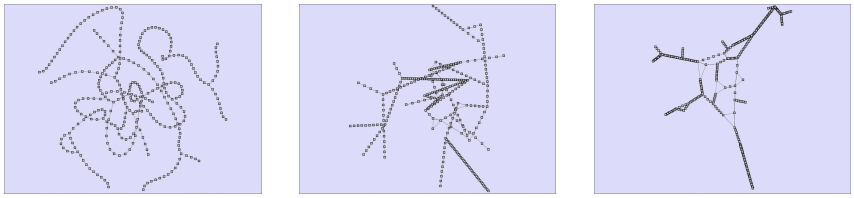

Fig. 3. *Sydney Cityrail NSW network produced by (a) method 1 (b) method 2 (c) method 3.*

92, $|E'| = 317$, *Train of thought network*: $|V| = 76, |E| = 120, |V'| = 36, |E'| = 67$, *O'Reilly book network*: $|V| = 116, |E| = 137, |V'| = 44, |E'| = 65$.

In summary, the results are comparable to hand drawn metro maps. Our method produces a good metro map layout very quickly. First, we present results of the Sydney Cityrail network. 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 are as follows. *Method 1*: 12, *Method 2*: 0.6, *Method 3*: 1.9, *Method 4*: 2.1, and *Method 5*: 2.3 seconds.

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.

Each method's results for Cityrail are illustrated in Figures 3 and 4. Note that each successive Figure improves over the previous one; and Method 5 is clearly the best. It satisfies most of the criteria that we wanted to achieve.

Since Method 5 produces the best result, we chose this layout for labeling. The results for the Barcelona, Sydney Cityrail NSW, Tokyo and London metro map layouts are shown in Figures 5 (b), 7 (a), 8 (a) and 8 (b) respectively. The running times are: *Barcelona*: 0.2, *Sydney Cityrail NSW*: 2.3, *Tokyo*: 9.2, and *London*: 22 seconds. Note that London is the most complex of these networks.

Figure 6 shows two examples of metro map metaphor visualisation; the train of thought network and the book network. Figure 7 (b) shows an example of the Sydney Cityrail NSW network with *fixed embedding*. The running times are: *Sydney*: 7.6, *Train of thought*: 3.3, and *O'Reilly book network*: 3.8 seconds.

6 Conclusion and Future Work

From our experiments using real world data sets, we have shown that carefully designed spring algorithms can produce good layouts of metro maps. Using a suitable preprocessing step and magnetic springs, we have obtained results that satisfy the given aesthetic criteria for metro maps. These results are comparable to hand drawn metro maps. We believe that automatic visualisation tools can be used as a fast preprocessing step for producing a good quality visualisation very quickly. Our current work is to extend this method to visualise more complex

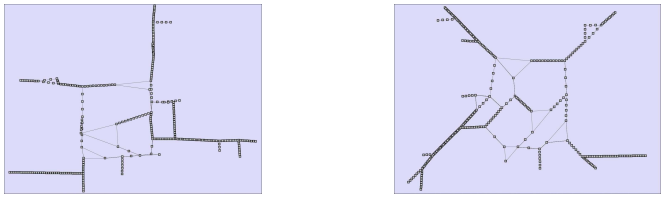

Fig. 4. *Sydney Cityrail NSW network produced by (b) method 4 (c) method 5.*

geographical networks, such as the European railway system. Furthermore, we want to apply other graph drawing approaches to the problem. For a multicriteria optimisation approach, see [10].

References

1. F. Bertault, A Force-Directed Algorithm that Preserves Edge Crossing Properties, *Graph Drawing 99*, LNCS 1731, pp. 351-358, Springer Verlag, 1999.
2. A. Frick, A. Ludwig and H. Mehldau, A Fast Adaptive Layout Algorithm for Undirected Graphs, *Graph Drawing 94*, LNCS 894, pp. 388403, Springer Verlag, 1995.
3. C. Friedrich, `jjgraph`, *personal communication.*
4. K. Garland, *Mr. Beck's Underground Map*, Capital Transport Publishing, England, 1994.
5. S.-H. Hong, D. Merrick and H. A. D. do Nascimento, The Metro Map Layout Problem, *Proc. of Australasian Symposium on Information Visualisation, (invis.au'04)*, *Conferences in Research and Practice in Information Technology*, vol 35, ACS, pp. 91-100, 2004.
6. S.-H. Hong, D. Merrick and H. A. D. do Nascimento, The Metro Map Layout Problem, Technical Report, 2004, `http://www.it.usyd.edu.au/~dmerrick/metromap/index.html`
7. H. A. D. do Nascimento and P. Eades, User Hints for Map Labelling, *Proc. of Australasian Computer Science Conference 2003*, *Conferences in Research and Practice in Information Technology*, vol 16, ACS, 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. J. M. Stott and P. Rodgers, Metro Map Layout Using Multicriteria Optimization, *Proc. of International Conference on Information Visualisation (IV04)*, pp. 355-362, 2004.
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. Cityrail Network, `http://www.cityrail.info/networkmaps/mainmap.jsp`
14. O'Reilly book network, `http://www.oreilly.de/artikel/routemap.pdf`

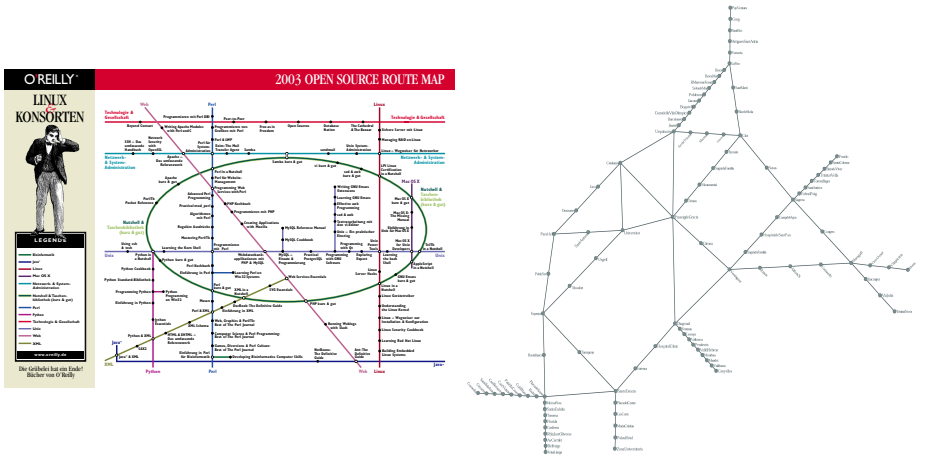

Fig. 5. *(a) O'Reilly book network (b) Barcelona city metro map with labeling.*

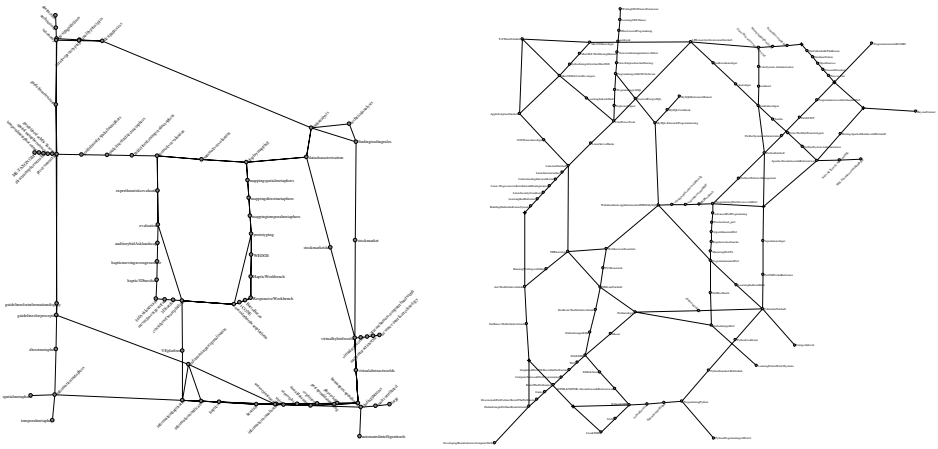

Fig. 6. *Metro map metaphors for (a) Train of thought network (b) Book network.*

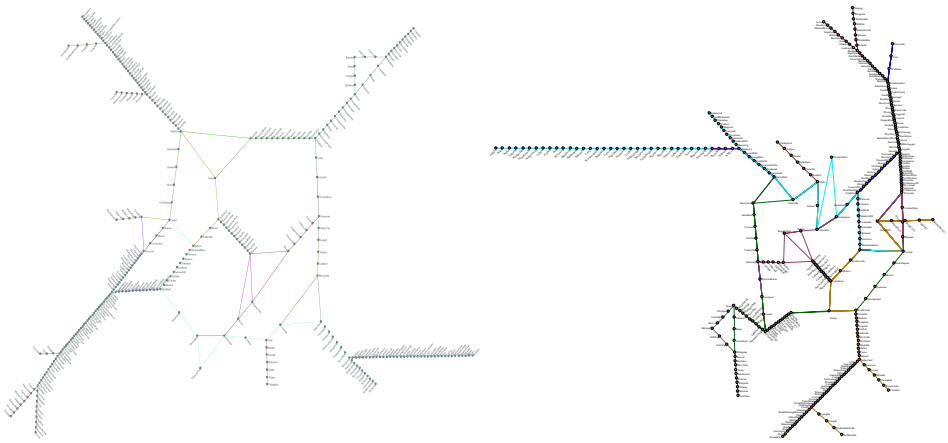

Fig. 7. *(a) Sydney Cityrail NSW network with labeling (b) Sydney Cityrail NSW network with fixed embedding.*

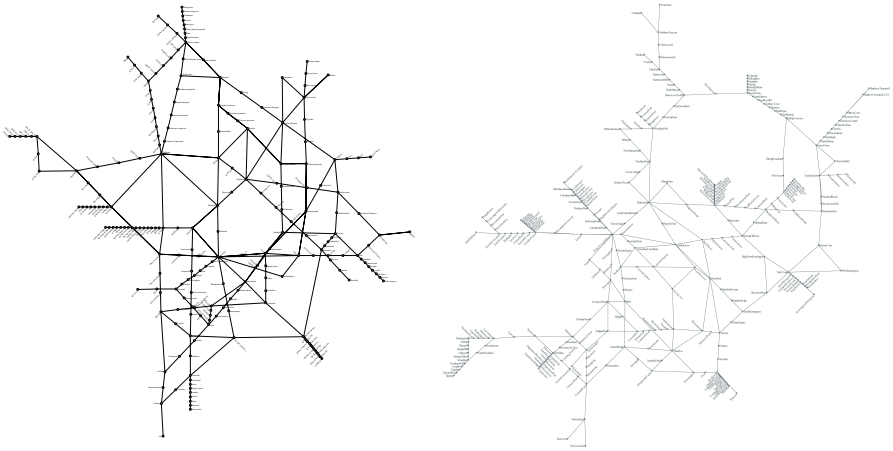

Fig. 8. *(a) Tokyo metro map with labeling (b) London metro map with labeling.*