A Mixed-Integer Program for Drawing High-Quality Metro Maps[⋆]

Martin Nöllenburg and Alexander Wolff

Fakultät für Informatik, Universität Karlsruhe, P.O. Box 6980, D-76128 Karlsruhe.
`http://i11www.ira.uka.de/algo/group`

Abstract. In this paper we investigate the problem of drawing metro maps which is defined as follows. Given a planar graph G of maximum degree 8 with its embedding and vertex locations (e.g. the physical location of the tracks and stations of a metro system) and a set $\mathcal{L}$ of paths or cycles in G (e.g. metro lines), draw G and $\mathcal{L}$ *nicely.* We first specify the niceness of a drawing by listing a number of *hard* and *soft* constraints. Then we present a mixed-integer program (MIP) which always finds a drawing that fulfills all hard constraints (if such a drawing exists) and optimizes a weighted sum of costs corresponding to the soft constraints. We also describe some heuristics that speed up the MIP. We have implemented both the MIP and the heuristics. We compare their output to that of previous algorithms for drawing metro maps and to official metro maps drawn by graphic designers.

1 Introduction

A metro map is a schematic drawing of the underlying geographic network that represents the different stations and metro lines of a metro system. The users of a metro map are the passengers of the public transport system. They want to quickly answer questions like "How do I get from A to B?" or "After how many stops do I have to change trains?". Thus the layout of a metro map must be as clear as possible whereas exact geometry or scale is less important. The problem of drawing maps of metro systems and other means of public transportation is an interesting compromise between schematic road maps [4] where vertex positions are (mostly) fixed and "conventional" graph drawing where vertices can go anywhere. The first approach maximizes maintenance of the user's mental map, the second approach maximizes esthetics. The mother of all modern metro maps is Henry Beck's 1933 map of the London Underground. In the meantime, graphic designers have come up with different layout styles all over the world [8].

After studying a large number of real-world metro maps [8] we formalized the problem of drawing high-quality metro maps as follows. As usual we say that an embedding of a graph G associates to each vertex a list of its adjacent vertices in clockwise order. We say that a set $\mathcal{L}$ of paths and cycles of G is a *line cover* of G if each edge of G belongs to at least one element of $\mathcal{L}$. Now the *metro-map layout*

⋆ Work supported by grant WO 758/4-2 of the German Science Foundation (DFG).

problem is the following. Given (a) a planar graph G of maximum degree 8, the *metro graph*, (b) the embedding of G, (c) for each vertex v its location $\pi(v)$ in the plane, and (d) a line cover $\mathcal{L}$ of G, the *metro lines*, find a *nice* drawing μ of G and $\mathcal{L}$. In order to be nice, μ must fulfill a number of *hard* constraints:

(H1) μ must respect the topology of G,
(H2) all edges of $\mu(G)$ must be *octilinear* line segments, i.e. parallel to one of the two coordinate axes or to either of their two bisectors,
(H3) each edge e in $\mu(G)$ has a minimum length ℓ_e, and
(H4) each edge in $\mu(G)$ has a certain minimum distance $d_{\min}$ from each non-incident edge.

Moreover, μ should conform to a number of *soft* constraints as tightly as possible:

(S1) the paths and cycles in $\mu(\mathcal{L})$ should have few bends,
(S2) the total edge length of $\mu(G)$ should be small, and
(S3) for each pair of adjacent vertices (u, v) their *relative position* should be preserved, i.e. the angle $\angle(\mu(u), \mu(v))$ should be similar to the $\angle(\pi(u), \pi(v))$, where $\angle(a, b)$ is the angle between the x-axis in positive direction and the line through a and b directed from a to b.

Note that if the embedding of a metro graph is not planar, this can be achieved by introducing dummy vertices at crossings, of which there are usually not many. We denote the number of vertices (including dummy vertices) and edges of G by n and m, respectively. Let m' be the total number of edges of the paths and cycles in $\mathcal{L}$. We have $m' \geq m$.

While the need for most of the above constraints is immediate, constraint (S3) may need a few explanatory words. The intuition behind requiring the preservation of the relative position is that users of metro system usually have a certain notion of compass directions above ground. Suppose a passenger is in $\pi(u)$ and wants to go to the adjacent metro station $\pi(v)$, which he knows to lie south of $\pi(u)$, then he would be confused if $\mu(u)$ was north of $\mu(v)$. Thus ensuring that the two angles in constraint (S3) do not deviate too much, say by no more than 90 degrees, can be seen as a hard constraint, while it seems to be appropriate to model smaller deviations as a soft constraint, e.g. by charging a cost proportional to the deviation. Our framework reflects this ambivalence, but modeling relative position as a purely soft constraint is also possible, see Sect. 3.1 and 3.6.

Compared to the orthogonal drawing of (embedded) graphs, the introduction of diagonal directions yields drawings that are more similar to the original embedding. In addition, the maximum vertex degree increases from 4 to 8. However, in contrast to the existence of several efficient algorithms for orthogonal drawings [12, 5], the problem becomes NP-complete in the octilinear case as we show in [7]. This partially motivates why we do not follow the topology-shape-metric approach [5] for orthogonal graph drawing: while we could compute a minimum-bend octilinear shape of a metro graph in polynomial time using Tamassia's flow model [12], we cannot efficiently embed the resulting shape without creating crossings even if an octilinear layout exists.

Therefore we decided to model the metro-map layout problem as a MIP, see Sect. 3. This gave us the necessary flexibility to achieve the following. If a layout that conforms to all hard constraints exists (which was the case in all examples we tried), then our MIP finds such a layout. Moreover our MIP optimizes the weighted sum of cost functions each of which corresponds to a soft constraint. Our MIP is the first method that guarantees octilinearity, which is essential for a clear layout of metro maps. Our MIP is also the first method dedicated to drawing metro maps that uses global optimization and thus avoids getting trapped in local minima. This contrasts with methods based on local optimization, see Sect. 2. In [7] we extend our model to combine graph drawing with the placement of non-overlapping station labels. Binucci et al. [2] have used a MIP formulations to combine *orthogonal* graph drawing and label placement.

In order to cope with the running time of MIP solvers, we give several heuristics that speed up our basic MIP, see Sect. 3.7. We have implemented an algorithm based on our MIP formulation. In Sect. 4 we present a metro map that our algorithm drew of a real-world metro system and compare it to the output of previous algorithms and to an official metro map.

We stress that our MIP formulation can be used not only for drawing metro maps, but for any kind of technical drawing with a restricted number of directions. Brandes et al. [3] introduced the concept of a *sketch* of a graph. A sketch can be handmade or the physical embedding of a geometric network like the real position of telephone cables. Brandes et al. compute an *orthogonal* drawing of a sketch in $O(n^2 \log n)$ time. However, their method cannot be extended to more directions or to incorporate the concept of metro lines. In contrast, our framework can be used to draw sketches (possibly dropping constraint (S1)) and can be extended to more than eight directions. Other possible extensions include user interaction (e.g. fixing the direction edges or lines), the drawing of maps in a given format, or the minimization of one dimension of the drawing area (instead of constraint (S2)).

2 Previous work

To the best of our knowledge the first attempt to automate the drawing of metro maps was made by Barkowsky et al. [1]. They use *discrete curve evolution*, i.e. an algorithm for polygonal line simplification, to treat the lines of the Hamburg subway system. However, their algorithm neither restricts the edge directions nor does it increase station distances in the crowded downtown area. Stations are labeled but no effort is made to avoid label overlap.

Hong et al. [6] give five methods for the metro-map layout problem. The most refined of these methods modifies a topology-maintaining spring embedder such that edge weights are taken into account and such that additional magnetic forces draw the straight-line edges towards the closest octilinear direction. In a preprocessing step the metro graph is simplified by contracting each edge that is incident to a degree-2 vertex. After performing all contractions, the weight of each remaining edge is set to the number of original edges it replaces. After the final layout has been computed, all degree-2 vertices are re-inserted into the corresponding edges in an equidistant manner. The contraction step reduces the running time considerably. Station labels are placed in one out of eight directions. While label–label overlaps are avoided, diagonally placed labels sometimes intersect network edges.

Stott and Rodgers [10] draw metro maps using multi-criteria optimization based on hill climbing. For a given layout they define metrics for evaluating the number of edge intersections, the octilinearity and the length of edges, the angular resolution at vertices and the straightness of metro lines. The quality of a layout is a weighted sum over these five metrics. Their iterative optimization process starts with a layout on the integer grid that is obtained from the original embedding. In each iteration they consider alternative grid positions for each vertex within a certain radius. For each of these grid positions they compute the quality of the modified layout. If any of the positions improves the quality of the layout, they move the current vertex to the best position among those that do not change the topology of the layout. They observed typical problems with local minima during their optimization process and give a heuristic fix that overcomes one of these problems. Stott and Rodgers have experimented with enforcing relative position, but report that it does not really improve the results. They can label stations, but do not check for overlaps other than with the edges incident to the current station. They use the same contraction method as Hong et al. [6] to preprocess the input graph.

The main advantage of our method over its predecessors is that we *guarantee* to keep all hard constraints (among them octilinearity) and that we avoid the problem of local optima.

Interestingly enough the layout principles of metro maps have not only been used in a geographic setting. E.g. Sandvad et al. [9] use the *metro-map metaphor* as a way to visualize abstract information related to the Internet.

3 The basic MIP model

A MIP consists of two parts: a set of linear constraints and a linear objective function. In Sect. 3.1 to 3.3 we describe four sets of constraints that model the hard constraints (H1)–(H4). We model the simultaneous optimization of the three soft constraints (S1)–(S3) in Sect. 3.4 to 3.6 using a weighted sum of three individual cost functions:

$$\text{Minimize } \lambda_{\text{length}}\,\text{cost}_{\text{length}} + \lambda_{\text{bends}}\,\text{cost}_{\text{bends}} + \lambda_{\text{dir}}\,\text{cost}_{\text{dir}}, \tag{1}$$

where the variables λ_i are positive user-defined weights, each of which individually emphasizes a certain esthetic criterion. The total number of constraints and variables in our model is of order $O(n + m' + m^2)$. Note that since G is planar we have $m \leq 3n - 6$ due to Euler's formula.

To be able to treat all four edge directions similarly, we use an (x, y, z_1, z_2)-coordinate system as depicted in Fig. 1, where each axis corresponds to one of the four feasible edge directions in the layout. For each vertex v we define $z_1(v) = x(v) + y(v)$ and $z_2(v) = x(v) - y(v)$.

3.1 Octilinearity and relative position

Before modeling the constraints we need some notation to address relative positions between vertices and to denote directions of edges. For each vertex v we define a partition of the plane into eight wedge-shaped sectors, numbered from 0 to 7 counterclockwise starting with the positive x-direction as in Fig. 2. To denote the rough relative position between two vertices u, v in the *original layout* we use the terms $\mathrm{sec}_u(v)$ and $\mathrm{sec}_v(u)$ representing the sector relative to u in which v lies and vice versa. Similarly, for each edge $\{u, v\}$, we define a variable $\mathrm{dir}(u, v)$ to denote the octilinear direction of $\{u, v\}$ in the *new layout*.

As mentioned in the introduction, we partially model the soft constraint (S3) as a hard constraint. As a compromise between conservation of relative positions and flexibility to obtain a nice drawing, we allow that an edge is drawn in three different ways. It can be drawn in the direction corresponding to its original sector relative to either endpoint or it can be drawn in the two neighboring directions. Let $\mathrm{sec}_u^{\mathrm{pred}}(v) = \mathrm{sec}_u(v) - 1 \pmod 8$, $\mathrm{sec}_u^{\mathrm{orig}}(v) = \mathrm{sec}_u(v)$ and $\mathrm{sec}_u^{\mathrm{succ}}(v) = \mathrm{sec}_u(v) + 1 \pmod 8$. We now restrict $\mathrm{dir}(u, v)$, which will be used in Sect. 3.5 and 3.6, to the set $\{\mathrm{sec}_u^{\mathrm{pred}}(v), \mathrm{sec}_u^{\mathrm{orig}}(v), \mathrm{sec}_u^{\mathrm{succ}}(v)\}$. This is expressed by the disjunction

$$\bigvee_{i \in \{\mathrm{pred},\mathrm{orig},\mathrm{succ}\}} (\mathrm{dir}(u, v) = \mathrm{sec}_u^i(v) \ \wedge \ \mathrm{dir}(v, u) = \mathrm{sec}_v^i(u)). \qquad (2)$$

To model (2) we introduce binary variables $\alpha_{\mathrm{pred}}, \alpha_{\mathrm{orig}}, \alpha_{\mathrm{succ}}$ and the constraint

$$\alpha_{\mathrm{pred}}(u, v) + \alpha_{\mathrm{orig}}(u, v) + \alpha_{\mathrm{succ}}(u, v) = 1 \qquad \forall\{u, v\} \in E. \qquad (3)$$

The variable that takes the value 1 will determine the direction in which edge $\{u, v\}$ is drawn, i.e. the term of disjunction (2) that will evaluate to true.

Now we model the correct assignment of $\mathrm{dir}(u, v)$ and $\mathrm{dir}(v, u)$. For each $i \in \{\mathrm{pred}, \mathrm{orig}, \mathrm{succ}\}$ we have the following set of constraints

$$\left.\begin{aligned} \mathrm{dir}(u, v) - \mathrm{sec}_u^i(v) &\le M(1 - \alpha_i(u, v)) \\ -\mathrm{dir}(u, v) + \mathrm{sec}_u^i(v) &\le M(1 - \alpha_i(u, v)) \\ \mathrm{dir}(v, u) - \mathrm{sec}_v^i(u) &\le M(1 - \alpha_i(u, v)) \\ -\mathrm{dir}(v, u) + \mathrm{sec}_v^i(u) &\le M(1 - \alpha_i(u, v)) \end{aligned}\right. \qquad \forall\{u, v\} \in E, \qquad (4)$$

where the variables of type $\mathrm{dir}(u, v)$ are integers in the range $\{0, \ldots, 7\}$ and M is a large constant. The use of the large constant M in connection with a set of binary variables as in (3) is a standard trick in MIP modeling for formulating a disjunction of constraints. The constant M must be an upper bound on the left-hand sides of the inequalities. Here, if $\alpha_i(u, v) = 0$, the constraints in (4) are trivially fulfilled and do not influence the left-hand sides. On the other hand, if $\alpha_i(u, v) = 1$, the four inequalities are equivalent to $\mathrm{dir}(u, v) = \mathrm{sec}_u^i(v)$ and $\mathrm{dir}(v, u) = \mathrm{sec}_v^i(u)$ as desired (equality constraints have to be transformed into two inequalities when using this trick). Due to (3), $\alpha_i(u, v) = 1$ for exactly one $i \in \{\mathrm{pred}, \mathrm{orig}, \mathrm{succ}\}$. Thus, exactly one term of the disjunction (2) must be fulfilled.

Further, depending on the actual values of $\sec_u^i(v)$, we add three more constraints for each $i \in \{\text{pred}, \text{orig}, \text{succ}\}$. For example let $\sec_u^{\text{orig}}(v) = 2$ (meaning v is vertically above u in the original layout). Then the constraints are as follows

$$\begin{aligned} x(u) - x(v) &\leq M(1 - \alpha_{\text{orig}}(u,v)) \\ -x(u) + x(v) &\leq M(1 - \alpha_{\text{orig}}(u,v)) \\ y(u) - y(v) &\leq M(1 - \alpha_{\text{orig}}(u,v)) - \ell_{\{u,v\}} \end{aligned} \qquad \forall \{u,v\} \in E, \tag{5}$$

where $\ell_{\{u,v\}} > 0$ is the minimum length of edge $\{u,v\}$. If $\alpha_{\text{orig}}(u,v) = 1$, these constraints force u and v to have the same x-coordinate and to keep a vertical distance of at least $\ell_{\{u,v\}}$. This is exactly what is needed for a vertical upward running edge. The other seven possibilities are formulated similarly by forcing one of the coordinates of both vertices to be equal and the distance along the respective direction to be at least $\ell_{\{u,v\}}$. Overall, this part needs $22m$ constraints and $5m$ variables.

3.2 Conservation of the Embedding

To guarantee conservation of the original embedding it suffices to maintain for each vertex $v \in V$ the circular ordering of all incident edges.

Let $N(v) = \{u_1, u_2, \ldots, u_{\deg(v)}\}$ denote the set of all neighbors of v. The counterclockwise ordering of the edges $\{v,u\} \in E$ incident to v implies an ordering on $N(v)$ by identifying each edge $\{v,u\}$ with the vertex u opposite of v. Assume the ordering is $u_1 < u_2 < \ldots < u_{\deg(v)}$. Then in the metro map layout one of these vertices, say u_i, is assigned the smallest direction number from the set of possible directions $\{0, \ldots, 7\}$. All other vertices in $N(v)$ must follow in the same order as before and must have strictly increasing direction numbers: $\text{dir}(v, u_i) < \text{dir}(v, u_{i+1}) < \ldots < \text{dir}(v, u_{i+\deg(v)-1})$, where in the following all indices greater than $\deg(v)$ are considered modulo $\deg(v)$. In other words, all but one of the inequalities $\text{dir}(v, u_1) < \text{dir}(v, u_2), \ldots, \text{dir}(v, u_{\deg(v)-1}) < \text{dir}(v, u_{\deg(v)}), \text{dir}(v, u_{\deg(v)}) < \text{dir}(v, u_1)$ must hold.

In order to determine the vertex with smallest direction number, we again use binary variables as in Sect. 3.1. But instead of using the standard trick to model a disjunction of $\deg(v)$ many terms with $\deg(v) - 1$ constraints each, we make use of the fact that in each case exactly one of the inequalities may be violated while the rest must hold. This requires about a factor $\deg(v)$ less constraints. They are as follows:

$$\beta_1(v) + \beta_2(v) + \ldots + \beta_{\deg(v)}(v) = 1 \qquad \forall v \in V, \deg(v) \geq 2, \tag{6}$$

with binary variables $\beta_i(v)$, and

$$\begin{aligned} \text{dir}(v, u_2) - \text{dir}(v, u_1) &\geq -M\beta_1(v) + 1 \\ \text{dir}(v, u_3) - \text{dir}(v, u_2) &\geq -M\beta_2(v) + 1 \\ &\vdots \\ \text{dir}(v, u_1) - \text{dir}(v, u_{\deg(v)}) &\geq -M\beta_{\deg(v)}(v) + 1 \end{aligned} \qquad \forall v \in V, \deg(v) \geq 2. \tag{7}$$

The variable β_i that takes value 1 in constraint (6) determines that vertex u_{i+1} has minimum direction number among $N(v)$ by not enforcing $\mathrm{dir}(v, u_{i+1}) - \mathrm{dir}(v, u_i) \geq 1$ in constraints (7). All other binary variables $\beta_j, (j \neq i)$ are set to 0 and thus $\mathrm{dir}(v, u_{j+1}) - \mathrm{dir}(v, u_j) \geq 1$ holds for all $j \neq i$.

These constraints not only enforce that the embedding is preserved but also that no two edges incident to the same vertex can have the same direction. An upper bound on the number of constraints and variables for this part of the MIP is given by $\sum_{v \in V}(\deg(v) + 1) \in O(m)$.

3.3 Planarity

For preserving planarity we have to ensure that certain pairs of edges do not intersect. This can be done in the octilinear setting by distinguishing eight possible relative positions for a pair $\{e_1, e_2\}$ of edges. We express these relative positions using compass orientations. Fixing an edge e_1 a second, non-intersecting edge e_2 can either be placed north, south, east, west or northeast, northwest, southeast, southwest of e_1. For example northeast means in terms of our coordinate system that both vertices incident to e_1 have strictly smaller z_1-coordinates than both vertices incident to e_2. The other relative positions are defined in a similar way.

Clearly, an octilinear drawing is planar if and only if each pair of non-incident edges is placed according to one of the above relative positions. Indeed, we model this disjunctive constraint for all pairs of edges. The constraint

$$\sum_{i \in \{\mathrm{N,S,E,W,NE,NW,SE,SW}\}} \gamma_i(e_1, e_2) \geq 1 \qquad \begin{array}{l} \forall (e_1, e_2) \in \binom{E}{2}, \\ e_1, e_2 \text{ not incident,} \end{array} \tag{8}$$

introduces the variables $\gamma_{\mathrm{N}}, \ldots, \gamma_{\mathrm{SW}}$. As an example we now give the constraints for the condition "e_2 is east of e_1"

$$\begin{array}{r} x(u_1) - x(u_2) \leq M(1 - \gamma_E(e_1, e_2)) - d_{\min} \\ x(u_1) - x(v_2) \leq M(1 - \gamma_E(e_1, e_2)) - d_{\min} \\ x(v_1) - x(u_2) \leq M(1 - \gamma_E(e_1, e_2)) - d_{\min} \\ x(v_1) - x(v_2) \leq M(1 - \gamma_E(e_1, e_2)) - d_{\min} \end{array} \qquad \begin{array}{l} \forall (e_1, e_2) \in \binom{E}{2}, \\ e_1, e_2 \text{ not incident.} \end{array} \tag{9}$$

Recall that $d_{\min}$ is the minimum distance between non-incident edges as given in (H4). Analogously, each of the other seven relative positions is modeled using four constraints each. This amounts to 33 constraints and 8 variables for each edge pair. The problem is that the number of edge pairs is $O(m^2)$. Therefore, we give several heuristics in Sect. 3.7 to reduce the number of constraints that enforce planarity.

3.4 Minimization of Edge Lengths

For modeling the edge lengths one has to specify the underlying metric. We decided to use the L^∞-metric, which defines the distance of two vertices u and v

to be $\max(|x(u) - x(v)|, |y(u) - y(v)|)$. We define new real-valued, non-negative variables $D(u, v)$ for all edges $\{u, v\} \in E$ which serve as upper bounds on the lengths of their respective edges. By setting

$$\text{cost}_{\text{length}} = \sum_{\{u,v\} \in E} D(u, v) \tag{10}$$

and by minimizing $\text{cost}_{\text{length}}$, the variables $D(u, v)$ indeed equal the corresponding edge lengths.

The constraints that bound $D(u, v)$ depend on the respective direction of the edge $\{u, v\}$. Note that the actual direction of this edge is determined according to the constraints in Sect. 3.1. Thus we can reuse the binary variables defined in that section to distinguish the three cases for the edge direction. As an example assume that $\text{sec}_u(v) = 1$. Then the constraints are

$$\left.\begin{array}{l} x(v) - x(u) \leq M(1 - \alpha_{\text{prev}}(u, v)) + D(u, v) \\ x(v) - x(u) \leq M(1 - \alpha_{\text{real}}(u, v)) + D(u, v) \\ y(v) - y(u) \leq M(1 - \alpha_{\text{next}}(u, v)) + D(u, v) \end{array}\right. \qquad \forall \{u, v\} \in E. \tag{11}$$

Note that for an edge $\{u, v\}$ drawn diagonally it holds that $|x(u) - x(v)| = |y(u) - y(v)|$. Hence we can use either of the x- or y-coordinates to determine the length $D(u, v)$. Edge lengths for other values of $\text{sec}_u(v)$ are modeled similarly. In total we use m variables and $3m$ constraints.

3.5 Avoiding Bends along Lines

Clarity in an octilinear drawing depends crucially on the ability to visually follow the metro lines. This can be partially enhanced by using distinguishable colors, but also by avoiding bends along the lines.

We define the bend cost subject to the actual angle between two adjacent edges on a path. Due to the octilinearity constraints and to the fact that two adjacent edges cannot have the same direction relative to their joint vertex the angles can only equal 180, 135, 90, and 45 degrees. In that order we define the corresponding bend cost to be 0, 1, 2, and 3, such that the cost increases with the acuteness of the angle, see Fig. 3.

In our model we can determine the angle between two adjacent edges $\{u, v\}$ and $\{v, w\}$ by using the values of $\text{dir}(u, v)$ and $\text{dir}(v, w)$. For ease of notation let $\Delta\text{dir}(u, v, w) = \text{dir}(u, v) - \text{dir}(v, w)$. Then, the bend cost can be expressed as

$$\text{bend}(u, v, w) = \begin{cases} |\Delta\text{dir}(u, v, w)| & \text{if } |\Delta\text{dir}(u, v, w)| \leq 4 \\ 8 - |\Delta\text{dir}(u, v, w)| & \text{if } |\Delta\text{dir}(u, v, w)| \geq 5. \end{cases} \tag{12}$$

Now we can set

$$\text{cost}_{\text{bends}} = \sum_{\{u,v\},\{v,w\} \in L,\, L \in \mathcal{L}} \text{bend}(u, v, w) \tag{13}$$

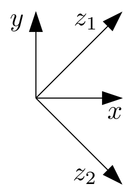


Fig. 1. Octilinear coordinate system.

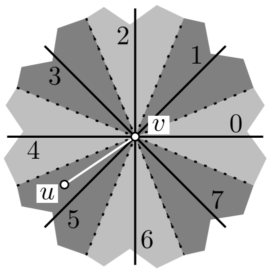


Fig. 2. Sectors relative to v, e.g. $\text{sec}_v(u) = 5$.

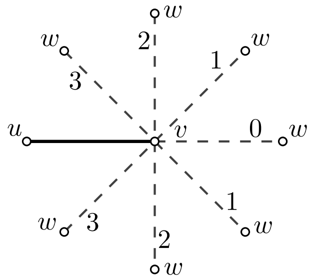


Fig. 3. Bend cost $b(u, v, w)$ for each value of $\text{dir}_v(w)$.

to minimize the number and acuteness of all bends along lines.

The formulation of bend cost in (12) cannot be transformed directly into a set of linear constraints because it involves absolute values and a case distinction. Here, we solve this problem using instead the following constraints for all lines $L \in \mathcal{L}$ and pairs of incident edges $\{u, v\}, \{v, w\}$ on L. Again, we need some binary variables, namely $\delta_1(u, v, w), \delta_2(u, v, w)$, and $\delta_3(u, v, w)$. The constraint

$$\delta_1(u, v, w) + \delta_2(u, v, w) + \delta_3(u, v, w) = 2 \tag{14}$$

makes sure that exactly one of them takes the value 0. Then, the set of constraints

$$\begin{aligned} \Delta\text{dir}(u, v, w) &\leq -5 + \delta_1(u, v, w)M \\ \Delta\text{dir}(u, v, w) &\geq 5 - \delta_2(u, v, w)M \\ \Delta\text{dir}(u, v, w) &\leq 4 + \delta_3(u, v, w)M \\ \Delta\text{dir}(u, v, w) &\geq -4 - \delta_3(u, v, w)M \end{aligned} \tag{15}$$

together with

$$-\text{bend}(u, v, w) \leq \Delta\text{dir}(u, v, w) - 8\delta_1(u, v, w) + 8\delta_2(u, v, w) \leq \text{bend}(u, v, w) \tag{16}$$

assign the bend cost $\text{bend}(u, v, w)$ for the bend between edges $\{u, v\}$ and $\{v, w\}$, where the variable $\text{bend}(u, v, w)$ is integer valued and non-negative. Verify that these constraints in combination with the minimization of (13) indeed model the bend cost as defined in (12). For a detailed explanation we refer to [7].

Minimizing the number of bends thus uses four variables and seven constraints for each pair of incident edges on a path $L \in \mathcal{L}$. Since there are in total at most m' such pairs we are using $4m'$ variables and at most $7m'$ constraints.

3.6 Preservation of Edge Directions

To preserve as much of the overall appearance of the metro system as possible we have already restricted the edge directions to the set of the three directions closest to the original one in Sect. 3.1. Ideally we want to draw an edge $\{u, v\}$ using the closest octilinear approximation, i.e. the direction where $\text{dir}(u, v) =$

$\text{sec}_u(v)$. Hence we introduce a cost in case that the layout does not use this direction. This models (S3).

For each edge $\{u, v\}$ we define as its cost a binary variable $\epsilon(u, v)$ which is 0 if and only if $\text{dir}(u, v) = \text{sec}_u(v)$. This is modeled as follows

$$-M\epsilon(u, v) \le \text{dir}(u, v) - \text{sec}_u(v) \le M\epsilon(u, v) \qquad \forall\{u, v\} \in E. \tag{17}$$

Now we can define the edge-direction cost

$$\text{cost}_{\text{dir}} = \sum_{\{u,v\}\in E} \epsilon(u, v) \tag{18}$$

which, for each edge, charges 1 when the MIP does not choose the closest octilinear direction. This part of our formulation needs m variables and $2m$ constraints.

3.7 Speed-Up Techniques

A common feature of metro maps is that they tend to have a large number of degree-2 vertices on tracks between two interchange stations. It is useful and common in real metro maps to draw paths between pairs of neighboring interchange or terminal stations as straight as possible. This leads to the idea of replacing chains of degree-2 vertices temporarily by single edges and reinserting the vertices in the final drawing equidistantly on these edge. While this data-reduction trick has been applied before [6, 10], we extend it by keeping two vertices on each chain of degree-2 vertices. The rationale behind this is that it allows for drawing the connection between the corresponding interchange vertices as a polyline with three segments. Our experiments showed that this is a good compromise. Remember that the target function penalizes bends along lines so that in many cases bends at these special degree-2 vertices are in fact avoided.

The only part of our MIP formulation that needs a quadratic number of constraints (and variables) is the one that ensures planarity. This is why we suggest several ways to reduce the number of these constraints. For a planar drawing of an embedded graph it suffices to require that non-incident edges of the same face do not intersect. This already guarantees that no two edges intersect except at common endpoints. So instead of using the constraints in Sect. 3.3 for *all* pairs of non-incident edges we only include them for pairs of non-incident edges of the same face.

In many real-world examples (see Sect. 4) this is still not enough to solve the MIP in an acceptable amount of time. To further reduce the number of constraints we rely on heuristic methods that relax the planarity requirements. These heuristics involve subdividing the external face using the convex hull and considering only pairs of edges where at least one edge is a *pendant* edge, i.e. an edge that leads to a degree-1 vertex. One can also try to skip the planarity constraints completely. In some of these experiments the results were indeed planar in spite of not being enforced in the model. For more details see [7].

	G	n	m	MIP	all pairs	faces	CH	PE	CH & PE	none
Sydney	uncontr.	174	183	constr.	81416	45182	21983	13535	6242	3041
10 lines	contr.	62	71	var.	20329	11545	5921	3873	2105	1329

Table 1. Total number of constraints and variables for six different planarity tests

4 Experiments

In this section we show how our method performs on the metro system of Sydney because Sydney has been used as a benchmark before [6, 10]. For more examples, see [7]. We solved our MIP with the optimizer CPLEX 9.0 running on a Power3-II processor with 375 MHz under the UNIX operating system AIX 5.1, the only system with a CPLEX license accessible to us. We have also experimented with the optimizer XpressMP but found that CPLEX generates better results.

Table 4 shows the size of the uncontracted and contracted network and the number of constraints and variables for the different planarity options in the contracted case. The numbers in the columns *faces* and *none* show that ensuring planarity is in fact responsible for about 90% of the constraints and variables. The other columns show that the convex-hull (CH) and pendant-edge (PE) heuristics as well as their combination effectively reduce the MIP size.

The CityRail system in Sydney (which we restricted to the more interesting suburban part) is a relatively large network and has several multiple edges. The geographic layout is displayed in Fig. 4(a), the official metro map in Fig. 4(f). The weights used in the objective function were $(\lambda_{\text{length}}, \lambda_{\text{bends}}, \lambda_{\text{dir}}) = (1, 5, 5)$. Combining convex-hull and pendant-edge heuristic yielded the planar layout in Fig. 4(e) within 22 minutes. Observe the influence of the soft constraints on the layout: There are no unnecessarily long edges (optimization of (S2)). Moreover, the metro lines only bend where geographically required and pass through interchange stations as straight as possible (optimization of (S1)). And, finally, the simplified edges tend to follow the original directions of Fig. 4(a) (optimization of (S3)). These goals were optimized while guaranteeing octilinearity and preserving the original embedding.

We now compare our layout of the Sydney map to the results of previous algorithms. Figure 4(b) is taken from Hong et al. [6] and shows their layout using a special spring-embedder method. Originally they draw a network that extends slightly further into the periphery but these extensions should not influence the layout of the central part of the network. For ease of comparison we clipped the lines appropriately in Fig. 4(b). Apart from the fact that Hong at al. show station labels, one can observe that edges are not strictly octilinear and that avoiding bends along lines is not a goal of their method. In addition, there is a large variance in the distribution of the edge lengths. Figure 4(c), taken from Stott and Rodgers [10], shows their layout when applying an edge contraction step before actually drawing the network. There are two edges that obviously violate octilinearity, which is an important drawback of this layout. Figure 4(d) displays the result of the same method without prior edge contraction. It again shows an almost octilinear layout, now with the exception of one edge.

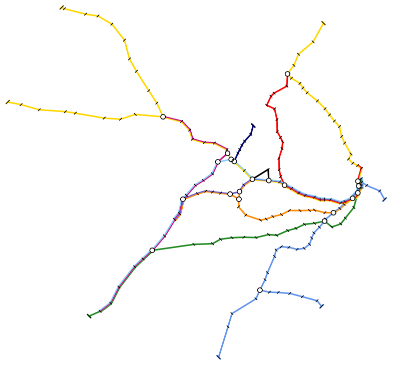
Our method overcomes the limitations of the previous results: there are no exceptions to octilinearity and we avoid the problems of local optima in [10]. In contrast to Hong et al. we actively minimize the number of line bends in the layout and maintain the overall geography using the concept of relative position.

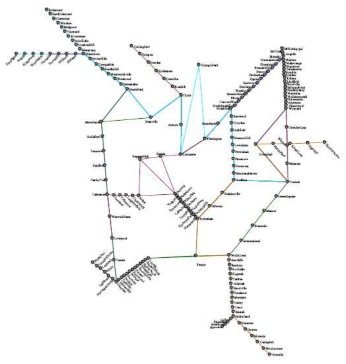
The disadvantage of our method is its running time. While we needed 22 minutes to produce our Sydney map, Hong et al. computed the layout in Fig. 4(b) within 7.6 seconds. Stott and Rodgers needed 4 minutes for a Sydney map using a contracted input graph (Fig. 4(c)) and about 28 minutes for the uncontracted graph (Fig. 4(d)). Experiments were carried out on very different machines.

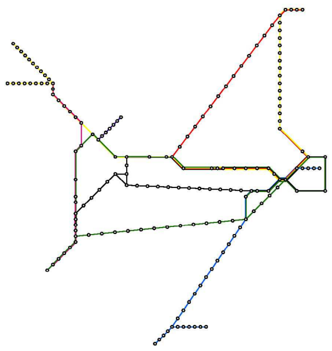
Acknowledgments. We thank Seok-Hee Hong and Herman Haverkort for interesting discussions and Damian Merrick for the Sydney data.

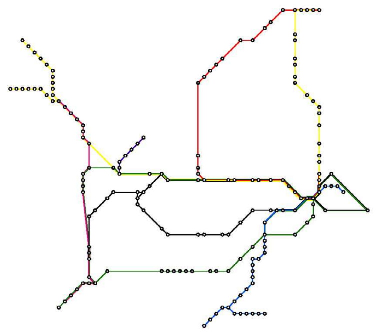
References

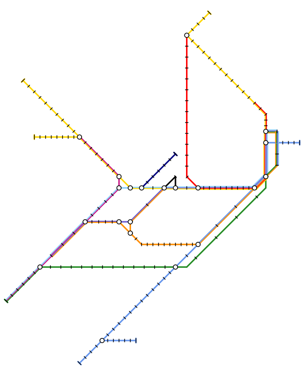
1. T. Barkowsky, L. J. Latecki, and K.-F. Richter. Schematizing maps: Simplification of geographic shape by discrete curve evolution. In C. Freksa, W. Brauer, C. Habel, and K. F. Wender, editors, *Proc. Spatial Cognition II*, volume 1849 of *Lecture Notes in Artificial Intelligence*, pages 41–53, 2000.
2. C. Binucci, W. Didimo, G. Liotta, and M. Nonato. Orthogonal drawings of graphs with vertex and edge labels. *Comp. Geometry: Theory & Appl.*, 32(2):71–114, 2005.
3. U. Brandes, M. Eiglsperger, M. Kaufmann, and D. Wagner. Sketch-driven orthogonal graph drawing. In *Proc. 10th Int. Symp. on Graph Drawing (GD'02)*, volume 2528 of *Lecture Notes in Computer Science*, pages 1–11. Springer-Verlag, 2002.
4. S. Cabello, M. d. Berg, S. v. Dijk, M. v. Kreveld, and T. Strijk. Schematization of road networks. In *Proc. 17th Annual Symp. on Computational Geometry (SoCG'01)*, pages 33–39, New York, 2001. ACM Press.
5. G. di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs.* Prentice Hall, 1999.
6. S.-H. Hong, D. Merrick, and H. A. D. d. Nascimento. The metro map layout problem. In J. Pach, editor, *Proc. 12th Int. Symp. on Graph Drawing (GD'04)*, volume 3383 of *Lecture Notes in Computer Science*, pages 482–491. Springer-Verlag, 2005.
7. M. Nöllenburg. Automated drawings of metro maps. Technical Report 2005-25, Universität Karlsruhe, 2005. Available at `http://www.ubka.uni-karlsruhe.de/cgi-bin/psview?document=/ira/2005/25`.
8. M. Ovenden. *Metro Maps of the World.* Capital Transport Publishing, 2003.
9. E. S. Sandvad, K. Grønbæk, L. Sloth, and J. L. Knudsen. A metro map metaphor for guided tours on the Web: the Webvise Guided Tour System. In *Proc. 10th Int. World Wide Web Conf. (WWW'01)*, p. 326–333, Hong Kong, 2001. ACM Press.
10. J. M. Stott and P. Rodgers. Metro map layout using multicriteria optimization. In *Proc. 8th Int. Conf. on Information Visualisation (IV'04)*, London, pages 355–362. IEEE, 2004.
11. Sydney CityRail. `www.cityrail.nsw.gov.au/networkmaps/network_map.pdf`.
12. R. Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM J. Comput.*, 16(3):421–444, 1987.


(a) Original geographic layout


(b) Hong et al. (clipping of Fig. 7(b) in [6])


(c) Stott and Rodgers using contracted edges (Fig. 14 in [10])


(d) Stott and Rodgers using uncontracted edges (Fig. 15 in [10])


(e) Final layout using our method

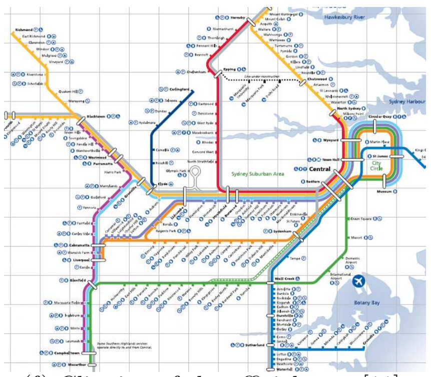

(f) Clipping of the official map [11]

Fig. 4. Various drawings of the Sydney CityRail system.