An Improved Algorithm for the Metro-Line Crossing Minimization Problem

Martin Nöllenburg

Fakultät für Informatik, Universität Karlsruhe (TH) and
Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
noellenburg@iti.uka.de

Abstract. In the metro-line crossing minimization problem, we are given a plane graph $G = (V, E)$ and a set $\mathcal{L}$ of simple paths (or *lines*) that *cover* G, that is, every edge $e \in E$ belongs to at least one path in $\mathcal{L}$. The problem is to draw all paths in $\mathcal{L}$ along the edges of G such that the number of crossings between paths is minimized. This crossing minimization problem arises, for example, when drawing metro maps, in which multiple transport lines share parts of their routes.
We present a new line-layout algorithm with $O(|\mathcal{L}|^2 \cdot |V|)$ running time that improves the best previous algorithms for two variants of the metro-line crossing minimization problem in unrestricted plane graphs. For the first variant, in which the so-called *periphery condition* holds and terminus side assignments are given in the input, Asquith et al. [1] gave an $O(|\mathcal{L}|^3 \cdot |E|^{2.5})$-time algorithm. For the second variant, in which all lines are paths between degree-1 vertices of G, Argyriou et al. [2] gave an $O((|E| + |\mathcal{L}|^2) \cdot |E|)$-time algorithm.

1 Introduction

Schematic *metro maps* are effective and popular visualizations of public transport networks all over the world; see Ovenden's comprehensive collection of metro maps [3]. Several methods for automatically drawing metro maps have been suggested in recent years [4–6]. These methods, however, focus on drawing the *underlying graph*, that is, the graph that represents stations as vertices and direct links between two stations as edges. This graph represents the infrastructure of the transport network, for example, railway tracks or roads. A schematic layout of the underlying graph, whether created manually or by one of the existing methods mentioned above, does not necessarily yield a proper metro map yet. The reason is that most real-world networks contain many different transport lines whose routes partially overlap, that is, some edges of the underlying graph are shared by multiple transport lines. In practice, each transport line is therefore drawn in a distinct color along the edges of its path in the underlying graph. Consequently, edges that belong to several lines consist in fact of a bundle of colored parallel curves. As an example, Fig. 1 shows a detail of the metro map of Cologne.

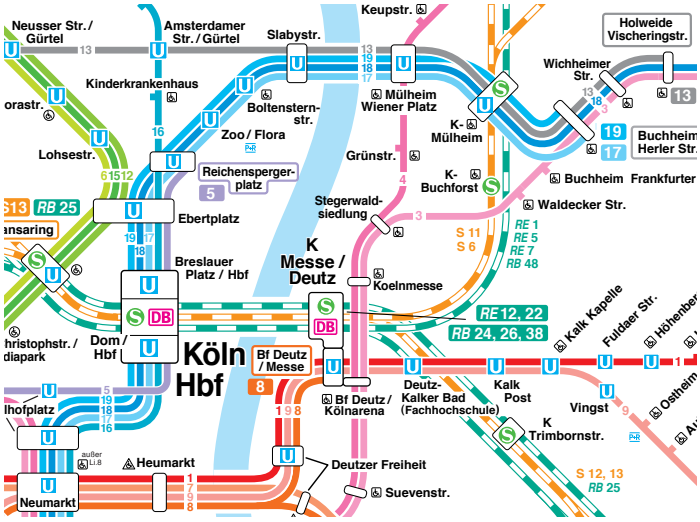
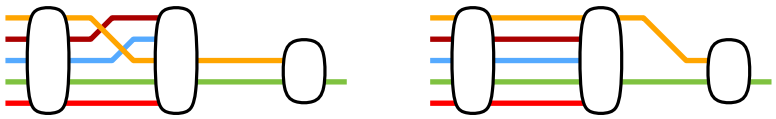


Fig. 1: Detail of the metro map of Cologne.

An immediate consequence of such a visualization is that there are situations in which two lines in the—otherwise plane—network cross. Some line crossings are mandatory, induced by the prescribed network topology, others depend on the line orders in each vertex and can be avoided by choosing the right orders. Hence, the *metro-line crossing minimization* (MLCM) problem arises as a secondary problem in the metro-map layout process: find an ordering of the parallel lines along each edge of the underlying graph such that as few pairs of lines as possible cross each other in the final layout. Additionally, the relative order of lines traversing a vertex in the same direction must not change within this vertex, that is, we do not allow to hide line crossings "below" the area occupied by the representation of a vertex. Note that the MLCM problem is independent of the actual layout of the underlying graph. The combinatorial embedding of the underlying graph, which is usually defined by its geographic input embedding, is all one needs to define the orderings of the parallel lines. Hence, algorithms for MLCM can be used both for reducing line crossings in existing layouts and, as a second step in combination with layout methods for the underlying graph, for creating metro maps from scratch.

Although we present our results in terms of the classic problem of visualizing transportation networks, we note that the metro map metaphor has also been used as a means to visualize potentially much larger networks in other fields, for example, metabolic pathways [7]. Actually, the MLCM problem appears whenever multiple parallel edges in a graph need to be drawn separately along a common geometric path with the minimum number of crossings among them.

Benkert et al. [8] introduced the general MLCM problem. Subsequently, MLCM was considered in several variants and for different classes of under-



(a) With periphery condition. (b) Without periphery condition.

Fig. 2: Layout of a terminus (middle vertex) with three terminating lines. The layout in (b) introduces a gap between the continuing lines.

lying graphs [1, 2, 9], which are discussed in detail in Section 3. One important variant, posed as an open problem by Benkert et al. [8] and addressed by Bekos et al. [9] and by Asquith et al. [1], restricts the positions of each line's start and end point (called *termini*) to be left- or rightmost in the ordered sequence of lines along the underlying edges leading to its termini. This restriction is called the *periphery condition* and prevents gaps between continuing lines, see Fig. 2. Gaps between parallel lines disrupt the uniform appearance of the underlying edge and hence are to be avoided in order to improve readability. Apart from avoiding gaps, an exposed outer position for terminating lines also allows for better highlighting or labeling of the termini. Often the final destination of a train or bus in a transport network is used to indicate its direction and hence termini and their names should be prominent features that are easy to recognize in a metro map. Many of the real-world maps in Ovenden's collection [3] adhere to the periphery condition, as does the metro map of Cologne in Fig. 1. Bekos et al. [9] showed that the MLCM problem is NP-hard under the periphery condition if each terminus can lie on either side of the respective final edge. On the other hand, Asquith et al. [1] showed that the MLCM problem under the periphery condition can be solved efficiently for general plane graphs if the terminus side assignment is given as part of the input.

In this paper, we investigate the MLCM problem under the periphery condition with terminus side assignments and present a new algorithm in Section 4 that solves this problem in $O(|\mathcal{L}|^2 \cdot |V|)$ time for a graph $G = (V, E)$ and a set of lines $\mathcal{L}$. The algorithm has two phases. First, for each pair of lines that share a common subpath, we determine their required relative order at the end of their common subpath. Then, in a second step, we iteratively insert one line at a time into the layout such that the relative orders computed in the first phase are respected and no unnecessary line crossings are created. Our algorithm improves the algorithm of Asquith et al. [1] for the same problem, which has a running time of $O(|\mathcal{L}|^3 \cdot |E|^{2.5})$. Our algorithm can also be used to solve a closely related problem considered by Argyriou et al. [2], where all lines must be paths connecting two degree-1 vertices in G. Hence, it also improves the algorithm of Argyriou et al., which has a running time of $O((|E| + |\mathcal{L}|^2) \cdot |E|)$. These are the only two variants of MLCM that are known to be efficiently solvable, and our algorithm is to the best of our knowledge currently the fastest method to solve both of them for general plane underlying graphs.

2 Model

The input to the MLCM problem is a *metro graph* $(G, \mathcal{L})$, where $G = (V, E)$ is a planar embedded graph and $\mathcal{L}$ is a *line cover* of G, that is, a set of simple paths (or *lines*) that cover G. Note that existing edge crossings in the input graph can easily be modeled as dummy vertices. For notational convenience, we consider each undirected edge $\{u, v\} \in E$ as a pair of directed edges uv and vu. Both notations refer to the same single edge just from two different perspectives.

The vertices v_0 and v_k of a line $\ell = (v_0, v_1, \ldots, v_k) \in \mathcal{L}$ of length $|\ell| = k$ are called the *termini* of ℓ, the vertices $v_1, \ldots, v_{k-1}$ are called *intermediate vertices* of ℓ. An edge uv is included in a line ℓ, in short $uv \in \ell$, if u and v are consecutive vertices in ℓ. We denote as $\mathcal{L}_{uv} = \mathcal{L}_{vu} = \{\ell \in \mathcal{L} \mid uv \in \ell\}$ the set of all lines that include an edge uv. The *total edge size* of $\mathcal{L}$ is defined as $N_{\mathcal{L}} = \sum_{\ell \in \mathcal{L}} |\ell| = \sum_{uv \in E} |\mathcal{L}_{uv}|$. Note that $N_{\mathcal{L}} \in O(|\mathcal{L}| \cdot |V|)$ since $|\ell| \leq |V|$ for each line $\ell \in \mathcal{L}$.

Each vertex u has a cyclic sequence of $\sum_{uv \in E} |\mathcal{L}_{uv}|$ consecutive *ports*, one for each line of each incident edge uv. Each port is a point on the boundary of the geometric representation of u, at which the individual lines in $\bigcup_{uv \in E} \mathcal{L}_{uv}$ enter (or leave) u. We are interested in the order in which the lines in $\mathcal{L}_{uv}$ connect to the consecutive subsequence of ports of u (and of v) that correspond to the lines along edge uv. So for each edge $uv \in E$, we define two *line orders* $<^u_{uv}$ and $<^v_{uv}$ of $\mathcal{L}_{uv}$ in the endpoints of uv. For two lines ℓ_1 and ℓ_2 in $\mathcal{L}_{uv}$ we write $\ell_1 <^u_{uv} \ell_2$ (or $\ell_1 <^v_{uv} \ell_2$) if ℓ_1 is right of ℓ_2 at the endpoint u (or v) with respect to the direction of uv. Note that the orders are reversed if we use the oppositely directed edge vu instead of uv, that is, $\ell_1 <^u_{uv} \ell_2$ if and only if $\ell_2 <^u_{vu} \ell_1$. The sorted sequence of the lines in $\mathcal{L}_{uv}$ with respect to $<^u_{uv}$ is denoted as s^u_{uv}; analogously s^v_{uv} is the sorted sequence of lines with respect to $<^v_{uv}$. Again, the sequences s^u_{vu} and s^v_{vu} are the reversed sequences of s^u_{uv} and s^v_{uv}.

A *line crossing* is a crossing between two lines ℓ_1 and ℓ_2 along a shared edge uv. The two lines cross on uv if $\ell_1 <^u_{uv} \ell_2$ and $\ell_2 <^v_{uv} \ell_1$ or vice versa. Abstracting from geometry, the number of line crossings along an edge uv is thus equal to the number of inversions in the sequences s^u_{uv} and s^v_{uv}.

In order to avoid confusion for the map viewer, it is not allowed to hide line crossings "below" a vertex. To that end we define a line order $<^v_{uv}$ to be *compatible* with the vertex v if the following holds. Apart from uv, let $vw_1, vw_2, \ldots, vw_k$ be the other edges incident to v in counterclockwise order starting from uv. We consider the sequence s^v_{uv} and the concatenated sequence $s' = \prod_{i=1}^{k} s^v_{vw_i}$. Then $<^v_{uv}$ is compatible with v if s^v_{uv} is a subsequence of s'. In other words, the lines that enter v through the edge uv and leave v through the edges $vw_1, vw_2, \ldots, vw_k$ do not change their relative order. We say that a vertex v is *admissible* if the line orders for all incident edges are compatible with v.

3 MLCM variants and previous work

In this section we present four different variants of the MLCM problem that have been considered in the literature so far. Previous results and the results obtained in this paper are summarized in Table 1.

problem	graph class	restrictions	result	reference
MLCM	single edge uv	–	$O(\lvert\mathcal{L}_{uv}\rvert^2)$	[8]
MLCM-P	path	–	NP-hard	[9]
	plane graph	–	ILP + MLCM-PA	[1]
MLCM-PA	path	2-side model	$O(\lvert\mathcal{L}\rvert \cdot \lvert V\rvert)$	[9]
	left-to-right tree	2-side model	$O(\lvert\mathcal{L}\rvert \cdot \lvert V\rvert)$	[9]
	plane graph	–	$O(\lvert\mathcal{L}\rvert^3 \cdot \lvert E\rvert^{2.5})$	[1]
	plane graph	2-side model	$O(\lvert V\rvert \cdot (\lvert E\rvert + \lvert\mathcal{L}\rvert))$	[2]
	plane graph	–	$O(\lvert\mathcal{L}\rvert^2 \cdot \lvert V\rvert)$	Theorem 1
MLCM-T1	left-to-right tree	2-side model	$O(\lvert\mathcal{L}\rvert \cdot \lvert V\rvert)$	[9]
	plane graph	2-side model	$O(\lvert V\rvert \cdot (\lvert E\rvert + \lvert\mathcal{L}\rvert))$	[2]
	plane graph	–	$O((\lvert E\rvert + \lvert\mathcal{L}\rvert^2) \cdot \lvert E\rvert)$	[2]
	plane graph	–	$O(\lvert\mathcal{L}\rvert^2 \cdot \lvert V\rvert)$	Corollary 1

Table 1: Overview of results for the MLCM problem and its variants. Algorithmic results are given by their running time.

The original *metro-line crossing minimization* problem as introduced by Benkert et al. [8] is as follows.

Problem 1 (MLCM). Given a metro graph $(G = (V, E), \mathcal{L})$, find for each edge $uv \in E$ two line orders $<_{uv}^u$ and $<_{uv}^v$ of the lines in $\mathcal{L}_{uv}$ such that the number of line crossings is minimal and all vertices are admissible.

A solution to MLCM is denoted as a *line layout*. Benkert et al. [8] gave a quadratic-time algorithm to solve MLCM for a single edge of G. Their algorithm does not extend to larger subgraphs and it is a remaining open problem whether MLCM is NP-hard in its general form.

We have already introduced the periphery condition, which additionally requires that each line terminates in an outer or *peripheral* position in each of its two termini (recall Fig. 2). Formally, this means that for each vertex v and each edge uv all lines in $\mathcal{L}_{uv}$, for which v is a terminus, must be placed in the beginning or in the end of the sequence s_{uv}^v. In other words, no terminating line can lie between two continuing lines in the order $<_{uv}^v$. We denote the following variant as MLCM *with periphery condition* (MLCM-P).

Problem 2 (MLCM-P). Given a metro graph $(G = (V, E), \mathcal{L})$, find for each edge $uv \in E$ two line orders $<_{uv}^u$ and $<_{uv}^v$ of the lines in $\mathcal{L}_{uv}$ such that the number of line crossings is minimal, all vertices are admissible, and each terminating line is placed at a peripheral position in each of its two termini.

Bekos et al. [9] showed that MLCM-P is NP-hard, even if G is a path, and Asquith et al. [1] formulated an integer linear program (ILP) to solve MLCM-P. Still, Problem 2 gives rise to a closely related (but computationally feasible) variant that additionally specifies in the input fixed terminus sides for each line. We denote this variant as MLCM *with periphery condition and terminus side assignments* (MLCM-PA).

Problem 3 (MLCM-PA). Given a metro graph $(G = (V, E), \mathcal{L})$ and terminus side assignments for all lines in $\mathcal{L}$, find for each edge $uv \in E$ two line orders $<_{uv}^{u}$ and $<_{uv}^{v}$ of the lines in $\mathcal{L}_{uv}$ such that the number of line crossings is minimal, all vertices are admissible, and each terminating line is placed at a peripheral position on the specified side of each of its two termini.

Problem 3 occurs in situations, in which, for example, the physical location of the tracks or the bus stop of the terminating line in a terminus yields this information. Alternatively, the optimal terminus side assignments can be obtained from the ILP formulation of Asquith et al. [1]. Asquith et al. also presented an $O(|\mathcal{L}|^3 \cdot |E|^{2.5})$-time algorithm to solve MLCM-PA for general plane graphs. Bekos et al. [9] gave two algorithms to solve MLCM-PA in the restricted *2-side model* for paths and for a special class of left-to-right directed trees with bounded vertex degree in $O(|\mathcal{L}| \cdot |V|)$ time, respectively. In the 2-side model, all vertices are drawn as rectangles and all lines are drawn as x-monotone paths that pass through vertices from the left to the right side. Argyriou et al. [2] recently presented an algorithm to solve MLCM-PA in the 2-side model for general plane graphs in $O(|V| \cdot (|E| + |\mathcal{L}|))$ time.

Another interesting MLCM variant restricts the lines in $\mathcal{L}$ to terminate at degree-1 vertices only, that is, all termini in $(G, \mathcal{L})$ are leaves of G.

Problem 4 (MLCM-T1). Given a metro graph $(G = (V, E), \mathcal{L})$ in which the degree of any terminus v of any path in $\mathcal{L}$ equals 1, find for each edge $uv \in E$ two line orders $<_{uv}^{u}$ and $<_{uv}^{v}$ of the lines in $\mathcal{L}_{uv}$ such that the number of line crossings is minimal and all vertices are admissible.

Problem 4 is of practical interest since in many real-world networks transport lines lead from one terminus station in the outskirts of a city through the city center to another terminus station in the outskirts. This is exactly the situation in which lines terminate at leaves of the underlying graph. Argyriou et al. [2] presented an algorithm to solve MLCM-T1 in general plane graphs in $O((|E| + |\mathcal{L}|^2) \cdot |E|)$ time. For MLCM-T1 in the previously mentioned 2-side model, they improved the running time to $O((|E| + |\mathcal{L}|) \cdot |V|)$.

We observe that a line layout for an MLCM-T1 instance trivially satisfies the periphery condition. Since each terminus v is a degree-1 vertex in G, there cannot be any continuing lines in v, and any position in the line order at v is peripheral by definition. Furthermore, there is no need to distinguish two different sides for the assignment of the terminus positions: not being separated by a continuing line, the two sides of the edge leading to v coincide. Hence, we can reduce any MLCM-T1 instance to an equivalent MLCM-PA instance by assigning all lines that terminate at the same leaf v to the same terminus side. This actually means that there is no restriction to the line order in v at all, and we indeed model the general setting of MLCM-T1. Obviously, the reduction takes only linear time. This is summarized in the following lemma.

Lemma 1. *An instance of MLCM-T1 can be reduced to an equivalent instance of MLCM-PA in linear time.*

4 An improved algorithm for MLCM-PA

In this section we present our main result, an $O(|\mathcal{L}|^2 \cdot |V|)$-time algorithm for MLCM-PA and MLCM-T1 in general plane graphs. We first show a simple lemma about the line crossings in an optimal layout for an MLCM-PA instance. We define a line crossing of two lines in a metro graph $(G, \mathcal{L})$ to be *unavoidable*, if it is present in any line layout of $(G, \mathcal{L})$.

Lemma 2. *Given a metro graph $(G, \mathcal{L})$ and terminus side assignments for all lines in $\mathcal{L}$, all line crossings in a crossing-minimal line layout are unavoidable crossings.*

Proof. By definition every unavoidable crossing is present in any crossing-minimal line layout. We want to show that the opposite is also true: every line crossing in a crossing-minimal line layout is unavoidable.

So let ℓ_1 and ℓ_2 be two lines that cross in a crossing-minimal line layout along an edge uv. By $P = (w_0, \ldots, w_i = u, w_{i+1} = v, \ldots, w_k)$, $0 \leq i < k$, we denote the maximal common subpath of ℓ_1 and ℓ_2 that contains uv. First of all note that the crossing along uv is the only crossing of ℓ_1 and ℓ_2 along P; any two consecutive crossings of two lines along a common subpath could be removed by routing the upper line just below the lower line along the edges between the two crossings—this contradicts the optimality of the line layout and has been observed by Asquith et al. [1] before.

We can assume that $\ell_1 <^u_{uv} \ell_2$ and $\ell_2 <^v_{uv} \ell_1$. Since there is a single crossing between ℓ_1 and ℓ_2 along P, this implies that $\ell_1 <^{w_0}_{w_0 w_1} \ell_2$ and $\ell_2 <^{w_k}_{w_{k-1} w_k} \ell_1$. This inversion of ℓ_1 and ℓ_2 in the line orders of vertices w_0 and w_k is either enforced by the combinatorial embedding of G as the line orders $<^{w_0}_{w_0 w_1}$ and $<^{w_k}_{w_{k-1} w_k}$ must be compatible with w_0 and w_k (if the respective line continues beyond w_0 or w_k) or by the given terminus side assignment (if the respective line terminates at w_0 or w_k). The only case where the relative order of ℓ_1 and ℓ_2 is not fixed by the compatibility requirements or the terminus side assignments is if both lines terminate at the same vertex, say w_0, and are assigned to the same terminus side. In that case, however, they can always be reordered in $<^{w_0}_{w_0 w_1}$ such that they reflect their relative order in $<^{w_k}_{w_{k-1} w_k}$ and the crossing would disappear. This contradicts the optimality of the layout.

We conclude that the crossing of ℓ_1 and ℓ_2 is unavoidable: the relative order of ℓ_1 and ℓ_2 at one end of P is the inverse of their order at the other end of P due to the given terminus side assignments or the compatibility requirements for the embedding of G. □

Lemma 2 implies that there is a line layout that realizes exactly the unavoidable crossings and, consequently, that any such layout is optimal. Algorithm 1 constructs such a line layout. It first computes all maximal common subpaths of all pairs of lines to determine their relative orders as induced by the topology or the terminus side assignments. In a second phase all lines are iteratively inserted into the line orders of their edges and the final line layout is fixed.

Algorithm 1: MLCM-PA line layout

Input: metro graph $(G, \mathcal{L})$, terminus side assignments for all $\ell \in \mathcal{L}$
Output: line orders $<^u_{uv}, <^v_{uv}$ for all edges $uv \in E$

```
/* Phase 1                                                              */
foreach (ℓ1, ℓ2) ∈ L × L, ℓ1 ≠ ℓ2, ℓ1 = (v0, v1, ..., vk) do
    compute set Λ(ℓ1, ℓ2) of all maximal common subpaths of ℓ1 and ℓ2
    foreach (vi, vi+1, ..., vj) ∈ Λ(ℓ1, ℓ2) do
        if ℓ2 leaves ℓ1 towards the left or terminates left of ℓ1 in vj then
            for l = i to j − 1 do
                side(ℓ1, ℓ2, vl vl+1) ← left
        else
            for l = i to j − 1 do
                side(ℓ1, ℓ2, vl vl+1) ← right
/* Phase 2                                                              */
foreach ℓ = (v0, v1, ..., vk) ∈ L do
    for i = 0 to k − 1 do
        insert ℓ into <^{vi}_{vi vi+1}
        insert ℓ into <^{vi+1}_{vi vi+1}
```

Theorem 1. *Given a metro graph $(G, \mathcal{L})$ and terminus side assignments for all lines in $\mathcal{L}$, Algorithm 1 computes a crossing-minimal line layout under the periphery condition in $O(|\mathcal{L}| \cdot N_{\mathcal{L}})$ time.*

Proof. In Phase 1 of Algorithm 1 we compute the value of a binary variable $\text{side}(\ell_1, \ell_2, uv)$ for each triple of two lines ℓ_1 and ℓ_2 and an edge uv such that uv is a common edge of ℓ_1 and ℓ_2. This value represents the side to which line ℓ_2 tends with respect to ℓ_1 on edge uv. So if $\text{side}(\ell_1, \ell_2, uv) = \textit{left}$ (*right*), we know that at the end of the maximal common subpath of ℓ_1 and ℓ_2 that contains uv the line ℓ_2 must be placed left (right) of ℓ_1.

In order to compute the set $\Lambda(\ell_1, \ell_2)$ of maximal common subpaths of ℓ_1 and ℓ_2 we walk along $\ell_1 = (v_0, \ldots, v_k)$ and check for each edge $v_i v_{i+1}$ whether ℓ_2 shares that edge with ℓ_1. If this is the case, we either open a new subpath or extend the current subpath. Otherwise we close the current subpath if there is one. We assume that the input $(G, \mathcal{L})$ contains a Boolean edge-line array of size $|E| \times |\mathcal{L}|$ so that we can check whether a line uses an edge in constant time.

For each subpath $\lambda = (v_i, v_{i+1}, \ldots, v_j) \in \Lambda(\ell_1, \ell_2)$ we need to determine whether ℓ_2 *tends* left- or rightward along λ with respect to ℓ_1, that is, whether at the end of λ the line ℓ_2 must be left or right of ℓ_1. There are three cases to consider.

(1) If $v_j = v_k$, that is, ℓ_1 terminates in v_j, and ℓ_2 does not terminate in v_j, then ℓ_2 tends leftward (rightward) if ℓ_1 is assigned a right (left) terminus position, respectively.

(2) If $v_j = v_k$ and ℓ_2 also terminates in v_j, then either ℓ_1 and ℓ_2 are assigned to different terminus sides and ℓ_2 tends to its assigned side, or both are assigned to the same side. In the latter case, ℓ_2 shall stay on the same side of ℓ_1 as in the first vertex v_i of λ. So if ℓ_2 enters v_i to the left of ℓ_1, then ℓ_2 also tends leftward along λ; otherwise it tends rightward.
(3) If $v_j \neq v_k$ then ℓ_2 tends leftward if either ℓ_2 is assigned to terminate on the left in v_j or ℓ_2 continues along an edge $v_j w$ that is left of ℓ_1 in the embedding of the underlying graph G; otherwise ℓ_2 tends rightward.

In all three cases the value of $\text{side}(\ell_1, \ell_2, uv)$ is either an immediate consequence of the lines' terminus assignments or can be determined by a constant-time query for the relative order of three incident edges in the embedding of G.

Summarizing the above, Phase 1 takes $O(|\mathcal{L}| \cdot N_{\mathcal{L}})$ time and space since we check for each edge of each line if any of the other lines in $\mathcal{L}$ share the edge; if this is the case we assign the leftward/rightward value to the corresponding variable.

In Phase 2 the actual line layout is computed by iteratively fixing the course of each line. We show the correctness of the algorithm by maintaining two invariants during Phase 2.

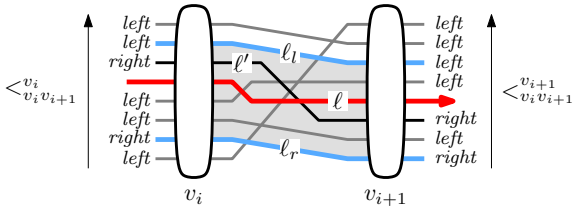
Invariant 1 There are no invalid intra-vertex crossings, that is, for each vertex u and each edge uv the line order $<_{uv}^{u}$ is compatible with u.
Invariant 2 All line crossings are unavoidable crossings with respect to the input embedding of G and the given terminus side assignments.

Inserting the first line as the only line into the empty line orders clearly satisfies both invariants. So assume that we already have a partial line layout that satisfies the invariants and that we want to insert the next line $\ell = (v_0, v_1, \ldots, v_k)$ into this partial layout.

We start by inserting ℓ into the order $<_{v_0 v_1}^{v_0}$. Let's assume ℓ is assigned to a left terminus in v_0 with respect to the first edge $v_0 v_1$ (for a right terminus the insertion is analogous). If ℓ is currently the only line with a left terminus on this edge, we insert ℓ as the last edge into $<_{v_0 v_1}^{v_0}$. Otherwise we scan the lines with a left terminus in $<_{v_0 v_1}^{v_0}$, starting with the largest (or leftmost) element, for the first line ℓ' for which $\text{side}(\ell, \ell', v_0 v_1) = \textit{right}$. We insert ℓ into $<_{v_0 v_1}^{v_0}$ immediately after (or left) of ℓ'. This first insertion does not create any intra-vertex crossings, so Invariant 1 is clearly satisfied. Furthermore, if there are multiple lines terminating along $v_0 v_1$ on the same side as ℓ then ℓ is inserted exactly between those lines that tend leftward and those lines that tend rightward with respect to ℓ. Hence all those lines are already on the correct side of ℓ and no line crossings are created; Invariant 2 is satisfied.

Next, we consider inserting ℓ into the order $<_{v_i v_{i+1}}^{v_i}$ for $i > 0$ such that Invariant 1 is satisfied. If one of the neighboring lines in the previous line order $<_{v_{i-1} v_i}^{v_i}$ also continues along $v_i v_{i+1}$, then ℓ simply keeps its position directly next to that line. Since the previous layout did not contain any invalid intra-vertex crossings and ℓ follows a previous line, Invariant 1 is still satisfied. This case is illustrated in Figure 3a, where the red line ℓ_1 follows the neighboring black line through the

(a) Lines ℓ_1 and ℓ_2 are inserted so that Invariant 1 is maintained.



(b) Line ℓ is inserted into $<^{v_{i+1}}_{v_iv_{i+1}}$ so that Invariant 2 is maintained. The values $\text{side}(\ell, \cdot, v_iv_{i+1})$ are indicated for all lines.

Fig. 3: Insertion of lines into an existing partial line layout.

vertex. Otherwise, if ℓ is the only line continuing along v_iv_{i+1}, we scan $<^{v_i}_{v_iv_{i+1}}$, starting with the smallest (rightmost) element, for the first line ℓ', whose previous edge wv_i is left of ℓ in the embedding of G or that terminates in v_i with a left terminus along v_iv_{i+1}. We insert ℓ immediately before ℓ' in $<^{v_i}_{v_iv_{i+1}}$. This is illustrated in Figure 3a by the blue line ℓ_2 which is inserted immediately before the yellow line ℓ'. If no line ℓ' is found then ℓ becomes the largest (leftmost) element in $<^{v_i}_{v_iv_{i+1}}$. The chosen position for ℓ ensures that $<^{v_i}_{v_iv_{i+1}}$ remains compatible with v_i and that Invariant 1 is satisfied.

It remains to determine the position of ℓ in the order $<^{v_{i+1}}_{v_iv_{i+1}}$. Figure 3b illustrates the situation. We scan the already determined line order $<^{v_i}_{v_iv_{i+1}}$ for the smallest (rightmost) line ℓ_l left of ℓ for which $\text{side}(\ell, \ell_l, v_iv_{i+1}) = \textit{left}$ and for the largest (leftmost) line ℓ_r right of ℓ, for which $\text{side}(\ell, \ell_r, v_iv_{i+1}) = \textit{right}$. Note that it is possible that one or both lines ℓ_l and ℓ_r do not exist. If they exist, these two lines ℓ_l and ℓ_r are the closest lines to ℓ that are already on the correct side. Since Invariant 2 holds for the previous partial layout, ℓ_l and ℓ_r do not cross each other along v_iv_{i+1}, that is, $\ell_r <^{v_i}_{v_iv_{i+1}} \ell_l$ and $\ell_r <^{v_{i+1}}_{v_iv_{i+1}} \ell_l$. Obviously, ℓ may not cross either of them and we must insert ℓ between ℓ_r and ℓ_l in $<^{v_{i+1}}_{v_iv_{i+1}}$ (otherwise Invariant 2 will be violated). More precisely, we insert ℓ immediately left of the largest (leftmost) line ℓ' in the interval $[\ell_r, \ell_l]$ of $<^{v_{i+1}}_{v_iv_{i+1}}$ for which $\text{side}(\ell, \ell', v_iv_{i+1}) = \textit{right}$, see Figure 3b. If ℓ_r (ℓ_l) does not exist we symbolically assign $\ell_r = -\infty$ ($\ell_l = \infty$) so that the interval $[\ell_r, \ell_l]$ may become unbounded. The position of ℓ is determined as before. If there is no line ℓ' then ℓ becomes the rightmost line in $<^{v_{i+1}}_{v_iv_{i+1}}$.

We claim that in the assigned position ℓ crosses only lines that were to its left and tend to the right or lines that were to its right and tend to the left—crossings that are unavoidable. Assume to the contrary that ℓ crosses a line $\hat{\ell}$ that was to its left and also tends to the left. Since we insert ℓ immediately to the left of ℓ', the two lines $\hat{\ell}$ and ℓ' also cross each other. This is a contradiction to Invariant 2 for the previous partial layout, though, since $\hat{\ell}$ crosses ℓ' from left to right but eventually needs to cross ℓ' again from right to left in order to reach its leftward destination. If there is no line ℓ' then ℓ is the rightmost

line in $<^{v_{i+1}}_{v_i v_{i+1}}$ by definition and cannot cross $\hat{\ell}$. Similarly, assume that ℓ crosses a line $\tilde{\ell}$ that was to its right and also tends to the right. Then $\ell_l <^{v_{i+1}}_{v_i v_{i+1}} \tilde{\ell}$ since otherwise we would have placed ℓ left of $\tilde{\ell}$ in the interval $[\ell_r, \ell_l]$. But this means that $\tilde{\ell}$ crosses ℓ_l from right to left, which again violates Invariant 2 for the previous partial layout: there must be a second crossing, where $\tilde{\ell}$ crosses ℓ_l from left to right in order to reach its rightward destination. If $\ell_l = \infty$ we would have placed ℓ left of $\tilde{\ell}$ which is also a contradiction. So Invariant 2 holds for the selected position of ℓ.

Finally, we show that Invariant 1 holds for the position of ℓ in $<^{v_{i+1}}_{v_i v_{i+1}}$. The first possibility for a violation is a line $\hat{\ell}$ with $\text{side}(\ell, \hat{\ell}, v_i v_{i+1}) = \textit{left}$ that is still to the right of ℓ but does not continue further along $v_{i+1}v_{i+2}$. By definition $\hat{\ell}$ can only be right of ℓ if $\hat{\ell} <^{v_{i+1}}_{v_i v_{i+1}} \ell'$. But then Invariant 1 would have been violated before by $\hat{\ell}$ and ℓ'. The other possibility for a violation of Invariant 1 is a line $\tilde{\ell}$ with $\text{side}(\ell, \tilde{\ell}, v_i v_{i+1}) = \textit{right}$ that is still to the left of ℓ but does not continue further along $v_{i+1}v_{i+2}$. By definition this can only be the case if $\ell_l <^{v_{i+1}}_{v_i v_{i+1}} \tilde{\ell}$. But this means that Invariant 1 would have been violated before by $\tilde{\ell}$ and ℓ'.

Since both invariants hold at the end of Algorithm 1, we have proven its correctness. By Invariant 1 all vertices are admissible, and by Invariant 2 the final line layout realizes exactly the unavoidable crossings and is thus crossing-minimal by Lemma 2. The running time of Phase 1 is $O(|\mathcal{L}| \cdot N_{\mathcal{L}})$. The running time of Phase 2 is again $O(|\mathcal{L}| \cdot N_{\mathcal{L}})$ since there are $2N_{\mathcal{L}}$ insertion operations, each of which determines a position for the current line by scanning the line orders of size $O(|\mathcal{L}|)$ of the current edge. □

We note that the size of a solution for MLCM-PA is $\Omega(N_{\mathcal{L}})$ and thus the running time of our algorithm is only a factor of $|\mathcal{L}|$ away from the output size. Since for the total edge size $N_{\mathcal{L}}$ we have $N_{\mathcal{L}} \in O(|\mathcal{L}| \cdot |V|)$, the running time of Algorithm 1 can also be expressed as $O(|\mathcal{L}|^2 \cdot |V|)$.

By Lemma 1, we can reduce any instance of MLCM-T1 to an equivalent instance of MLCM-PA in linear time. We thus obtain the following corollary.

Corollary 1. *Given a metro graph $(G, \mathcal{L})$ in which the degree of any terminus v of any line in $\mathcal{L}$ equals 1, we can use Algorithm 1 to compute a crossing-minimal line layout in $O(|\mathcal{L}| \cdot N_{\mathcal{L}})$ time.*

5 Conclusions

In this paper we have presented a new algorithm that improves the best previous algorithms for both the MLCM-PA and the MLCM-T1 problem. The running time of the new algorithm is $O(|\mathcal{L}| \cdot N_{\mathcal{L}})$, where $N_{\mathcal{L}} \in O(|\mathcal{L}| \cdot |V|)$.

We conclude with two observations about practical MLCM instances as found, for example, in Ovenden's book [3]. First, the number of lines $|\mathcal{L}|$ in a transport network is usually much smaller than the size of the underlying graph G. Since the output size is already $\Omega(N_{\mathcal{L}})$, our algorithm runs in linear

time if the number of lines is constant. Second, many lines in practice indeed terminate at degree-1 vertices of the underlying graph as modeled in the MLCM-T1 variant. Still, most networks also have some lines that start or end in non-leaf vertices. We therefore suggest to use the ILP formulation of Asquith et al. [1] (or a simple exhaustive-search algorithm) to determine an optimal terminus side assignment for those lines. We can then transform the original MLCM-P instance together with the additional terminus side assignments into an MLCM-PA instance that can be solved efficiently with our algorithm.

There are a few remaining open problems in MLCM. First of all, it is still an unsolved question whether the general MLCM problem (without periphery condition) is NP-hard for general plane graphs or even for paths. Another interesting open question is whether the NP-hard problem MLCM-P is fixed-parameter tractable for a suitable small parameter, such as the maximum multiplicity of the edges. Furthermore, no approximation algorithms for MLCM-P are known so far.

Acknowledgments. We thank Joachim Gudmundsson, Damian Merrick, and Thomas Wolle for initial discussions about the problem during a visit in Sydney.

References

1. Asquith, M., Gudmundsson, J., Merrick, D.: An ILP for the metro-line crossing problem. In Harland, J., Manyem, P., eds.: Proc. 14th Computing: The Australasian Theory Symp. (CATS'08). Volume 77 of CRPIT., Australian Comput. Soc. (2008) 49–56
2. Argyriou, E., Bekos, M.A., Kaufmann, M., Symvonis, A.: Two polynomial time algorithms for the metro-line crossing minimization problem. In Tollis, I.G., Patrignani, M., eds.: Proc. 16th Internat. Symp. Graph Drawing (GD'08). Volume 5417 of Lecture Notes Comput. Sci., Springer-Verlag (2009) 336–347
3. Ovenden, M.: Metro Maps of the World. Capital Transport Publishing (2003)
4. Stott, J.M., Rodgers, P.: Metro map layout using multicriteria optimization. In: Proc. 8th Internat. Conf. Information Visualisation (IV'04), IEEE (2004) 355–362
5. Hong, S.H., Merrick, D., do Nascimento, H.A.D.: Automatic visualization of metro maps. J. Visual Languages and Computing **17** (2006) 203–224
6. Nöllenburg, M., Wolff, A.: A mixed-integer program for drawing high-quality metro maps. In Healy, P., Nikolov, N.S., eds.: Proc. 13th Internat. Symp. Graph Drawing (GD'05). Volume 3843 of Lecture Notes Comput. Sci., Springer-Verlag (2006) 321–333
7. Hahn, W.C., Weinberg, R.A.: A subway map of cancer pathways (2002) Poster in Nature Reviews Cancer.
8. Benkert, M., Nöllenburg, M., Uno, T., Wolff, A.: Minimizing intra-edge crossings in wiring diagrams and public transportation maps. In Kaufmann, M., Wagner, D., eds.: Proc. 14th Internat. Symp. Graph Drawing (GD'06). Volume 4372 of Lecture Notes Comput. Sci., Springer-Verlag (2007) 270–281
9. Bekos, M.A., Kaufmann, M., Potika, K., Symvonis, A.: Line crossing minimization on metro maps. In Hong, S.H., Nishizeki, T., Quan, W., eds.: Proc. 15th Internat. Symp. Graph Drawing (GD'07). Volume 4875 of Lecture Notes Comput. Sci., Springer-Verlag (2008) 231–242