Automatic Drawing for Tokyo Metro Map

Masahiro Onda[1], Masaki Moriguchi[2], and Keiko Imai[3]

1 **Graduate School of Science and Engineering, Chuo University**
`monda@imai-lab.ise.chuo-u.ac.jp`

2 **Meiji Institute for Advanced Study of Mathematical Sciences, Meiji University**
`moriguchi@meiji.ac.jp`

3 **Faculty of Science and Engineering, Chuo University**
`imai@ise.chuo-u.ac.jp`

Abstract

The Tokyo subway is one of the most complex networks in the world, and it is difficult to obtain its easy-to-read metro map using existing layout methods. In this paper, we present a new method that can generate complex metro maps like the Tokyo metro map. Our method consists of two phases. The first phase generates rough metro maps. It decomposes the metro networks into smaller subgraphs, and generates rough metro maps partially. In the second phase, we use a local search technique to improve the aesthetic quality of the rough metro maps. The experimental results including the Tokyo metro map are shown.

1 Introduction

Metro maps are useful tools for passengers in complex metro networks. They help us in planning the route from one station to another promptly. Metro maps give us information, such as where to change stations, which direction to go for the destination, and how many stations that are from the departure point to the goal. Passengers access the topology of the network before the exact geographical location.

Such a metro map was first created by Henry Beck [1], who was an English technical draftsman. He proposed the criteria for drawing maps that are easy to read. It is still difficult to draw them manually, even if a draftsman takes sufficient time. Therefore, the problem of drawing metro map layouts automatically (in general, we call it the *metro map layout problem*) has been investigated for a long time.

Hong *et al.* [3] presented five methods for the metro map layout problem. These methods are based on the spring-embedder paradigm, where several attracting and repelling forces act between the vertices. The forces iteratively optimize the metro map until a locally optimal equilibrium. They experimented with real-world data, and were able to produce metro maps quickly. The Tokyo metro map can be obtained by using their method. However, there are crossings of edges and labels.

Stott *et al.* [4] described a method of drawing metro maps using multi-criteria optimization based on hill climbing. They defined various metrics, which are used in a weighted sum to find a fitness value for a layout of metro maps. Their approach uses some clustering mechanisms to avoid local minima.

Nöllenburg and Wolff [2] considered the layout problem of railroad lines connecting stations and the labeling problem for the stations simultaneously using mixed-integer programming (MIP). The solvability of the problem depends on the size of the MIP. If their method can solve the problem in terms of the size of the MIP, the results obtained by their method almost satisfy Beck's criteria and they are easy to read.

34th European Workshop on Computational Geometry, Berlin, Germany, March 21–23, 2018.
This is an extended abstract of a presentation given at EuroCG'18. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

2 The Metro Map Layout Problem

This section describes the metro map layout problem based on the formulation defined by Nöllenburg and Wolff [2]. A metro network consists of metro lines and stations, and it can be seen as a planar graph. If two metro lines intersect at the stations, by adding such intersections as vertices, the network can be considered as a planar graph $G = (V, E)$. Let L be a set of metro lines. Each edge of G belongs to at least one metro line. An embedded graph of G in the plane is called a *metro map* Γ. Let $l_e > 0$ be the minimum length of $\Gamma(e)$ for each edge $e \in E$, and $d_{\min} > 0$ be the minimum distance between each pair of non-incident edges in Γ. Nöllenburg and Wolff gave seven design rules for drawing metro maps, and split these rules into hard and soft constraints. The following formulation of the metro map layout problem has been defined by Nöllenburg and Wolff [2].

(H1) For each edge e, $\Gamma(e)$ must be octilinear.
(H2) For each vertex v, the circular order of its neighbors must agree in Γ and the input embedding.
(H3) For each edge e, $\Gamma(e)$ must have a length at least l_e.
(H4) For each edge e, $\Gamma(e)$ must have a distance of at least $d_{\min} > 0$ from each non-incident edge in Γ.
(S1) The lines in L should have few bends in Γ, and the bend angle($< 180°$) should be as large as possible.
(S2) For each pair of adjacent vertices (u, v), their relative position should be preserved in Γ.
(S3) The total edge length of Γ should be small.

The objective function is the sum obtained by multiplying soft constraints by each weight. Thus the problem can be defined in [2] as follows:

Metro-Map Layout Problem

Input: a plane graph $G = (V, E)$ with maximum degree eight, metro lines L of G, minimum edge lengths $l_e > 0$, for each $e \in E$, and a minimum distance $d_{\min} > 0$.
Output: a metro map Γ that satisfies the hard constraints and optimizes the soft constraints.

The method can be applied to metro networks having station name labels as follows. They considered the label region of some stations as a quadrilateral. They added edges expressing the boundaries of the label regions into the metro network. They formulated a MIP model for this modified graph. Further, they presented some heuristics that reduce the size of the graph and the MIP model because the size of the MIP is generally large. The approach that reduces the size of the graph replaces each path of continued vertices with degree two temporarily by a path of length 3. The approach does not consider overlapping-free constraints in the initial MIP formulation. Then, during the optimization process, they add the constraints on demand, and repeat until the constraints are satisfied.

3 The Metro Map Drawing Method

If the solution can be obtained by the MIP formulation defined by Nöllenburg and Wolff [2], the results are good metro maps. However, the metro map in Tokyo is too complex and, in our experiment, their method did not succeed to generate the Tokyo metro map; our implementation of their method with CPLEX ran out of memory and terminated after running for five days without finding any feasible solution. The implementation used the size reduction techniques (simplification of degree-2 vertices and callback-based resolution of edge intersections) and ran on the system described in Section 4. Therefore, in this paper, we try to modify their method and optimize partially rather than overall. Our method has two phases: a brief incremental construction phase and an iterative improvement phase. The first incremental construction phase draws rough metro maps by MIP based on the methods in [2]. The second iterative improvement phase improves the maps generated by the incremental construction phase to make them more aesthetic. We forgo the optimum solution for the soft constraints. Instead, our method improves the metro maps that satisfy the hard constraints by a local search technique, and gives a local optimum solution.

3.1 Preprocessing

We preprocess the Tokyo metro network by adding the label regions and replace every long path with a path with three edges using an approach similar to Nöllenburg and Wolff's method. The Tokyo metro network has stations whose degree is greater than eight. However, such stations cannot use the MIP formulation in [2]. Therefore, we modify the graph such that for every vertex, its degree is less than nine. A vertex $v_{\rm in}$, whose degree is greater than two, is called an *interchange vertex* (See Figure 1 (a)). We split each interchange vertex into three vertices and connect these three vertices by horizontal line segments to reduce its degree (See Figure 1 (b)). Subsequently, the vertices that are initially adjacent to the interchange vertex $v_{\rm in}$ reconnect to one of the three new vertices by the direction of the original edge. We set the label of the interchange vertex using line segments connecting the three vertices as the interchange label region. For the rest of this paper, the graph obtained after splitting interchange vertices is described as $\widetilde{G}$

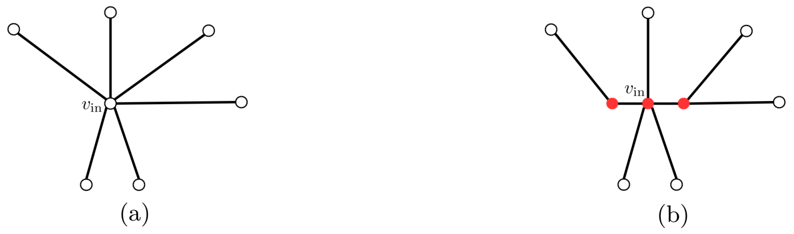


Figure 1 (a) An interchange vertex $v_{\rm in}$. (b) An interchange vertex after adding two vertices and reconnecting.

3.2 Reduced Optimization

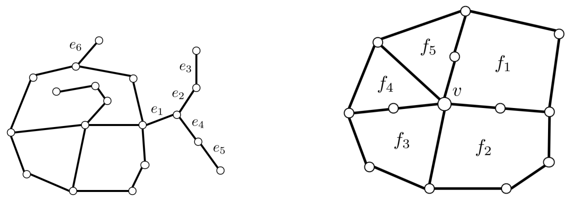
We generate metro maps by reduced optimization using MIP formulation [2]. In reduced optimization, we formulate as some edges can move and the others cannot move but these length can be adjustable. By doing so, it is easy to move edges while satisfying the constraint of overlapping-free (H4), and to generate good metro maps.

3.3 Incremental Construction

The incremental construction method presented in this section generates metro maps that satisfy the hard constraints.

Division method

It is difficult to generate a metro map from a large input graph even if we use Nöllenburg and Wolff's method. Therefore, our method decomposes the input graph into smaller subgraphs, external trees, and faces. We define a *external edge* as an edge incident with only non-boundary face, and an *external tree* T as a tree consisting of connected external edges. The trees consisting of the edges $\{e_1, e_2, e_3, e_4, e_5\}$ and $\{e_6\}$ in Figure 2(a) are external trees. For each face f of G, there is a subgraph that forms a boundary of f. We call it the *facial subgraph* of f. Next, for a vertex v incident with an inner face, we define its *facial neighborhood* $N(v)$ as the union of the facial subgraphs of all the *inner faces* incident with v. Note that the union of graphs $G_1 = (V_1, E_1), G_2 = (V_2, E_2), ..., G_k = (V_k, E_k)$ is defined as the graph whose vertex set is $\cup_{i=1}^{k} V_i$ and whose edge set is $\cup_{i=1}^{k} E_i$. See Figure 2(b). We sort $\{N(v)\}$ in terms of the degree of the vertex v in G in descending order, and this sorted list is described as $\mathcal{N} = \{N(v_1), N(v_2), ..., N(v_m)\}$.



(a) External trees are $\{e_1, e_2, e_3, e_4, e_5\}$ and $\{e_6\}$.

(b) A facial neighborhood of v, and facial subgraphs of f_1, f_2, f_3, f_4 and f_5.

Figure 2 Division method in the incremental construction method.

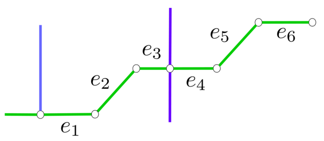


Figure 3 An interchange path that is defined as a set of edges that is from a non-degree 2 vertex to another non-degree 2 vertex.

The Incremental Construction Algorithm

Suppose that all external trees and the sorted list $\mathcal{N} = \{N(v_1), N(v_2), ..., N(v_m)\}$ are known. For this input data, the incremental construction algorithm treats subgraphs in order of the facial neighborhoods and external trees. When we generate a rough metro map by reduced optimization, we use the MIP formulation and heuristics from [2]. To reduce the size of the input graph, all the paths between two interchange vertices and external trees $\{T_1, T_2, ..., T_l\}$ in the graph $\widetilde{G}$ are simplified, and they consist of at most three edges. Note that the paths in $N(v_i)$ are also simplified.

First, we treat the facial neighborhoods in order of the sorted list. Then, we treat the external trees one at a time.

Step 1

Set G_0 to an empty set.

For each $i = 1, ..., m$:

Add $N(v_i) = (V_{N(v_i)}, E_{N(v_i)})$ to G_{i-1} and the obtained graph is called $G_i = (V_i, E_i)$.

That means $V_i = V_{i-1} \cup V_{N(v_i)}$ and $E_i = E_{i-1} \cup E_{N(v_i)}$.

However, if all vertices and edges in $N(v_i)$ have already been added to G_{i-1}, $N(v_i)$ is not added to G_{i-1}, and G_{i-1} becomes G_i.

Then, we solve the MIP for G_i by fixing the direction of the edge in Γ_{i-1}, and obtain the new map Γ.

Step 2

For each $j = 1, ..., l$:

Add T_j to G_{m+j-1} and obtain a graph G_{m+j}.

Solve the MIP for G_{m+j} by fixing the direction of the edge in Γ_{m+j-1}, and obtain the new map Γ_{m+j}.

After step 2, the map Γ satisfies all the hard constraints and this can be used as the input of the next iterative improvement method. By generating metro maps from facial neighborhood of the vertex with large degree, it is possible to generate metro maps satisfying the constraint of cyclic ordering (H2).

3.4 Iterative Improvement

An iterative improvement method improves the metro map generated by the incremental construction algorithm to obtain a local optimal solution. We decompose a metro map into paths. In Γ, an *interchange path* is defined as a set of edges that are from a non-degree 2 vertex to another non-degree 2 vertex. For example, the edges e_1, e_2, and e_3, or the edges e_4, e_5, and e_6 in Figure 3.

In the iterative improvement method, for each interchange path we generate metro maps by reduced optimization. Thus, an iterative improvement method generates optimal metro maps so that only each interchange path can move direction.

Because the original objective function is linear with respect to the variables representing edge lengths, it does not penalize non-uniform edge lengths. Therefore, the results might be poorly balanced. To prevent this, we add the following soft constraint:

(S4) For each degree 2 vertex v whose adjacent vertices u, w do not have degree 2, the difference between the lengths of $\Gamma(uv)$ and $\Gamma(vw)$ should be as small as possible.

Accordingly, we add the following cost function to the objective function

$$\sum_{v\in V,\deg(v)=2,\deg(u)\neq 2,\deg(w)\neq 2} |\lambda(uv) - \lambda(vw)|,$$

where $\deg(v)$ is the degree of vertex v and $\lambda(uv)$ is the length of edge uv. This cost function, proposed in [4], is linear and explicitly penalizes non-uniform edge lengths.

4 Experimental Results

We applied our methods to the Tokyo metro network that has 434 vertices and 531 edges. We solved the MIP with the optimizer CPLEX 12.7.1 running on a computer with a 6-core Intel Xeon 3.60 GHz processor, and 128 GB RAM.

The Tokyo metro map drawn using our methods is shown in Figure 4. The weights used in the objective function were as following: in the incremental construction method, (S1) is 1, (S2) is 30, and (S3) is 1, and in the iterative improvement method, (S1) is 1, (S2) is 20, (S3) is 5, and (S4) is 1. To speed up the computation, we experimentally increased the weight of relative position (S2). Conversely, increasing the weight of the edge length (S3) tends to increase the calculation time. The layout was mostly generated within 5 hours: the incremental construction method took 13 minutes, and the iterative improvement method took 5 hours.

5 Conclusion

We have proposed an incremental construction method and an iterative improvement method that improves metro maps generated by the incremental construction method. The former

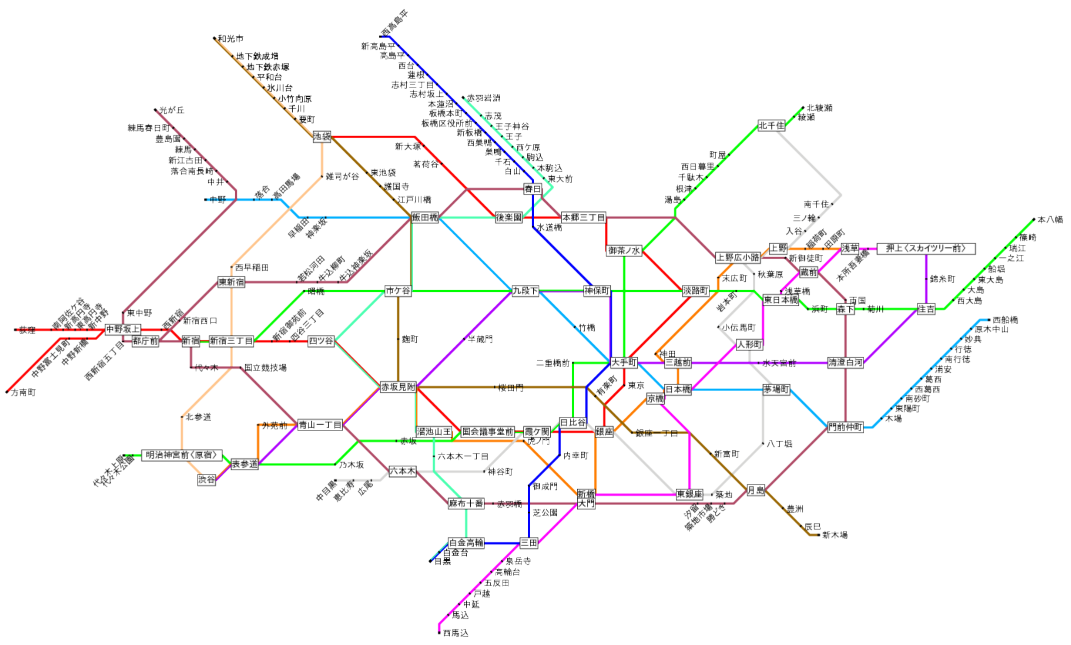


Figure 4 Layout of Tokyo railway network produced by our approach.

can generate complex metro maps that satisfy hard constraints quickly, because we decompose the input graph into subgraphs and generate a rough metro map partially. The latter improves metro maps and arranges vertices in a balanced position. Our methods can generate complex metro maps such as the Tokyo metro map that cannot be directly produced by the approach of Nöllenburg and Wolff [2].

One limitation of the iterative improvement method is that it is slow. We plan to reveal the cause for why phase 2 takes so much longer than phase 1. Another limitation is that there are some artifacts around terminal stations and dummy crossings in the results. We also plan to improve it using a post-processing.

Acknowledgments. We would like to thank the anonymous reviewers for their valuable and constructive comments. This work was supported by JSPS KAKENHI Grant Numbers 16K00024, 16H01728, and 17K14580.

References

1 K. Garland. *Mr Beck's Underground Map.* Capital Transport, 1994.

2 M. Nöllenburg and A. Wolff. Drawing and labeling high-quality metro maps by mixed-integer programming. *IEEE Transactions on Visualization and Computer Graphics*, 17(5):626–641, 2011.

3 S.-H. Hong, D. Merrick, and H. A. D. do Nascimento. Automatic visualization of metro maps. *J. Visual Languages and Computing*, 17(3):203–224, 2006.

4 J. Stott, P. Rodgers, J. C. Martinez-Ovando, and S. G. Walker. Automatic metro map layout using multicriteria optimization. *IEEE Transactions on Visualization and Computer Graphics*, 17(1):101–114, 2011.