Interactive Metro Map Editing

Yu-Shuen Wang and Wan-Yu Peng

Abstract—Manual editing of a metro map is essential because many aesthetic and readability demands in map generation cannot be achieved by using a fully automatic method. In addition, a metro map should be updated when new metro lines are developed in a city. Considering that manually designing a metro map is time-consuming and requires expert skills, we present an interactive editing system that considers human knowledge and adjusts the layout to make it consistent with user expectations. In other words, only a few stations are controlled and the remaining stations are relocated by our system. Our system supports both curvilinear and octilinear layouts when creating metro maps. It solves an optimization problem, in which even spaces, route straightness, and maximum included angles at junctions are considered to obtain a curvilinear result. The system then rotates each edge to extend either vertically, horizontally, or diagonally while approximating the station positions provided by users to generate an octilinear layout. Experimental results, quantitative and qualitative evaluations, and user studies show that our editing system is easy to use and allows even non-professionals to design a metro map.

Index Terms—Metro Map, Interactive editing, octilinear, least squares optimization

1 Introduction

Metro maps have become important tools as the number of metro lines in cities increases. These maps are typically used by passengers to orient themselves in a complex traffic network. Unlike general maps, metro maps display network topology rather than exact geography. Such information is helpful to passengers, who instantly need to know which station to get on or off a train. Therefore, most metro maps worldwide have *octilinear* layouts, which have been used at least since 1931, when Beck [1] introduced his famous map of London. Typically, an octilinear layout has: 1) evenly spaced nodes, 2) straight routes, and 3) octilinear edge directions (i.e., vertical, horizontal, or diagonal). Distorted station positions, which are only for the purpose of locality recognition, provide freedom to achieve clear navigation.

When a metro network is developed in a city, cartographers have to redesign the map for passengers. However, designing a metro map is time-consuming and requires expert skills. Although several automatic layout algorithms have been presented to reduce the burden of cartographers, these algorithms may not satisfy aesthetic and readability requirements. Given that fully automatic methods are difficult to control, cartographers who attempt to modify a metro map still have to manipulate most station positions. Consequently, an interactive interface that allows cartographers or even non-professionals to easily design a metro map is essential (Figure 1).

• *Y. S. Wang and W. Y. Peng are with the Department of Computer Science, National Chiao Tung University, Taiwan*
E-mail: yushuen@cs.nctu.edu.tw
E-mail: u11na.am96@g2.nctu.edu.tw

One of the main challenges of editing an octilinear metro map is performance. The layout of a metro map can be considered to be a graph drawing problem in which connecting edges are provided and only node positions are unknown. However, the involved octilinear constraint results in high computational cost. On the one hand, discrete edge directions cause the system to search for the optimal solution in a highly concave feasible region if the objective function is defined in a continuous domain [2]. On the other hand, preventing edge intersections is NP-hard if the constraints have a discrete formulation [3], [4]. The aforementioned techniques are not interactive neither consider all layout aspects and thus, they are incapable of adapting to metro map editing. A recent work [5] achieved real-time performance by first computing a curvilinear layout and then simply rotating each edge to the closest octilinear direction. However, this method does not consider node positions. In Figure 2.a, the pink edges highlighted by a dashed rectangle can only become horizontal because their angles are close to zero. Given that the edges do not have a vertical component, expecting the terminal station of this route to be located at the required position is impossible. Because satisfying positional constraints is essential in a map editing system, adapting their approach by a simple extension is inadequate, as demonstrated in Figure 2 and the accompanying video.

We present an interactive editing system for users to manipulate a metro map by controlling a small number of nodes. We call these nodes *handles*. That is, given a set of handle positions, our system optimizes an objective function to satisfy metro map requirements and positional constraints simultaneously. Considering different style and aesthetic concerns, our system supports both curvilinear and octilinear layouts when creating metro maps. However, we focus

Fig. 1. From left to right are the geographic, curvilinear and octilinear metro maps of Singapore. We provide users with an intuitive interface to edit the map. The criteria for a visually pleasing result, such as evenly spaced stations, smooth/straight transportation routes, and maximal included angles at junctions, are satisfied.

on the octilinear layout because this layout is widely used. To achieve interactive performance, the solver neglects the octilinear constraints initially and then rotates each edge to extend either vertically, horizontally, or diagonally. To produce a visually pleasing result, edge lengths and edge directions should match well. However, we solve for these variables independently and alternatively to achieve interactive performance. That is, we rotate each edge to the optimal direction while conserving the net rotation to retain the vertical and horizontal components of each curvilinear line. Afterward, we solve for edge lengths to approximate the required handle positions while maintaining map topology and preventing unwanted line intersections. The interactive performance and the approximation of handle positions provide an intuitive editing interface.

To the best of our knowledge, this paper presents the first metro map editing system. Users can control a small number of nodes to manipulate a metro map. The remaining nodes are then automatically relocated. The presented interface allows users to edit a region of the map, including translations, rotations, and scaling. This feature is particularly useful when the region demands an identical transformation. Our system can dynamically consider the human knowledge and adjust the layout to make it consistent with user expectations. The experimental results shown in Figures 1, 8, and 10, the accompanying video, the quantitative and qualitative evaluations, and the user studies demonstrate the feasibility of our technique.

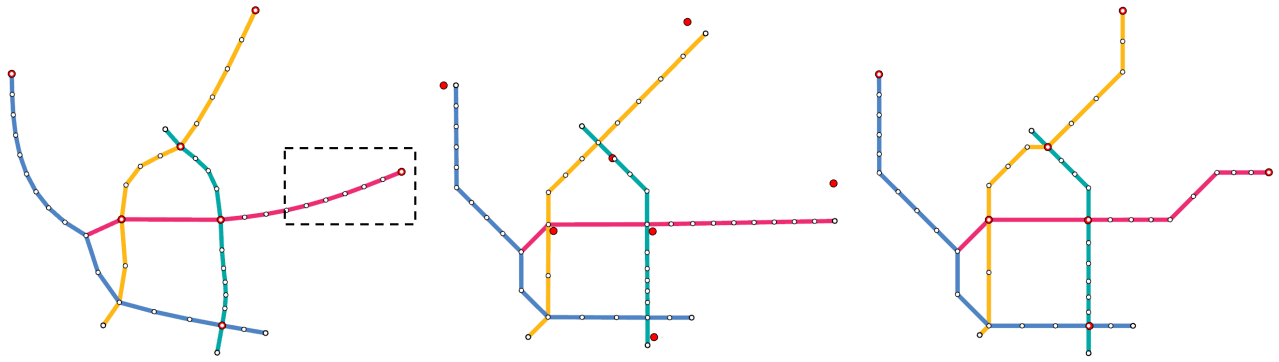
2 Related Work

Metro map design. Octilinear layouts have been widely used in metro network navigation since Beck introduced the London underground map in 1931 [1]. The map aims to provide a graph topology to help users travel by themselves when transferring from one train to another [6]. Designing metro maps is time consuming and requires professionals; thus, many algorithms have been presented to compute map layout automatically. Avelar and Müller [7] presented a transformation method to place stations. They applied simple geometric operations and tests to preserve topology. Merrick and Gudmundsson [8] simplified a polygonal path while constraining edges that are parallel to certain orientations. However, such simplification may change route topology and lead to misunderstandings.

Several methods define geometric and aesthetic requirements by energy terms and apply an optimization technique to solve for the layout. Hong et al. [9] applied the magnetic spring algorithm to iteratively update node positions with a cooling schedule until the system converges. Stott et al. [2] presented a hill climbing optimizer that combines node clustering and movement to improve fitness and achieve fast convergence. Nöllenburg and Wolff [3] applied mixed-integer programming to address map layout and station names, and produce appealing results. Wang and Chi [5] solved a quadratic optimization to obtain a curvilinear map layout, and then they rotated each edge to the closest octilinear direction to achieve interactive performance. However, all the aforementioned techniques fail to achieve metro map editing because they either cannot achieve interactive performance or ignore the handle positions.

In addition to octilinear layouts, some techniques promote the use of a curvilinear drawing style. Fink et al. [10] represented metro lines by using Bézier curves. The force-directed method was applied to prevent visually disruptive bending of metro lines when control points are determined. Roberts et al. [11] measured the usability of curvilinear and octilinear layouts both objectively and subjectively. The conducted user studies showed that no rule set can be regarded as a gold standard and that design rules should match the properties of a network.

Metro tourist map. Recent methods were presented to combine metro maps with points of interest (POI). These maps highlight parks, museums, and shopping



(a) (b) (c)

Fig. 2. The edited metro map layouts of Lisbon. In this example, the handles are highlighted by large red circles. (a) The curvilinear layout is initially computed without considering octilinear constraints. (b) Given that the closest octilinear direction and the length of each edge in the curvilinear layout are independent of handle positions [5], adapting this method does not satisfy the objective of editing. (c) By contrast, our optimized edge directions and edge lengths match well with each other. The handles are located at user-specified positions.

malls that tourists may want to visit. Wu et al. [12] represented each POI by a photo and elongated the travel route along the centerline of a map. When linking stations and photos, their method minimizes the number of intersections between metro lines and connecting lines to enhance visual clarity. These researchers also ensured the placement of large annotation labels close to the corresponding stations to prevent occluding metro lines [13]. Given that many POI can exist in a city, the visualization easily results in visual clutter. To address this issue, Claudio and Yoon [14] applied hierarchical clustering to display a portion of the POI collection at different levels. The holistic visualization allows users to plan their trip easily. Although our system is developed for interactive metro map editing, the presented layout optimization can potentially benefit these applications.

Constrained optimization. The presented method is mainly based on least squares optimization with constraints, the details of which can be obtained in [15], [16]. Some geographic network visualization methods [17], [18] also applied this technique to edit the layout of a map.

3 Interactive Metro Map Editing

We provide a graphical interface that allows users to edit a metro map by manipulating handle positions. Given a geographic metro map, wherein the position of a station is specified by longitude and latitude, our objective is to achieve a visually pleasing layout while enforcing each handle to be located at a user-specified position. That is, 1) nodes are evenly spaced, 2) included angles formed by neighboring edges are maximized, and 3) lines that do not allow users to switch trains are intersection free. In addition to providing an appealing metro map layout, manipulation should be intuitive. Therefore, the design guidelines are as follows:

- Interactive: The system must perform in realtime.
- Intelligent: The determined layouts are consistent with user expectations.
- Stable: Irrelevant routes should remain constant when a part of the map is being edited.
- Effective: Few interaction steps and short manipulation time are required to create a metro map.

To fulfill the guidelines, we formulate the requirements into energy terms and minimize the objective function to obtain the result. In the following, we denote by $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ the map layout, where $\mathbf{V} = \{\mathbf{v}_0, \mathbf{v}_1, ...\mathbf{v}_{n-1}\}$, $\mathbf{v}_i \in \mathcal{R}^2$ is the geographic node position, n is the total number of nodes, and $\mathbf{E}$ is the set of graph edges. Users can control the handle positions $\mathbf{h} \in \mathbf{H}$ and then our system determines the node positions $\mathbf{v}' \in \mathbf{V}'$ in the octilinear map layout. Given that edges are expected to only extend vertically, horizontally, or diagonally, we limit the angle of edge $\{i, j\}$ to be $\phi_{ij} = \frac{\pi}{4}x$, where $x \in \mathbf{Z}$. In addition, to prevent irrelevant edges from crossing, we keep the distance between an arbitrary node k and an arbitrary edge $\{i, j\}$ larger than the threshold ε. That is, we minimize

$$\begin{aligned} D = D_d + D_a, \quad \text{subject to} \\ \mathbf{v}'_i = \mathbf{h}_i, \quad \mathbf{h}_i \in \mathbf{H}, \\ \phi_{ij} = \frac{\pi}{4}x, \quad |\mathbf{v}'_k - \mathbf{p}_{ij(k)}| > \varepsilon, \quad \{i,j\} \in \mathbf{E}, \end{aligned} \tag{1}$$

where

$$D_d = \sum_{\{i,j\} \in \mathbf{E}} \left|(\mathbf{v}'_i - \mathbf{v}'_j) - s_{ij}\mathbf{R}_{ij}(\mathbf{v}_i - \mathbf{v}_j)\right|^2,$$

$$D_a = \sum_{\mathbf{v}' \in \mathbf{V}'} \sum_{\{j,k\} \in \mathbf{N}(i)} \left|\mathbf{v}'_i - (\mathbf{v}'_j + \mathbf{u}'_{jk} + \tan(\frac{\pi - \theta_i}{2})\mathbf{u}'_{jk})\right|^2,$$

and $\mathbf{p}_{ij(k)}$ is the point on edge $\{i, j\}$ closest to node k. The energy term D_d constrains edge lengths, where s_{ij} and $\mathbf{R}_{ij}$ are the unknown scale factor and rotation

matrix of edge $\{i, j\}$, respectively. The term D_a maximizes the included angles of neighboring edges to enforce metro lines to extend in different directions and to achieve straightness, where $\mathbf{N}(i)$ denotes the neighbors of node i, $\theta_i = \frac{2\pi}{|\mathbf{N}(i)|}$, and $\mathbf{u}'_{jk} = \frac{1}{2}(\mathbf{v}'_k - \mathbf{v}'_j)$. We refer readers to [5] for additional details because the formulations used are similar.

Considering that octilinear constraints are highly non-linear, minimizing such an objective function typically entails expensive cost. An effective method to achieve interactive performance is to compute a curvilinear layout by minimizing D without considering $\phi_{ij} = \frac{\pi}{4}x$ (Figure 2.a), and then to rotate each edge to an octilinear direction. This strategy separates the constraints defined by discrete and continuous variables, and solves for the map layout in two steps. However, as demonstrated in Figure 2.b, simply rotating each edge to the closest octilinear direction cannot approximate handle positions. This problem results in a highly unstable layout when handles are manipulated, which makes metro map editing challenging. Readers can refer to the accompanying video for the example of such an editing problem because unstable layouts are difficult to visualize in still images.

The main improvement of the presented work to [5] is the optimization technique used to minimize the objective function (Equation 1). The goal is to create an octilinear metro map that can approximate the demanded handle positions (Figure 2.c). Considering that a curvilinear layout is close to the optimal solution, our system also neglects octilinear constraints at the beginning. However, it does not simply rotate each edge to the closest octilinear direction afterward because this local property is unrelated to handle positions. Instead, we adopt a graduate solution that can also satisfy constraints such as line straightness and rotation conservation (Section 3.2.1). The former reduces the number of bends in a layout. The latter maintains the overall shape of each curvilinear line when the edges on it are rotated. Once the edge directions are obtained, our system solves for the edge lengths while preserving map topology and approximating the required handle positions (Section 3.2.2). Compared with naïve rotation, this strategy considers user inputs and enables an intuitive editing interface.

3.1 Curvilinear layout optimization

Our system solves for a curvilinear layout by minimizing Equation 1 without limiting each edge angle to have a $\frac{\pi}{4}x$ degree. To gain additional freedom, we allow edges to be lengthened or shortened when handles are manipulated. Specifically, let a *line* be a map structure, where nodes on it have one or two neighbors. Our system expects only the edges on the same line to have identical lengths. That is, s_{ij} is considered to be an unknown variable and will be iteratively updated during optimization. This unknown variable is initially set to $\frac{\rho_m}{|\mathbf{v}_i - \mathbf{v}_j|}$ and becomes $\frac{\rho_b}{|\mathbf{v}_i - \mathbf{v}_j|}$ whenever the layout is updated, where ρ_m and ρ_b are the average edge lengths of the map and the line, respectively.

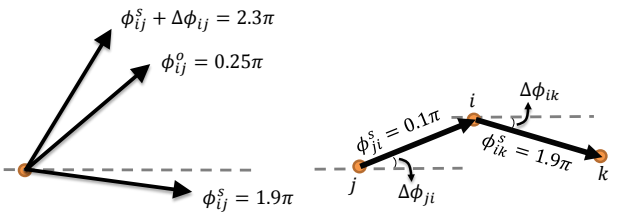


Fig. 3. (Left) We update the octilinear angle ϕ^o_{ij} from 0.25π to 2.25π if $\phi^o_{ij} - \phi^s_{ij} < -\pi$ to obtain a small rotation angle $|\Delta\phi_{ij}|$. (Right) Similarly, given that the two edges have similar directions, the rotation angles $\Delta\phi_{ji}$ and $\Delta\phi_{ik}$ should be small. We add 2π to $\phi^s_{ji} - \phi^s_{ik}$ and set this updated value in our optimization system.

3.2 Octilinear layout optimization

3.2.1 Edge direction

We rotate each edge to satisfy positional constraints while achieving straightness and octilinear edge directions. That is, 1) edge angles are equal to the $\frac{\pi}{4}x$ degree, 2) neighboring edges have similar angles, and 3) the layout of each route is similar to its curvilinear version. We solve the problem by using an iterative procedure because the aforementioned constraints are non-linear. Denote by ϕ^s_{ij} and ϕ^o_{ij} the angles of edge $\{i, j\}$ on the curvilinear and octilinear layouts, respectively; and by $\mathbf{B}$ the set of lines, wherein each node on a line has only one or two neighbors. We formulate the criteria into energy terms.

Octilinear direction. The essential requirement of an octilinear layout is maintaining edges to extend in the vertical, horizontal, or diagonal directions. This property provides simple and orderly topological representations to enable passengers to navigate by themselves. To achieve this requirement, we rotate each edge $\{i, j\}$ by the angle $\Delta\phi_{ij}$ to an octilinear direction. Specifically, we minimize

$$\Phi_o = \sum_{\{i,j\} \in \mathbf{E}} \alpha_{ij} \left| (\phi^s_{ij} + \Delta\phi_{ij}) - \phi^o_{ij} \right|^2, \quad (2)$$

where ϕ^o_{ij} is the closest octilinear angle to $\phi^s_{ij} + \Delta\phi_{ij}$, and α_{ij} is a weight that controls the proximity of the angle to ϕ^o_{ij}. We will explain how α_{ij} is determined in a later section. Note that the edge can rotate either clockwise or counterclockwise to satisfy the octilinear constraint. We choose the direction wherein $|\Delta\phi_{ij}| < \pi$. For the example shown in Figure 3 (left), ϕ^o_{ij} can be either 0.25π or 2.25π. We set $\phi^o_{ij} = 2.25\pi$ in our system.

Straightness. Each metro line should be straight to allow users to trace stations without frequently

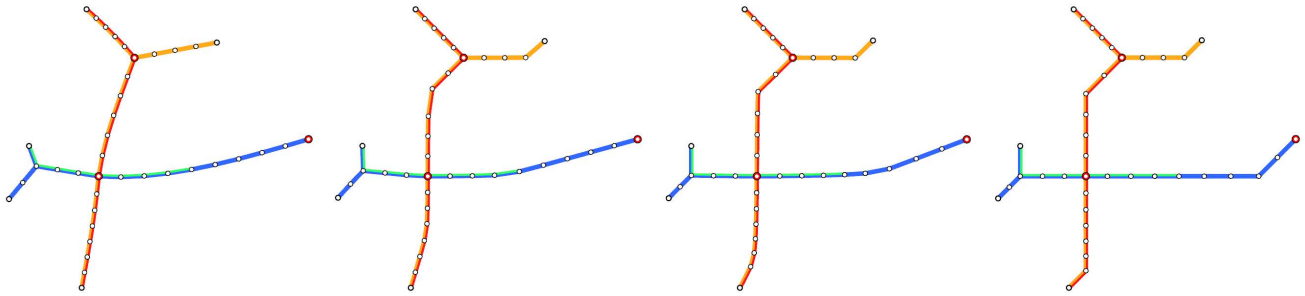


Fig. 4. From left to right are the curvilinear layout and the updated octilinear layouts after 3, 5, and 15 iterations. We highlight the handles with large red nodes to indicate user specification. As shown in the figure, the edges close to junction nodes become octilinear first. The remaining edges are then rotated inversely by our rotation conservation to retain the vertical and horizontal components. The octilinear constraint becomes increasingly hard as the iteration number increases, and an octilinear layout is obtained. In this example, we compute edge lengths whenever edge directions are updated. However, edge lengths are optimized when all edge directions become octilinear in our actual implementation.

switching attention when reading the map. Consequently, our system ensures that neighboring edges will have similar directions. This constraint is not added to the edges adjacent to a junction because the intersecting metro routes are expected to extend in different directions. Recall that $\mathbf{N}(i)$ denotes the neighbors of node i and let $\{j,k\} \in \mathbf{N}(i)$. We obtain straight lines by minimizing

$$\begin{aligned}\Phi_s &= \sum_{|\mathbf{N}(i)|=2} \left|(\phi^s_{ji} + \Delta\phi_{ji}) - (\phi^s_{ik} + \Delta\phi_{ik})\right|^2 \\ &= \sum_{|\mathbf{N}(i)|=2} \left|(\phi^s_{ji} - \phi^s_{ik}) + (\Delta\phi_{ji} - \Delta\phi_{ik})\right|^2. \end{aligned} \quad (3)$$

Similarly, Figure 3 (right) illustrates that the numerical value of $\phi^s_{ji} - \phi^s_{ik}$ is not close to 0, although the two edges have similar directions. We set $\phi^s_{ji} - \phi^s_{ik}$ to $\min(|\phi^s_{ji} - \phi^s_{ik}|, |\phi^s_{ji} - \phi^s_{ik} \pm 2\pi|)$ to prevent the angle discontinuity at 2π.

Rotation conservation. In addition to octilinear directions and straight lines, the vertical and horizontal components of a curvilinear line should be maintained when edges on the line are rotated. Otherwise, positional constraints will never be achieved regardless of edge lengths, as illustrated in Figure 2. Given that our objective is to approximate the octilinear layout to its curvilinear version, our system attempts to conserve rotation when computing for octilinear edge angles. That is, when some edges on a line rotate clockwise, others will rotate counterclockwise, and the sum of edge rotation is expected to approximate zero. This strategy prevents the global shape of an octilinear layout from deviating from that of its curvilinear version. Let $\mathbf{B}_k$ be the set of edges on the k^{th} line, we introduce the term

$$\Phi_p = \sum_{\mathbf{B}_k} \left|\sum_{\{i,j\}\in\mathbf{B}_k} \Delta\phi_{ij}\right|^2. \quad (4)$$

Optimization details. We minimize the integrated energy terms $\Phi = \alpha_o\Phi_o + \alpha_s\Phi_s + \alpha_p\Phi_p$ to solve for the edge angles. We set $\alpha_o = 1$, $\alpha_s = 1$, and $\alpha_p = 10$ in our system. Given that $\Delta\phi_{ij}$ and ϕ^o_{ij} in Equation 2 are unknown and correlated, the objective function is non-linear and may contain many local minimums. Accordingly, we can only iteratively update these two variables to prevent the solution from quickly falling into a local minimum. Specifically, we assume $\Delta\phi_{ij} = 0$ in the first step and determine the closest octilinear edge angle ϕ^o_{ij}. Afterward, ϕ^o_{ij} is considered to be a known variable and $\Delta\phi_{ij}$ is solved by a linear system. We then apply this new $\Delta\phi_{ij}$ to update ϕ^o_{ij} in the next iteration. Our system computes the two variables alternatively until the system converges.

Intuitively, α_o should be large to ensure an octilinear layout. However, because $\Delta\phi_{ij}$ and ϕ^o_{ij} are unknown, the solution will immediately fall into a local minimum if the initial guess of ϕ^o_{ij} is imperfect. To prevent this problem, we first approach the octilinear edge directions gently and allow the other constraints to be satisfied. Then, the force of this octilinear constraint strengthens gradually to fulfill the requirements. Considering that edges close to a junction mainly determine the layout of a map, we control the optimization by updating the local weight α_{ij} (Equation 2) instead of the global one α_o. That is, we set a large initial value for α_{ij} if the number of hops $d_{ij} \geq 0$ from edge $\{i,j\}$ to the closest junction is small. All weights α_{ij} increase as the number of iteration increases. Under this setting, the edges close to a junction will become octilinear first and those far from a junction will be rotated in the opposite direction to conserve rotation and thus, the shape of each route can be retained (Figure 4). In other words, the closer the edges are to a junction, the fewer iterations are required for the edge to become octilinear. Let $r > 0$ be the iteration number, we set $\alpha_{ij} = r + 5 \times \max(r - d_{ij}, 0.1)$ during optimization.

We transform the criteria for an octilinear layout into energy terms and compute for the solution by

minimizing the objective function in a least square sense. Although α_{ij} increases and will become large eventually, the obtained edge angles are very close but not exactly equal to $\frac{\pi}{4}x$. To ensure that the layout is octilinear, we simply set the edge angles to ϕ^o_{ij} after optimization.

3.2.2 Edge length optimization

When edge directions are optimized, we compute for edge lengths to approximate handle positions. During this step, edge directions are considered to be known variables and only edge lengths are determined. Let c be the center node. A handle position can then be represented by using the center node position $\mathbf{v}_c$ and a set of vectors on the path from node c to the handle. To reduce the complexity of this representation and improve performance, we determine the center node c with the smallest number of hops to the other nodes. To achieve evenly spaced nodes, we also expect equal neighboring edge lengths. Considering that metro maps typically have loops, wherein nodes have more than one path connected to center node c, we constrain the paths to reach an identical position to prevent disrupting the map topology.

Positional constraints. Our system expects each handle to be located at the specified position $\mathbf{h}_k$ to achieve intuitive manipulation. Let $\mathbf{E}_{c\to k}$ be the set of edges on the path from node c to node k, ω be the unknown edge length, and $\mathbf{x}$ be the unit length vector with the direction obtained from Section 3.2.1. We approximate the handle position $\mathbf{h}_k$ by minimizing

$$\Omega_p = \sum_{\mathbf{h}_k \in \mathbf{H}} \left| \left(\mathbf{v}_c + \textstyle\sum_{\{i,j\}\in \mathbf{E}_{c\to k}} \omega_{ij}\mathbf{x}_{ij} \right) - \mathbf{h}_k \right|^2. \quad (5)$$

We also expect non-handle nodes to be located close to their original positions because the curvilinear layout provides a good reference for achieving a high quality metro map. The applied constraint is similar to Ω_p but the handles are replaced by non-handles. That is,

$$\Omega_g = \sum_{\mathbf{v}_k \in \mathbf{V}\setminus\mathbf{H}} \left| \left(\mathbf{v}_c + \textstyle\sum_{\{i,j\}\in \mathbf{E}_{c\to k}} \omega_{ij}\mathbf{x}_{ij} \right) - \mathbf{v}_k \right|^2. \quad (6)$$

Equal edge lengths. A metro map with evenly spaced nodes has an orderly and aesthetic appearance. This kind of map also allows users to trace stations without frequently switching attention. Hence, let $\{j,k\} \in \mathbf{N}(i)$, we present the term

$$\Omega_e = \sum_{|\mathbf{N}(i)|=2 \wedge \phi^o_{ji}=\phi^o_{ik}} \left| \omega_{ji} - \omega_{ik} \right|^2 \quad (7)$$

to achieve similar lengths among neighboring edges.

Topology preservation. Most metro maps have loops with nodes that have more than one path connected to the center node. To maintain topology during metro map editing, our system enforces two incident paths to arrive at the same position. That is, by limiting each node to have only one shortest path

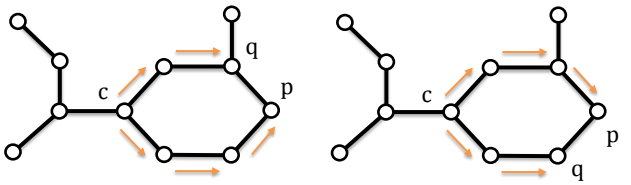


Fig. 5. Let c be the center node of a metro map. Suppose that each node toward center node c has only one shortest path. Then, an edge $\{p,q\}$ must exist on the loop, wherein both the shortest paths from node c to node p and those from node c to node q do not pass the edge. The left and right images illustrate the two possible cases. Both cases are managed well using our framework.

departing from center node c, an edge $\{p,q\}$ must exist on the loop, wherein both the shortest paths from node c to node p and those from node c to node q do not pass the edge. Our system constrains the former path plus vector $\mathbf{v}_q - \mathbf{v}_p$ to arrive at the end position of the latter path, as illustrated in Figure 5. Let $\mathbf{L}$ denote the set of edges $\{p,q\}$ that do not belong to the shortest path tree of c, and $\mathbf{E}_p$ and $\mathbf{E}_q$ be the sets of edges on the paths to $\mathbf{p}$ and $\mathbf{q}$, respectively. To maintain graph topology, for each $\{p,q\} \in \mathbf{L}$, we constrain

$$\sum_{\{i,j\}\in \mathbf{E}_{c\to p}} \omega_{ij}\mathbf{x}_{ij} + (\mathbf{v}_q - \mathbf{v}_p) = \sum_{\{i,j\}\in \mathbf{E}_{c\to q}} \omega_{ij}\mathbf{x}_{ij}. \quad (8)$$

Intersection prevention. Similar to curvilinear layout optimization, nodes and edges in an octilinear layout should also be sufficiently far to prevent unwanted intersections. Otherwise, users may be misled and may attempt to switch trains at a wrong station. Let k and $\{i,j\}$ be an arbitrary node and an arbitrary edge, respectively. To prevent unwanted intersections, we maintain

$$\left| \mathbf{y}_k - (\delta \mathbf{y}_i + (1-\delta)\mathbf{y}_j) \right| > \varepsilon, \quad (9)$$

where $\mathbf{y}_\eta = \mathbf{v}_c + \sum_{\{a,b\}\in \mathbf{E}_{c\to\eta}} \omega_{ab}\mathbf{x}_{ab}$, η is a node index, ε is a threshold that indicates the minimum distance between a node and an edge, and δ is the interpolation coefficient obtained from point line theory.

Optimization details. We minimize the integrated energy term $\Omega = \beta_p\Omega_p + \beta_g\Omega_g + \beta_e\Omega_e$ subject to topology preservation and intersection prevention constraints. We set $\beta_p = 10$, $\beta_g = 1$, and $\beta_e = 1$ in our system. Obviously, Ω_p is given by a large weight because handle positions should be satisfied to achieve an intuitive manipulation. We optimize this constrained objective function by using the Lagrangian method [15]. The edge lengths obtained from the previous iteration are assigned to the initial guess to achieve fast convergence. Node positions are then determined whenever edge lengths are updated. Note that we assume that the metro map initially has no unwanted intersections and all intersection prevention

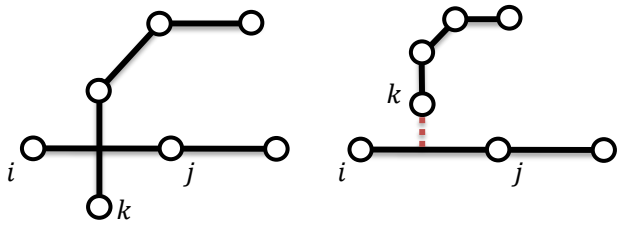


Fig. 6. (Left) An intersection may occur during editing. (Right) If the edges intersect, we iteratively transform the layout back to that of the previous step. An intersection prevention constraint is then added to maintain minimal distance between edges.

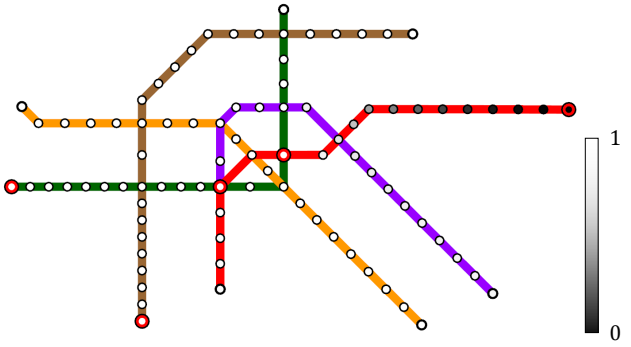


Fig. 7. We manipulate the right handle in this Vienna metro map. The transfer function of the obtained stiffness value is shown at the right. The nodes shaded in black indicate freedom of movement, whereas the white nodes in white should be kept stable during manipulation.

constraints are inactive. When an intersection occurs during editing, we iteratively transform the layout back to that of the previous step until the intersection is free (Figure 6). Afterward, we find a node k and an edge $\{i, j\}$ from the intersection and add

$$\Omega_i = \left|(\mathbf{y}_k - (\delta \mathbf{y}_i + (1-\delta)\mathbf{y}_j)) - \mathbf{g}_{ij(k)}\right|^2 \qquad (10)$$

to the system, where $\mathbf{g}_{ij(k)}$ is an ε-length vector from node k to edge $\{i, j\}$, which is applied to maintain the distance between the node and the edge, as well as to prevent intersection occurrence.

3.3 Stability constraints for an octilinear layout

Our system solves for the positions of all nodes whenever handles are manipulated. This framework ensures a visually pleasing result at each step. However, it does not consider temporal stability when consecutive layouts are determined. Given that different octilinear layouts may have similar energies, unexpected changes in irrelevant routes may occur during editing. This problem is visually salient because edges can only extend in discrete directions.

We consider handle positions and determine the influence region during manipulation to prevent changes in irrelevant routes. Let the manipulated handles be *active* and those fixed at the original positions be *inactive*. Our system expects the nodes influenced by active handles to move and the others to remain stable. To achieve the aim, we determine a smooth distribution based on the map topology to indicate the stiffness of each node. That is, we minimize the objective function

$$\sum_{i \in \mathbf{H}} |f_i - d_i|^2 + \sum_{\{i,j\} \in \mathbf{E}} |f_i - f_j|^2, \qquad (11)$$

where $\mathbf{H}$ is the set of handles, and $d_i = \{1, 0\}$, which depends on whether node i is active or inactive. The former part of the objective function indicates the stiffness of active and inactive handles and the latter part enforces neighboring nodes to have similar values. The obtained f_i will be a fractional number because the sum of the L_2 norm of $f_i - f_j$ is the minimum. Figure 7 shows the distribution of the Vienna metro map. The nodes shaded in black and in white are movable and stable, respectively, because the handle at the right is manipulated. Therefore, to achieve stable manipulation, we constrain each node to stay at its previous position according to its stiffness when computing for the curvilinear layout. Namely, we prevent edges in irrelevant routes from being rotated when an octilinear layout is computed. Therefore, in Equation 2, we retain the angle ϕ^o_{ij} and simply set α_{ij} with a large value ($\alpha_{ij} = 10$ in our system) if both stiffness f_i and f_j are larger than 0.95. This strategy stabilizes the map layout when it is manipulated. Our system applies the stable constraint only to edge directions because the change of edge lengths does not induce popping artifacts.

We apply animation to transit consecutive layouts. Nodes are moved with small steps in space each time to reduce popping effects. We refer the readers to our accompanying video for the results.

3.4 Labeling station names

Our system computes for the position of each station name once the octilinear layout is obtained. The name is placed around the station and can extend in one of the octilinear directions. Three requirements should be satisfied when placing these labels: 1) extending neighboring station names in the same direction, 2) preferentially positioning its name on either side of a station, and 3) preventing occlusion of station names. The first requirement allows users to read station names without frequently switching attention. The latter two constraints ensure that all characters can be recognized easily. We also allow users to specify the direction of a station name for particular needs such as aesthetic concerns and personal preferences if necessary. These requirements are formulated into energy terms, and the positions of station names can be obtained by solving a labeling problem. We refer readers to [5] for additional details.

Fig. 8. Stockholm (left), Montreal (middle), and Taipei (right) metro maps generated using our system.

4 Results and Discussions

We have implemented the presented algorithm and run the code on a desktop PC with a Core i7 3.0 GHz CPU. Although generating an octilinear layout while approximating the demanded handle positions is challenging, our system still achieves interactive performance. For the Berlin metro map, which contains 170 nodes and 182 edges, our system takes approximately 0.08, 0.02, and 0.05 seconds to solve for the curvilinear layout, edge directions and edge lengths, respectively. Our system is fast because the objective function is solved indirectly. Some hard constraints are relaxed at the beginning and then added back to the system. We first determine the curvilinear layout by solving a quadratic optimization problem, which has been proven to be efficient [15], [16]. This layout is then considered to be a reference. When rotating edges to become octilinear, edge directions that significantly deviate from their curvilinear versions will be pruned. In addition, we formulate the requirements into soft constraints and minimize the objective function in a least squares sense. The obtained edge directions are very close to but not exactly octilinear. Finally, our system solves for unknown variables with different natures in separate passes. It prevents the solution from quickly falling into a local optimum but does not guarantee that the solution will be the global optimum.

We show the metro maps generated by using our system in Figures 1, 8, and 10, as well as in the accompanying video. In these results, edges extend either vertically, horizontally, or diagonally. Although the metro maps are solved in a least squares sense, the force of the octilinear constraint is large when the system converges. Therefore, the deviation of each edge from its octilinear direction is experimentally less than 1° when a layout is optimized. Afterward, we simply set the angle of edge $\{i, j\}$ to ϕ_{ij}^o and ensure that the layout is octilinear. We found that this strategy does not introduce visual difference.

Some metro systems such as those in Sydney and Singapore are built in 3D. These lines intersect geographically without letting passengers switch trains and introduce unwanted intersections that are defined in our algorithm. Given that such intersections are unavoidable when rendering 3D metro lines on a 2D plane, our optimization may continue regardless of the number of iterations. Thus, we create a dummy node at each geographic intersection, which does not represent a station, before computing for the layout to prevent this problem.

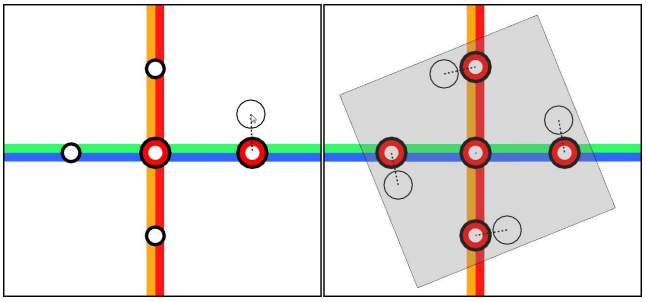


Fig. 9. We visualize target handle positions by circles and dashed lines to inform users that positional and octilinear constraints cannot be exactly achieved in some extreme cases. (Left) Keeping the center handle unmovable while pulling the right handle upward within a short distance. (Right) Rotating the five handles simultaneously with a small angle.

User interface. Our system allows users to manipulate not only a single node but also a portion of the map. Users can select the region of interest, and then translate, rotate, and scale it. All nodes within this region will be transformed identically and then constrained at the new positions. This interface is particularly useful in regions where nodes require an identical transformation. For example, to enlarge a region where nodes are close to one another, the scaling operator can simultaneously change all relative distances between them. Hence, repeatedly moving the node is not necessary.

In some extreme cases, a layout that perfectly satisfies both positional and octilinear constraints does not exist. For example, expecting two nodes of an edge to be located at $(0, 0)$ and $(1, 0.5)$, respectively,

Fig. 10. Berlin (left) and Sydney (right) metro maps generated using our system. Both metro systems have over 170 stations.

while ensuring the edge direction to be octilinear is impossible. This problem occurs when the free nodes between handles are few and edges are rotated with an arbitrary angle, as illustrated in Figure 9. Hence, we can only fulfill octilinear constraints while minimizing the deviation of each handle from the demanded position. Considering that user inputs are not fully satisfied, we inform users that our system trades handle positions for an octilinear layout by visualization. That is, the target positions of the manipulated handles are rendered in black circles. The deviations between solved and expected positions are indicated in dashed black lines.

Comparisons. We introduce an editing system that allows metro maps to be manipulated interactively. Although adopting existing methods may achieve this objective, the optimization techniques based on hill climbing [2] and mixed integer programming [3] are expensive. The performance of [2] is slow because it solves edge lengths and edge directions simultaneously during optimization. These two variables have significantly different natures and should not be optimized together. Otherwise, the search region will be highly nonlinear. Nöllenburg et al. [3] formulated the requirements into discrete constraints. However, the optimization problem is NP-hard and entails expensive computational cost. Improving the performance of these two methods to achieve interactive manipulation is difficult unless the whole optimization is reformulated. Instead, we add positional constraints to the technique presented by Wang and Chi [5] and mainly compare this extension to our method. Given that edge directions are independent of handle positions, this simple extension is highly unstable. Handles typically move to unexpected positions because of edges with wrong directions. The accompanying video verifies this limitation. By contrast, handles are controllable when users apply our system

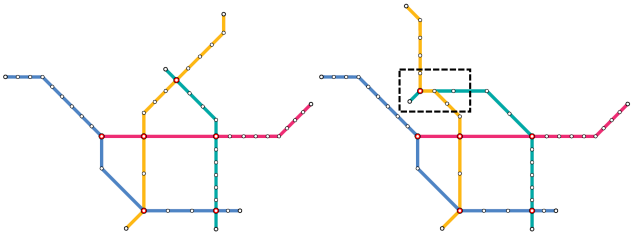


Fig. 11. Two different layouts of the Lisbon metro map. Our method determines the direction and then the length of each edge when computing a map layout. Intersections are inevitable when the directions of neighboring edges are identical, as highlighted by the dashed rectangle.

to manipulate metro maps. Hence, we claim that our technique is the only system that performs in real time and that provides an intuitive editing interface.

Limitations. The presented system solves for edge directions and edge lengths independently to achieve interactive performance. Consequently, the obtained layouts are not guaranteed to be globally optimal. Intersections are inevitable when neighboring edges extend in the same direction (Figure 11), although we have strove to maximize the included angle of neighboring edges in the optimization. This problem potentially occurs at junction nodes with four neighbors and above because of limited edge directions. Accordingly, using our system to edit metro maps with many junction nodes can be less effective. We plan to address this problem under the condition of interactive performance in future.

5 Evaluations

5.1 Subjective evaluation

To evaluate our editing system, we presented the prototype program to a professor and three research

assistants who work in the map and multimedia laboratory of the Department of Geography. We first explained our objectives and then demonstrated how to use our system to these cartographers. After the cartographers edited some metro maps and became familiar with our interface, we conducted a semi-structure interview and requested for their feedback. The interview aimed to make clear whether our system is intuitive, helpful to metro map creation, and beneficial to their work.

The cartographers indicated that they typically need one day to create a metro map using Adobe® Illustrator. One of the cartographers showed us how he creates a metro map:

"I will set the geographic metro map as a background and overlay my designed layout on it. Each transportation line will be straightened from its geographic version and checked whether the distortion is acceptable. The station names are then labeled once the layout is determined. At this step, I have to check whether the labels are sufficiently large and ensure that no labels are occluded. However, in many scenarios, the layout has to be refined if the station names cannot be appropriately labeled. This process has to be repeated until I feel everything is satisfactory. (S3)"

Although the presented system can help cartographers create a metro map within a few minutes, they pointed out that the system may not be beneficial to them because metro maps are not frequently updated. However, this tool should be useful to general users because it reduces the effort of metro map editing. They believed that many creative ideas and applications will be developed when a required map can be easily created. This opinion is based on the fact that metro maps are unnecessarily created by cartographers because geographic accuracy is not the main issue. Instead, they pay more attention to network topology, display space, map users, and passenger knowledge on the geography of a city. For example, maps designed for foreign passengers have to focus on topology to prevent the passengers from getting lost; whereas maps designed for citizens should consider geography, travel time, and travel fee to save daily transportation cost. In all, the cartographers indicated that topology is the most important and should be preserved when the display area is small or even flat [1]. Other information can be neglected when the display area is insufficiently large.

The cartographers indicated that our system is easy to use because they can manipulate the map layout by controlling only a small number of stations. This characteristic is important because they typically have no exact idea about the map layout before they start creating it. Because trial and error is inevitable, placing node positions to achieve an aesthetic presentation and prevent label occlusions is a tedious work. Therefore, the cartographers agreed that our editing system is intuitive. One of the cartographers said:

"The automatic relocation of stations is very convenient. In that case, I don't need to specify all details when editing a map." (S1)

However, they expected station names to be always displayed on the map during editing. This characteristic allows them to immediately view the final appearance of a map and check its geographic accuracy. Specifically, the cartographers said:

"The system does not label the station names whenever the layout is updated. However, I need to know the name of each station so as to compare the real and the schematic maps, and make sure whether the geographic distortion is acceptable." (S3)

We admitted that our system can only interactively label station names of small metro maps because solving a labeling problem is computationally expensive. We attempt to solve this performance issue in future. Another problem pointed out by the cartographers was the lack of undo and redo operations. We added these two functions to our system after the user study.

5.2 Quantitative evaluation

Because the cartographers indicated that our system could be useful to general users and helpful to realize interesting ideas in a metro map, we quantitatively evaluate the effort that users have to take when creating a map. Particularly, we would like to know whether users can create a map within the attention span, which is approximately 5 minutes and no longer than 20 minutes for an adult. [19]. We also attempt to understand the numbers of control stations, interaction steps, and whether intersections frequently occur during the editing. In other words, the hypothesis is that users can create a metro map within 5 minutes if they have a certain idea about the map. We conducted a user study with several participants to quantitatively evaluate our system.

Participants. We asked 8 participants to edit metro maps and recorded the editing statistics. These participants were the students recruited from the department of computer science; aged from 20 to 24; and 50% of them were females. They were trained to use our system by a five-minute presentation and allowed to interact with maps until they became familiar with the interface. During the study, they had to complete the task by themselves without any assistance.

Task. We gave the participants three metro maps and asked them to produce the layouts as close as possible to the ones we presented, which were shown in our supplemental material. The task was designed in this way because a good metro map is difficult to define. Instead, our goal is to understand the effort when users have a certain idea and attempt to create a metro map by using our system.

Environment. The evaluation was mainly performed in a quite room. Each participant operated a

1. https://www.flickr.com/photos/erussell1984/11315336303/

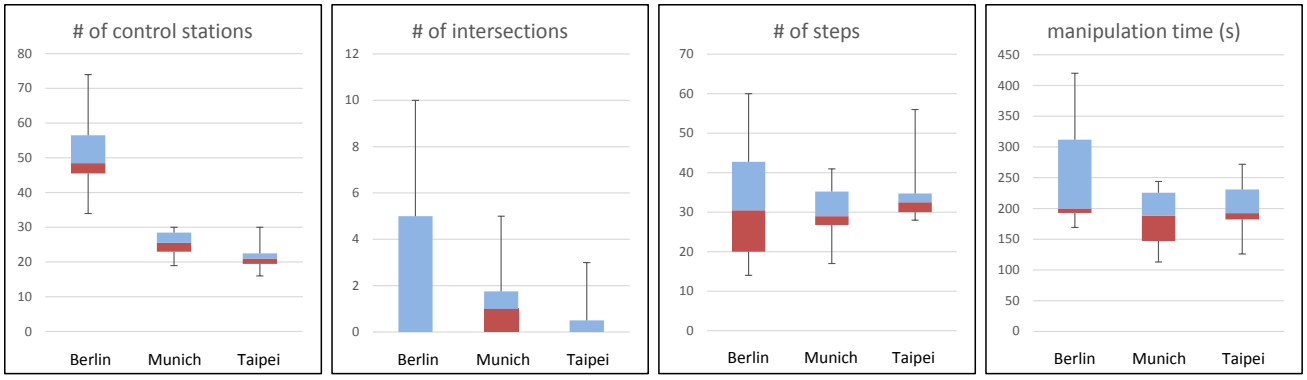


Fig. 12. The metro maps of Berlin, Munich, and Taipei are composed of 170, 96, and 97 stations, respectively. The Box and Whisker plots show the numbers of control stations, intersections, and interaction steps, as well as the manipulation time recorded from eight participants when they edited these maps.

desktop PC that can run our program interactively. The editing result was displayed on a 27-inch screen with a resolution of 1920×1080. The editing statistics, such as the numbers of controlled stations, intersections, and interaction steps, as well as the total manipulation time, were automatically recorded by our program during the study.

Results. The box and whisker plots in Figure 12 show the statistics. The five-number-summaries from top to bottom are max, the 25^{th}, 50^{th}, and 75^{th} percentiles, and min. As indicated, approximately 20-30% of the stations were controlled; less than 10 intersections occur, and the manipulation time was approximately 3-5 minutes. The number of controlled stations could be large because several stations were transformed simultaneously when a region was edited. However, editing the map by using our system was not tedious, which was justified by the few interaction steps and the short manipulation time. We also applied a t-test to analyze whether our hypothesis was supported. The result indicated a strong evidence that the alternative hypothesis "manipulation time is within 5 minutes" is true for the three cases ($p < 0.05$ for Berlin, $p < 0.005$ for Munich and Taipei).

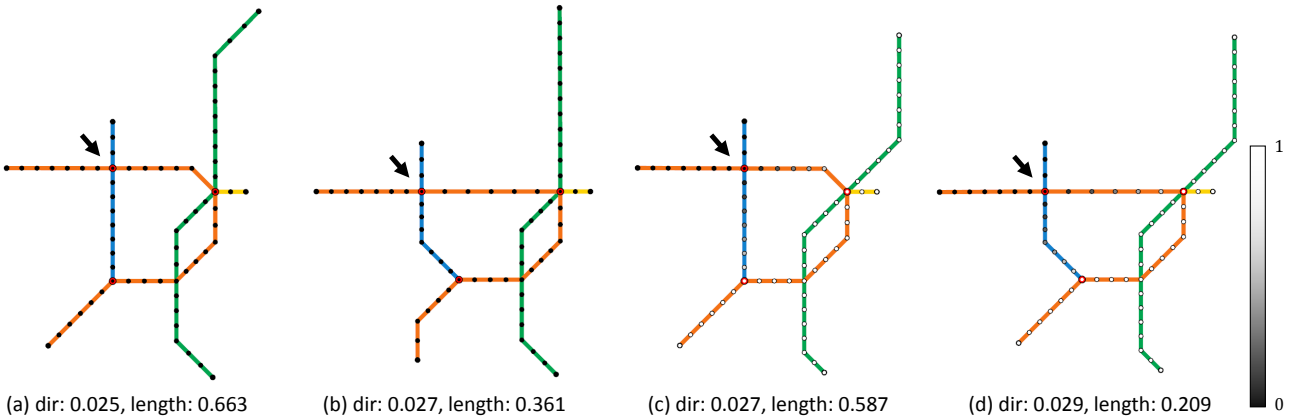
We observed that the number of interaction steps had a large variation among participants and was roughly proportional to the manipulation time. To make sense of this phenomenon, we conducted a short interview to the participants who took longer time to create a map. These participants mentioned that they seldom or even did not use region editing in the study. They simply preferred controlling one station at a time and had no idea the way would take longer time.

5.3 Stability evaluation

Our system applies the stability constraint to prevent irrelevant routes from being changed when handles are manipulated. The added energy terms potentially result in the system more constrained. To understand how much layout quality is decreased, we compute for the layout energies and present the results in Figure 13 with and without the constraint. Because the objective function is solved in a least squares sense $\mathbf{Ax} = \mathbf{b}$ when edge directions are optimized, we determine the mean energy through $\frac{1}{n}|\mathbf{Ax}-\mathbf{b}|^2$, where n is the number of constraints. Note that energies related to stability constraints are not measured. As indicated, the mean energy of the edge directions and edge lengths does not always become large when the stability constraint is considered. This result is reasonable because the objective function used to define good edge directions is non-linear. As our method cannot guarantee to obtain globally optimal solutions, the layouts with similar energies can be considered to have equal quality. We refer readers to the accompanying video for quality comparison with and without stability constraint because dynamic layout changes are difficult to observe in still images.

5.4 Intuition analysis

We present an intuitive interface that allows cartographers and non-professionals to manipulate node positions when creating metro maps. Several design guidelines are presented to achieve this objective, including interactive, intelligent, stable, and effective. We analyze whether these guidelines are achieved. First, our system performs interactively because the optimization takes less than 0.1 second to determine the layout of a medium-size metro map. Second, the determined layouts satisfy metro map criteria and approximate the demanded handle positions. Third, nodes that are close to inactive handles are retained at their previous positions during editing and thus, global layouts are stable when a local region is manipulated. Finally, editing a metro map approximately takes 30 interaction steps and 3-5 minutes if the user has a clear idea of the metro map layout, as indicated in Figure 12. Given that all the aforementioned design guidelines are satisfied, we claim that our editing system is easy to use.



(a) dir: 0.025, length: 0.663 (b) dir: 0.027, length: 0.361 (c) dir: 0.027, length: 0.587 (d) dir: 0.029, length: 0.209

Fig. 13. Montreal metro maps. Left and right show the manipulations without (a-b) and with (c-d) our stability constraints. In this example, only the handle pointed by the arrow is manipulated. The determined node stabilities are indicated by colors, and the transfer function is shown at the right. The top right and bottom left (irrelevant) routes change when the map is manipulated without the stability constraint, which may lead to unexpected outcomes during manipulation. The problem occurs because both layouts fulfill the requirements. This finding can be explained by the fact that the mean averages of edge directions and edge lengths do not increase considerably when the stability constraint is considered.

6 Conclusion and Future Works

We presented an editing system that allows users to design metro maps by controlling a small number of nodes. Our system supports curvilinear and octilinear layouts during manipulation. Both layouts are expected to have straight lines, evenly spaced stations, and the maximized included angles at junctions. We further constrain routes to extend either vertically, horizontally, or diagonally when creating an octilinear map. Although some methods have been presented to lay out metro maps automatically, a system that considers user input remains essential. The presented work can dynamically consider human knowledge and adjust the layout to make it consistent with user expectations. Furthermore, our algorithm is widely applicable. It can be used to edit other octilinear layouts such as water or gas utility networks [20]. We also believe that our optimization framework is beneficial to generating metro tourist maps [12], [13], [14] because of its efficiency.

Metro maps such as that of the New York City Subway[2] are designed to overlap street maps. In such cases, cartographers trade the clarity of map topology for geography. The advantage of this map is that users can self-navigate to their destinations by walking after leaving the train station. We intend to design a layout algorithm that can maximize the clarity of both information in the near future.

Acknowledgements

We thank the anonymous reviewers for their constructive comments. We are grateful to Prof. Jinn-Guey Lay, Prof. Ming-Te Chi, Mr. Hsin-Hsiang Tseng, Miss Yuan-Hua Chang, and Miss Yi Lin for helping on the user study. We also appreciate Mr. Chia-Lun Ku for the discussion of statistical analysis in our evaluation. This work was supported in part by the National Science Council (101-2628-E-009-020-MY3, 100-2628-E-006-031-MY3, and 100-2221-E-006-188-MY3).

2. http://web.mta.info/maps/submap.html

References

[1] K. Garland, "Mr. Beck's underground map." Capital Transport, 1994.

[2] 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*, vol. 17, pp. 101–114, 2011.

[3] M. Nöllenburg and A. Wolff, "Drawing and labeling high-quality metro maps by mixed-integer programming," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, pp. 626–641, 2011.

[4] D. Delling, A. Gemsa, M. Nöllenburg, T. Pajor, and I. Rutter, "On d-regular schematization of embedded paths," *Computational Geometry: Theory and Applications*, vol. 47, no. 3, pp. 381–406, 2014.

[5] Y.-S. Wang and M.-T. Chi, "Focus+context metro maps," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2528–2535, Dec. 2011.

[6] A. Wolff, "Drawing subway maps: A survey." *Informatik - Forschung und Entwicklung*, vol. 22, no. 1, pp. 23–44, 2007.

[7] S. Avelar and M. Müller, "Generating topologically correct schematic maps," in *Proceedings of the 9th International Symposium on Spatial Data Handling*, 2000, pp. 4–28.

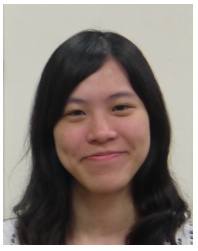
[8] D. Merrick and J. Gudmundsson, "Path simplification for metro map layout," in *Proceedings of the 14th International Symposium on Graph drawing*, 2007, pp. 258–269.

[9] S.-H. Hong, D. Merrick, and H. A. D. do Nascimento, "Automatic visualisation of metro maps," *Journal of Visual Languages and Computing*, vol. 17, pp. 203–224, 2006.

[10] M. Fink, H. Haverkort, M. Nöllenburg, M. Roberts, J. Schuhmann, and A. Wolff, "Drawing metro maps using bézier curves," in *Proceedings of the 20th International Symposium on Graph drawing*, 2013, pp. 463–474.

[11] M. J. Roberts, E. J. Newton, F. D. Lagattolla, S. Hughes, and M. C. Hasler, "Objective versus subjective measures of paris metro map usability: Investigating traditional octolinear versus all-curves schematics," *International Journal of Human-Computer Studies*, vol. 71, no. 3, pp. 363–386, 2013.
[12] H.-Y. Wu, S. Takahashi, C.-C. Lin, and H.-C. Yen, "Travel-route-centered metro map layout and annotation," *Computer Graphics Forum*, vol. 31, no. 3, pp. 925–934, 2012.
[13] H.-Y. Wu, S. Takahashi, D. Hirono, M. Arikawa, C.-C. Lin, and H.-C. Yen, "Spatially efficient design of annotated metro maps," *Computer Graphics Forum*, vol. 32, no. 3, pp. 261–270, 2013.
[14] P. Claudio and S.-E. Yoon, "Metro transit-centric visualization for city tour planning," *Computer Graphics Forum*, vol. 33, no. 3, pp. 271–280, 2014.
[15] K. Madsen, H. B. Nielsen, and O. Tingleff, *Optimization with Constraints*. Informatics and Mathematical Modelling, Technical University of Denmark, 2004.
[16] ——, *Methods for Non-Linear Least Squares Problems*. Informatics and Mathematical Modelling, Technical University of Denmark, 2004.
[17] S.-S. Lin, C.-H. Lin, Y.-J. Hu, and T.-Y. Lee, "Drawing road networks with mental maps," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 9, pp. 1241–1252, Sept 2014.
[18] T. C. van Dijk and J.-H. Haunert, "Interactive focus maps using least-squares optimization," *International Journal of Geographic Information Science*, vol. 28, no. 10, pp. 2052–2075, 2014.
[19] D. C. M. F. A. David Cornish, M.d. and D. Dukette, *The Essential 20: Twenty Components of an Excellent Health Care Team*. RoseDog Books, 2009.
[20] S. Anand, S. Avelar, J. Ware, and J. Jackson, "Automated schematic map production using simulated annealing and gradient descent approaches," in *Proceedings of the GIS research UK 15th Annual Conference*, 2007.

Yu-Shuen Wang received the BS and PhD degrees from the Department of Computer Science and Information Engineering, National Cheng-Kung University, in 2004 and 2010, respectively. He is currently an assistant professor of the Department of Computer Science at National Chiao Tung University (http://people.cs.nctu.edu.tw/ yushuen/). He leads the Computer Graphics and Visualization Lab at the Institute of Multimedia Engineering. His research interests include computer graphics, computational photography, and visualization.



Wan-Yu Peng received the BS and MS degrees from the Department of Applied Mathematics and from the Department of Computer Science, National Chiao Tung University, Taiwan, in 2011 and 2013, respectively. Now, she is working in Trend Micro Inc. Her research interests include computer graphics and visualization.