Metro-Line Crossing Minimization: Hardness, Approximations, and Tractable Cases

Martin Fink[1] and Sergey Pupyrev[2,3⋆]

[1] Lehrstuhl für Informatik I, Universität Würzburg, Germany.
[2] Department of Computer Science, University of Arizona, USA.
[3] Institute of Mathematics and Computer Science, Ural Federal University, Russia.

Abstract. Crossing minimization is one of the central problems in graph drawing. Recently, there has been an increased interest in the problem of minimizing crossings between paths in drawings of graphs. This is the *metro-line crossing minimization* problem (MLCM): Given an embedded graph and a set L of simple paths, called *lines*, order the lines on each edge so that the total number of crossings is minimized. So far, the complexity of MLCM has been an open problem. In contrast, the problem variant in which line ends must be placed in outermost position on their edges (MLCM-P) is known to be NP-hard.
Our main results answer two open questions: (i) We show that MLCM is NP-hard. (ii) We give an $O(\sqrt{\log |L|})$-approximation algorithm for MLCM-P.

1 Introduction

In metro maps and transportation networks, some edges, that is, railway tracks or road segments, are used by several lines. Usually, lines that share an edge are drawn individually along the edge in distinct colors; see Fig. 1. Often, some lines must cross, and one normally wants to have as few crossings of metro lines as possible. In the *metro-line crossing minimization* problem (MLCM), the goal is to order different metro-lines along each edge of the underlying network so that the total number of crossings is minimized. Although the problem has been studied, many questions remain open.

Fig. 1. A part of the official metro map of Paris.

⋆ Research supported in part by NSF grant DEB 1053573.

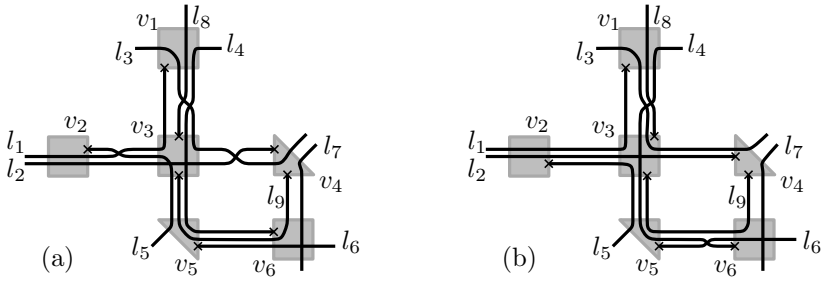


Fig. 2. Nine lines on a portion of an underlying network with 6 vertices and 8 edges. (a) $\pi_{v_3v_4} = (l_3, l_2)$ and $\pi_{v_3v_1} = (l_1, l_8, l_4, l_3)$. The lines l_3 and l_4 have an unavoidable edge crossing on $\{v_1, v_3\}$. In contrast, the crossing of l_2 and l_3 on $\{v_3, v_4\}$ is avoidable. In v_3 there is an unavoidable vertex crossing of the lines l_2 and l_8. As the vertex crossing of l_2 and l_5 in v_3 is avoidable the solution is not feasible. (b) A feasible solution satisfying the periphery condition.

Apart from the visualization of metro maps, the problem has various applications including the visual representation of biochemical pathways. In very-large-scale integration (VLSI) design, there is the closely related problem of minimizing intersections between nets (physical wires) [8,10]. Net patterns with fewer crossings have better electrical characteristics and require less area. In graph drawing, the number of edge crossings is one of the most important aesthetic criteria. In *edge bundling*, groups of edges are drawn close together—like metro lines—emphasizing the structure of the graph; minimizing crossings between parallel edges arises as a subproblem [14].

Problem Definitions. The input is a planarly embedded graph $G = (V, E)$ and a set L of simple paths in G. We call G the *underlying network*, the vertices *stations*, and the paths *lines*. The endpoints v_0, v_k of a line $(v_0, \ldots, v_k) \in L$ are *terminals*, and the vertices $v_1, \ldots, v_{k-1}$ are *intermediate stations*. For each edge $e = (u, v) \in E$, let $L_e = L_{uv}$ be the set of lines passing through e.

Following previous work [2,12], we use the *k-side* model; each station v is represented by a polygon with k sides, where k is the degree of v in G; see Fig. 2. For $k \leq 2$ a rectangle is used. Each side of the polygon is called a *port* of v and corresponds to an incident edge $(v, u) \in E$. A line $(v_0, \ldots, v_k)$ is represented by a polyline starting at a port of v_0 (on the boundary of the polygon), passing through two ports of v_i for $1 \leq i < k$, and ending at a port of v_k. For each port of $u \in V$ corresponding to $(u, v) \in E$, we define the *line order* $\pi_{uv} = (l_1 \ldots l_{|L_{uv}|})$ as an ordered sequence of the lines in L_{uv}, which specifies the clockwise order at which the lines L_{uv} are connected to the port of u with respect to the center of the polygon. Note that there are two different line orders π_{uv} and π_{vu} on any edge (u, v) of the network. A *solution*, or a *line layout*, specifies line orders π_{uv} and π_{vu} for each edge $(u, v) \in E$.

A line crossing is a crossing between polylines corresponding to a pair of lines. We distinguish two types of crossings; see Fig. 2(a). An *edge crossing* between lines l and l' occurs whenever $\pi_{uv} = (\ldots l \ldots l' \ldots)$ and $\pi_{vu} = (\ldots l \ldots l' \ldots)$ for some

edge $(u, v) \in E$. We now consider the concatenated cyclic sequence π_u of the orders $\pi_{uv_1}, \ldots, \pi_{uv_k}$, where $(u, v_1), \ldots, (u, v_k)$ are the edges incident to u in clockwise order. A *vertex crossing* between l and l' occurs in u if $\pi_u = (\ldots l \ldots l' \ldots l \ldots l' \ldots)$. Intuitively, the lines change their relative order inside u. A crossing is called *unavoidable* if the lines cross in each line layout; otherwise it is *avoidable*. A crossing is unavoidable if neither l nor l' have a terminal on their common subpath and the lines split on both ends of this subpath in such a way that their relative order has to change; see Fig. 2. By checking all pairs of lines, we can determine all unavoidable crossings in $O(|L|^2|E|)$ time. Following previous work, we insist that (i) *avoidable vertex crossings are not allowed* in a solution, that is, these crossings are not hidden below a station symbol, and (ii) *unavoidable vertex crossings are not counted* since they occur in any solution.

A pair of lines may share several common subpaths, and the lines may cross multiple times on the subpaths. For simplicity of presentation, we assume that there is at most one common subpath of two lines. Our results do, however, also hold for the general case as every common subpath can be considered individually.

Problem variants. Several variants of the problem have been considered in the literature. The original metro-line crossing minimization problem is formulated as follows.

*Problem 1 (*MLCM*).* For a given instance (G, L), find a line layout with the minimum number of crossings.

In practice, it is desirable to avoid gaps between adjacent lines; to this end, every line is drawn so that it starts and terminates at the topmost or bottommost end of a port; see Fig. 2(b). In fact, many manually created maps follow this *periphery condition* introduced by Bekos et al. [4]. Formally, we say that a line order π_{uv} at the port of u satisfies the periphery condition if $\pi_{uv} = (l_1 \ldots l_p \ldots l_q \ldots l_{|L_{uv}|})$, where u is a terminal for the lines $l_1, \ldots, l_p, l_q, \ldots, l_{|L_{uv}|}$ and u is an intermediate station for the lines $l_{p+1}, \ldots, l_{q-1}$. The problem is known as MLCM *with periphery condition*.

*Problem 2 (*MLCM-P*).* For a given instance (G, L), find a line layout, subject to the periphery condition on every port, with the minimum number of crossings.

In the special case of MLCM-P with *side assignment* (MLCM-PA), the input additionally specifies for each line end on which side of its port it terminates; Nöllenburg [12] showed that MLCM-PA is computationally equivalent to the version of MLCM in which all lines terminate at vertices of degree one.

As MLCM and MLCM-P are NP-hard even for very simple networks, we introduce the additional constraint that no line is a subpath of another line. Indeed, this is often the case for bus and metro transportation networks; if, however, there is a line that is a subpath of a longer line then one can also visualize it as a part of the longer line. We call the problems with this new restriction PROPER-MLCM and PROPER-MLCM-P.

Previous Work. Benkert et al. [5] described a quadratic-time algorithm for MLCM when the underlying network consists of a single edge with attached leaves, leaving open the complexity status of MLCM.

Table 1. Overview of results for the metro-line crossing minimization problem.

problem	graph class	result	reference
MLCM	caterpillar	NP-hard	Thm. 1
MLCM	single edge	$O(\|L\|^2)$-time algorithm	[5]
MLCM	general graph	crossing-free test	Thm. 2
MLCM-P	path	NP-hard	[2]
MLCM-P	general graph	ILP	[3]
MLCM-P	general graph	$O(\sqrt{\log \|L\|})$-approximation	Thm. 5
MLCM-P	general graph	crossing-free test	Thm. 3
PROPER-MLCM-P	general graph with consistent lines	$O(\|L\|^3)$-time algorithm	Thm. 7
MLCM-PA	general graph	$O(\|V\| + \|E\| + \|V\|\|L\|)$-time	[14]
MLCM-PA	general graph	crossing-free test	[12]

Bekos et al. [4] studied MLCM-P and proved that the variant is NP-hard on paths. Motivated by the hardness, they introduced the variant MLCM-PA and studied the problem on simple networks. Later, polynomial-time algorithms for MLCM-PA were found with gradually improving running time by Asquith et al. [3], Argyriou et al. [2], and Nöllenburg [12], until Pupyrev et al. [14] presented a linear-time algorithm. Asquith et al. [3] formulated MLCM-P as an integer linear program that finds an optimal solution for the problem on general graphs. Note that in the worst case this approach requires exponential time. Fink and Pupyrev studied a variant of MLCM in which a crossing between two blocks of lines is counted as a single crossing [7]. Okamoto et al. [13] presented exact and fixed-parameter tractable algorithms for MLCM-P on paths.

In the circuit design community (VLSI), Groeneveld [8] considered the problem of adjusting the routing so as to minimize crossings between the pairs of nets, which is equivalent to MLCM-PA, and suggested an algorithm for general graphs. Marek-Sadowska et al. [10] considered a related problem of distributing the line crossings among edges of the underlying graph in order to simplify the net routing.

Our Results. Table 1 summarizes our contributions and previous results. We first prove that the unconstrained variant MLCM is NP-hard even on caterpillars (paths with attached leaves), thus, answering an open question of Benkert et al. [5] and Nöllenburg [11]. As crossing minimization is hard, it is natural to ask whether there exists a *crossing-free* solution. We show that there is a crossing-free solution if and only if there is no pair of lines forming an unavoidable crossing.

We then study MLCM-P. Argyriou et al. [2] and Nöllenburg [11] asked for an approximation algorithm. To this end, we develop a 2SAT model for the problem. Using the model, we get an $O(\sqrt{\log |L|})$-approximation algorithm for MLCM-P. This is the first approximation algorithm in the context of metro-line crossing minimization. We also show how to find a crossing-free solution (if it exists) in polynomial time. Moreover, we prove that MLCM-P is fixed-parameter tractable with respect to the maximum number of allowed crossings by using the fixed-parameter tractability of 2SAT.

We then study the new variant PROPER-MLCM-P and show how to solve it on caterpillars, *left-to-right trees* (considered in [4,2]), and other instances described in Section 4. An optimal solution can be found by applying a maximum flow algorithm

on a certain graph. This is the first polynomial-time exact algorithm for the variant in which avoidable crossings may be present in an optimal solution.

2 The MLCM Problem

We begin with MLCM, the most general formulation of the problem, and show that it is hard to decide whether there is a solution with at most $k > 0$ crossings, even if the underlying network is a caterpillar. In contrast, we give a polynomial-time algorithm for deciding whether there exists a crossing-free solution.

Theorem 1. MLCM *is* NP*-hard on caterpillars.*

Proof. We prove hardness by reduction from MLCM-P which is known to be NP-hard on paths [2]. Suppose we have an instance of MLCM-P consisting of a path $G = (V, E)$ and lines L on the path. We want to decide whether it is possible to order the lines with periphery condition and at most k crossings.

We create a new underlying network $G' = (V', E')$ by adding vertices and edges to G. We assume that G is embedded along a horizontal line and specify positions relative to this line. For each edge $e = (u, v) \in E$, we add vertices u_1, u_2, v_1, and v_2 and edges (u, u_1), (u, u_2), (v, v_1), and (v, v_2) such that v_1 and u_1 are above the path and v_2 and u_2 are below. Next, we add $\ell = |L|^2$ lines from u_1 to v_2, and ℓ lines from u_2 to v_1 to $L' \supseteq L$; see Fig. 3. We call the added structure the *red cross* of e and the added lines *red lines*.We claim that there is a number K such that there is a solution of MLCM-P on (G, L) with at most k crossings if and only if there is solution of MLCM on (G', L') with at most $k + K$ crossings.

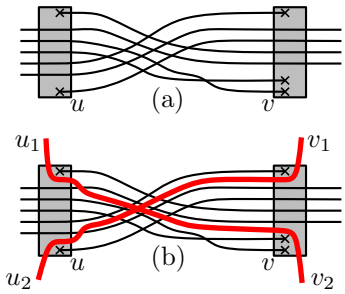
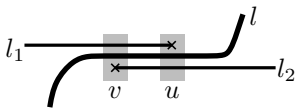


Fig. 3. (a) MLCM-P-solution on (u, v). (b) Insertion of a red cross.

Let $e = (u, v) \in E$ be an edge of the path, and let $l \in L_e$. If l has its terminals on u and v, that is, completely lies on e, it never has to cross in G or G'; hence, we assume such lines do not exist. Assume l has none of its terminals on u or v. It has to cross all 2ℓ lines of the red cross of e. Finally, suppose l has just one terminal at a vertex of e, say on u. If the line end of l at u is above the edge (u, u_1) in the order π_{uv}, then it has to cross all red lines from u_2 to v_1 but can avoid the red lines from u_1 to v_2, that is, ℓ crossings with red lines are necessary. Symmetrically, if the line end is below (u, u_2) then only the ℓ crossings with the red lines from u_1 to v_2 are necessary. If the terminal is between the edges (u, u_1) and (u, u_2) then all 2ℓ red edges must be crossed. There are, of course, always ℓ^2 unavoidable internal crossings of the red cross of e.

Let $\ell_e = \ell_e^t + \ell_e^m$ be the number of lines on e, where ℓ_e^t and ℓ_e^m are the numbers of lines on e that do or do not have a terminal at u or v, respectively. In any solution there are at least $\ell_e^t \cdot \ell + 2 \cdot \ell_e^m \cdot \ell + \ell^2$ crossings on e in which at least one red line is involved. It is easy to see that placing a terminal between red lines leaving towards a leaf never brings an advantage. On the other hand, if just a single line has an avoidable crossing with a



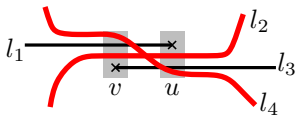


Fig. 4. A separator l of lines l_1 and l_2. **Fig. 5.** Unavoidable crossing of 2 separators of l_1 and l_3.

block of red lines, the number of crossings increases by $\ell = |L|^2$, which is more than the number of crossings in any solution for (G, L) without double crossings. Hence, any optimal solution of the lines in G' has no avoidable crossings with red blocks and, therefore, satisfies the periphery condition; thus, after deleting the added edges and red lines, we get a feasible solution for MLCM-P on G.

Let $K := |E| \cdot \ell^2 + \sum_{e \in E} (\ell_e^t + 2\ell_e^m) \cdot \ell$ be the minimum number of crossings with red lines involved on G'. Suppose we have an MLCM-solution on G' with at most $K+k$ crossings. Then, after deleting the red lines, we get a feasible solution for MLCM-P on G with at most k crossings. On the other hand, if we have an MLCM-P-solution on G with k crossings, then we can insert the red lines with just K new crossings: Suppose we want to insert the block of red lines from u_1 to v_2 on an edge $e = (u, v) \in E$. We start by putting them immediately below the lines with a terminal on the top of u. Then we cross all lines below until we see the first line that ends on the bottom of v and, hence, must not be crossed by this red block. We go to the right and just keep always directly above the block of lines that end at the bottom side of v; see Fig. 3. Finally, we reach v and have not created any avoidable crossing. Once we have inserted all blocks of red lines, we get a solution for the lines on G with exactly $K + k$ crossings. □

Crossing-Free Instances. Given an instance of MLCM, we want to check whether there exists a solution without any crossings. If there exists such a crossing-free solution then there cannot be a pair of lines with an unavoidable crossing. We show that this condition is already sufficient. We sketch the proofs; see full version for the complete proof [6].

We assume that no line is a subpath of another line as a subpath can be reinserted parallel to the longer line in a crossing-free solution. Consider a pair of lines l_1, l_2 whose common subpath P starts in u and ends in v. If u (similarly, v) is not a terminal for both l_1 and l_2 then there is a unique relative order of the lines along P in any crossing-free solution. Hence, we assume u is a terminal for l_1, v is a terminal for l_2, and we call such a pair *overlapping*. Suppose there is a *separator* for l_1 and l_2, that is, a line l on the subpath of l_1 and l_2 that has to be below l_1 and above l_2 (or the other way round) as shown in Fig. 4. Then, l_1 has to be above l_2 in a crossing-free solution. The only remaining case is a pair of lines l_1, l_2 without a separator. Suppose l_1, l_2 is chosen such that the number of edges of the common subpath is minimum. If there exists a crossing-free solution then there is also a crossing-free solution in which l_1 and l_2 are immediate neighbors in the orders on their common subpath; see full version [6].

Theorem 2. *Any instance of* MLCM *without unavoidable crossings has a crossing-free solution.*

Proof (sketch). We can merge a pair of overlapping lines without a separator into a new line. This merging step does not introduce an unavoidable crossing. We iteratively

perform merging steps until any overlapping pair has a separator. There might be multiple separators for a pair, but all of them separate the pair in the same relative order; otherwise, there would be a pair of separators with an unavoidable crossing; see Fig. 5. After the merging steps, we get a relative order for every pair of lines sharing an edge. One can show that the relative orders are acyclic. We build a crossing-free solution by putting all lines in the (only) right order on the edge. As the relative order of any pair of lines is the same on any edge, there cannot be a crossing. □

The proof yields an $O(|L|^2|E|)$-time algorithm for finding crossing-free solutions.

3 The MLCM-P Problem

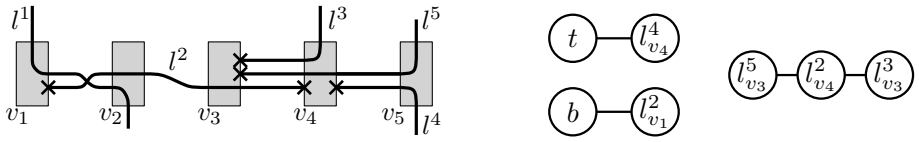
Let $(G = (V, E), L)$ be an instance of MLCM-P. Our goal is to decide for each line end on which side of its terminal port it should lie. We arbitrarily choose one side of each port and call it "top", the opposite side is called "bottom". For each line l starting at vertex u and ending at vertex v, we create binary variables l_u and l_v, which are true if and only if l terminates at the top side of the respective port. We formulate the problem of finding a truth assignment that minimizes the number of crossings as a 2SAT instance. Note that Asquith et al. [3] already used 2SAT clauses as a tool for developing their ILP for MLCM, where the variables represent above/below relations between line ends. In contrast, in our model a variable directly represents the position of a line on the top or bottom side of a port. We first prove a simple property of lines.

Lemma 1. *Let l, l' be a pair of lines sharing a terminal. We can transform any solution in which l and l' cross to a solution with fewer crossings in which the lines do not cross.*

Proof. Assume l and l' cross in a solution. We switch the positions of line ends at the common terminal v between l and l' and reroute the two lines between the crossing's position and v. By reusing the route of l for l' and vice versa, the number of crossings does not increase. On the other hand, the crossing between l and l' is eliminated. □

Let l, l' be two lines whose common subpath P starts at vertex u and ends at v. Terminals of l and l' that lie on P can only be at u or v. If neither l nor l' has a terminal on P, then a crossing of the lines does not depend on the positions of the terminals; hence, we assume that there is a terminal at u or v. A possible crossing between l and l' is modeled by a 2SAT formula, the *crossing formula*, consisting of at most two clauses. This formula evaluates to true if and only if l and l' do not cross. For simplicity, we assume that the top sides of the terminal ports of u and v are located on the same side of P; see Fig. 6. If it is not the case, a variable l_u should be substituted with its inverse $\neg l_u$ in the formula. Note that generating all crossing formulas needs $O(|E||L|^2)$ time. We consider several cases; see also the illustrations in the full version [6].

(f1) Suppose u and v are terminals for l and intermediate stations for l', that is, l is a subpath of l'. Then, l does not cross l' if and only if both terminals of l lie on the same side of P. This is expressed by the crossing formula $(l_u \wedge l_v) \vee (\neg l_u \wedge \neg l_v) \equiv (\neg l_u \vee l_v) \wedge (l_u \vee \neg l_v)$, which may occur multiple times, caused by a different l'.



(a) An instance (G, L) of PROPER-MLCM-P. (b) Graph G_{ab} for the instance (G, L).

Fig. 6. A small instance of MLCM-P. The generated 2SAT formulas are: $(l^2_{v_1})$ for the crossing of l^1 and l^2; $(\neg l^4_{v_4})$ for the crossing of l^5 and l^4; $(l^2_{v_4} \vee l^3_{v_3}) \wedge (\neg l^2_{v_4} \vee \neg l^3_{v_3})$ for the crossing of l^2 and l^3; $(l^2_{v_4} \vee l^5_{v_3}) \wedge (\neg l^2_{v_4} \vee \neg l^5_{v_3})$ for the crossing of l^2 and l^5.

(f2) Suppose u is a terminal for l and intermediate for l', and v is a terminal for l' and intermediate for l. Then there is no crossing if and only if both terminals lie on opposite sides of P. This is described by the formula $(l_u \wedge \neg l'_v) \vee (\neg l_u \wedge l'_v) \equiv (l_u \vee l'_v) \wedge (\neg l_u \vee \neg l'_v)$.

(f3) Suppose both l and l' terminate at the same vertex u or v. By Lemma 1, a solution of MLCM-P with a crossing of l and l' can be transformed into a solution in which l and l' do not cross. Hence, we do not introduce formulas in this case.

(f4) In the remaining case, there is only one terminal of l and l' on P. Without loss of generality, let l terminate at u. A crossing is triggered by a single variable. Depending on the fixed terminals or leaving edges at v and u, we get the single clause (l_u) or $(\neg l_u)$. The same clause can occur multiple times, caused by different lines l'.

Crossing-free solutions. Note that the 2SAT formulation of the problem yields an algorithm for deciding whether there exists a crossing-free solution of an MLCM-P instance. First, we check for unavoidable crossings by analyzing every pair of lines individually. Second, the 2SAT model is satisfiable if and only if there is a solution of the MLCM-P instance without avoidable crossing. Since 2SAT can be solved in linear time and there are at most $|L|^2$ crossing formulas, we conclude as follows.

Theorem 3. *Deciding whether there exists a crossing-free solution for* MLCM-P *can be accomplished in* $O(|E||L|^2)$ *time.*

For MLCM the existence of a crossing-free solution is equivalent to the absence of unavoidable crossings. In contrast, there is no such simple criterion for MLCM-P.

Fixed-parameter tractability. We can use the 2SAT model for obtaining a fixed-parameter tractable algorithm on the number k of allowed crossings. We must show that we can check in $f(k) \cdot \mathrm{poly}(\mathcal{I})$ time whether there is a solution with at most k avoidable crossings, where f must be a computable function and $\mathcal{I}$ is the input size.

First, note that minimizing the number of crossings is the same as maximizing the number of satisfied clauses in the corresponding 2SAT instance. Maximizing the number of satisfied clauses, or solving the MAX-2SAT problem, is NP-hard.

However, the problem of deciding whether it is possible to remove a given number k of m 2SAT clauses so that the formula becomes satisfiable is fixed-parameter tractable with respect to the parameter k [15]. This yields the following theorem.

Theorem 4. MLCM-P *is fixed-parameter tractable with respect to the maximum allowed number of avoidable crossings.*

Proof. We show that the SAT formula can be made satisfiable by removing at most k clauses if and only if there is a solution with at most k crossings.

First, suppose it is possible to remove at most k clauses from the 2SAT model so that there is a truth assignment satisfying all remaining clauses. Fix such a truth assignment, and consider the corresponding assignment of sides to the terminals. Any crossing leads to an unsatisfied clause in the SAT formula, and no two crossings share an unsatisfied clause. Hence, we have a side assignment that causes at most k crossings.

Now, we assume that there is an assignment of sides for all terminals that causes at most k crossings. We know that in the corresponding truth assignment for all pairs of clauses of types (f1)–(f2) of the SAT model at most one is unsatisfied. Hence, there are at most k unsatisfied clauses since any crossing just leads to a single unsatisfied clause. The removal of these clauses creates a new, satisfiable formula. □

Using the $O(15^k km^3)$-time algorithm for 2SAT [15] our algorithm has a running time of $O(15^k \cdot k \cdot |L|^6 + |L|^2|E|)$.

Approximating MLCM-P. The proof of Theorem 4 yields that the number of crossings in a crossing-minimal solution of MLCM-P equals the minimum number of clauses that we need to remove from the 2SAT formula in order to make it satisfiable. Furthermore, a set of k clauses, whose removal makes the 2SAT formula satisfiable, corresponds to an MLCM-P solution with at most k crossings. Hence, an approximation algorithm for the problem of making a 2SAT formula satisfiable by removing the minimum number of clauses (also called MIN 2CNF DELETION) yields an approximation for MLCM-P of the same quality. As there is an $O(\sqrt{\log m})$-approximation algorithm for MIN 2CNF DELETION [1], we have the following result.

Theorem 5. *There is an* $O(\sqrt{\log |L|})$*-approximation algorithm for* MLCM-P.

4 The PROPER-MLCM-P Problem

In this section we consider the PROPER-MLCM-P problem, where no line in L is a subpath of another line. First we focus on graphs whose underlying network is a caterpillar. There, the top and bottom sides of ports are given naturally; see Fig. 6.

Based on the 2SAT model described in the previous section, we construct a graph G_{ab}, which has a vertex l_u for each variable of the model and two additional vertices b and t. Since no line is a subpath of another line, our 2SAT model has only the two types of crossing formulas (f2) and (f4); compare Section 3. For case (f2), we create an edge (l_u, l'_v). The edge models a possible crossing between lines l and l'; that is, the lines cross if and only if l terminates on top (bottom) of u and l' terminates on top (bottom) of v. For a crossing formula of type (l_u) (case (f4)), we add an edge (b, l_u) to G_{ab}; similarly, we add an edge (t, l_u) for a formula $(\neg l_u)$; see Fig. 6(b) for an example.

Any truth assignment to the variables is equivalent to a b-t cut in G_{ab}, that is, a cut separating b and t. Indeed, any edge in the graph models the fact that two lines should

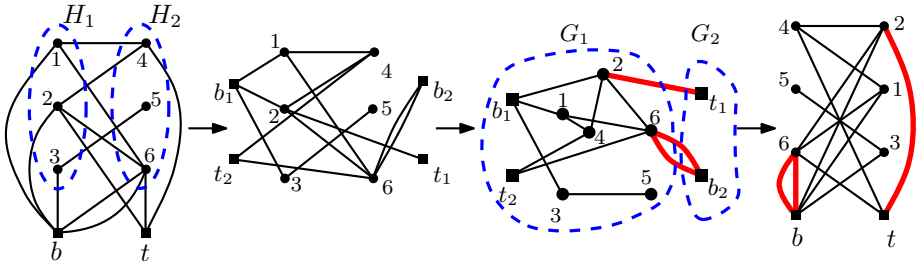


Fig. 7. Solving MIN-UNCUT on an almost bipartite graph. The maximum flow (minimum cut) with value 3 results in vertex partitions $V_b^1 = \{b_1, 4, 5, 6\}$, $V_t^1 = \{t_2, 1, 2, 3\}$, $V_b^2 = \{b_2\}$, and $V_t^2 = \{t_1\}$. The optimal partition $S_b = \{b, 4, 5, 6\}, S_t = \{t, 1, 2, 3\}$ induces 3 uncut edges $(b, 6), (b, 6), (t, 2)$.

not be assigned to the same side as they would cause a crossing otherwise. Hence, any line crossing corresponds to an *uncut* edge. Therefore, for minimizing the number of crossings, we need to solve the known MIN-UNCUT problem, which asks for a partitioning of the vertices of a graph into sets S_t, S_b so that the number of uncut edges ((v, u) with $v, u \in S_t$ or $v, u \in S_b$) is minimized. Although MIN-UNCUT is NP-hard, it turns out that the graph G_{ab} has a special structure, which we call *almost bipartite*.

Definition 1. *A graph $G = (V, E)$ is called* almost bipartite *if it is a union of a bipartite graph $H = (V_H, E_H)$ and two additional vertices b, t whose edges may be incident to vertices of both partitions of H, that is, $V = V_H \cup \{b\} \cup \{t\}$ and $E = E_H \cup E'$, where $E' \subseteq \{(b, v) \mid v \in V\} \cup \{(t, v) \mid v \in V\}$.*

The bipartition is given by the fact that "left" (similarly, "right") terminals of two lines can never be connected by an edge in G_{ab}. We show that MIN-UNCUT can be solved optimally for almost bipartite graphs.

Theorem 6. MIN-UNCUT *can be solved in $O(|V|^3)$ on almost bipartite graphs.*

Proof. Almost bipartite graphs are a subclass of *weakly bipartite graphs*. It is known that MIN-UNCUT can be solved in polynomial time on weakly bipartite graphs using the ellipsoid method [9]. We present a simple and efficient combinatorial algorithm.

The special vertices b and t have to belong to different partitions of G_{ab}. We create a new graph G' from G_{ab}. We split vertex b into b_1, b_2 and t into t_1, t_2 such that b_1 and t_1 are connected to the vertices of the first partition H_1 of H, and b_2 and t_2 are connected to the second partition H_2. Formally, for each edge $(b, v) \in E, v \in H_1$, we create an edge (b_1, v); for each edge $(b, v) \in E, v \in H_2$, we create an edge (v, b_2). Similarly, edges (v, t_1) are created for all $(t, v) \in E, v \in H_1$, and edges (t_2, v) are created for all $(t, v) \in E, v \in H_2$. The construction is illustrated in Fig. 7.

Now, for each edge (u, v) of G' we assign capacity 1, and compute a maximum flow between the pair of sources b_1, t_2 to the pair of sinks b_2, t_1. This can be done in $O(|V|^3)$ time using a maximum flow algorithm with a supersource (connected to b_1 and t_2) and a supersink (connected to b_2 and t_1). Indeed, there is an integral maximum flow in G'.

A maximum flow corresponds to a maximum set of edge-disjoint paths starting at b_1 or t_2 and ending at b_2 or t_1. Such a path corresponds to one of the following structures in the original graph G: (i) an odd cycle containing vertex b (a cycle with an odd number of edges); (ii) an odd cycle containing vertex t; (iii) an even path between b and t.

Note that if a graph has an odd cycle then at least one of the edges of the cycle belongs to the same partition in any solution of MIN-UNCUT. The same holds for an even path connecting b and t in G since b and t have to belong to different partitions. Since the maximum flow corresponds to the edge-disjoint odd cycles and even paths in G, the value of the flow is a lower bound for a solution of MIN-UNCUT.

Let us prove that the value of the maximum flow in G' is also an upper bound. By Menger's theorem, this value is the cardinality of a minimum edge cut separating sources and sinks. Let E^* be the minimum edge cut and let G_1 and G_2 be the correspondent disconnected subgraphs of G'; see Fig. 7. Note that G_1 is bipartite since $H \cap G_1$ is bipartite; vertex b_1 is only connected to vertices of H_1 and vertex t_2 is only connected to vertices of H_2. Therefore, there is a 2-partition of vertices of G_1 such that b_1 and t_2 belong to different partitions; let us denote the partitions V_b^1 and V_t^1. Similarly, there is a 2-partition of G_2 into V_b^2 and V_t^2 with $b_2 \in V_b^2$ and $t_1 \in V_t^2$. We combine these partitions so that $S_b = \{b\} \cup \big(V_b^1 \cup V_b^2\big) \setminus \{b_1, b_2\}$ and $S_t = \{t\} \cup \big(V_t^1 \cup V_t^2\big) \setminus \{t_1, t_2\}$, which yield the required partition of vertices of G for MIN-UNCUT. The set of uncut edges is E^*, which completes the proof of the theorem. □

As a direct corollary, we get a $O(|L|^3)$-time algorithm for PROPER-MLCM-P on caterpillars. I can be applied for some other underlying networks. Let $(G = (V, E), L)$ be an instance of PROPER-MLCM-P. The lines L have *consistent* directions on G if the lines can be directed so that for each edge $e \in E$ all lines L_e have the same direction. If the underlying graph is a path then we can consistently direct the lines from left to right. Similarly, consistent line directions exist for "left-to-right" [4,2] and "upward" [7] trees, that is, trees for which there is an embedding with all lines being monotone in some direction. It is easy to test whether there are consistent line directions by giving an arbitrary direction to some first line, and then applying the same direction on all lines sharing edges with the first line until all lines have directions or an inconsistency is found. Hence, we get the following result; see full version for the proof [6].

Theorem 7. PROPER-MLCM-P *can be solved in* $O(|L|^3)$ *time for instances* (G, L) *admitting consistent line directions.*

5 Conclusion and Open Problems

We proved that MLCM is NP-hard and presented an $O(\sqrt{\log |L|})$-approximation algorithm for MLCM-P, as well as an exact $O(|L|^3)$-time algorithm for PROPER-MLCM-P on instances with consistent line directions. We also suggested polynomial-time algorithms for crossing-free solutions for MLCM and MLCM-P. From a theoretical point of view, there are still interesting open problems: 1. Is there an approximation algorithm for MLCM? 2. Is there a constant-factor approximation algorithm for MLCM-P? 3. What is the complexity status of PROPER-MLCM/PROPER-MLCM-P in general?

On the practical side, the visualization of the computed line crossings is a possible future direction. So far, the focus has been on the number of crossings, although two line orders with the same crossing number may look quite differently [7]. For example, a metro line is easy to follow if it has few bends. Hence, an interesting question is how to visualize metro lines using the minimum total number of bends.

Acknowledgments: We thank Martin Nöllenburg, Jan-Henrik Haunert, Joachim Spoerhase, Lukas Barth, Stephen Kobourov, and Sankar Veeramoni for discussions about problem variants. We are especially grateful to Alexander Wolff for help with the paper.

References

1. Agarwal, A., Charikar, M., Makarychev, K., Makarychev, Y.: $O(\sqrt{\log n})$ approximation algorithms for min UnCut, min 2CNF deletion, and directed cut problems. In: STOC 2005. pp. 573–581. ACM, New York (2005)
2. Argyriou, E.N., Bekos, M.A., Kaufmann, M., Symvonis, A.: On metro-line crossing minimization. Journal of Graph Algorithms and Applications 14(1), 75–96 (2010)
3. Asquith, M., Gudmundsson, J., Merrick, D.: An ILP for the metro-line crossing problem. In: Harland, J., Manyem, P. (eds.) CATS 2008. CRPIT, vol. 77, pp. 49–56. Australian Computer Society (2008)
4. Bekos, M.A., Kaufmann, M., Potika, K., Symvonis, A.: Line crossing minimization on metro maps. In: Hong, S.H., Nishizeki, T., Quan, W. (eds.) GD 2007. LNCS, vol. 4875, pp. 231–242. Springer, Heidelberg (2008)
5. Benkert, M., Nöllenburg, M., Uno, T., Wolff, A.: Minimizing intra-edge crossings in wiring diagrams and public transport maps. In: Kaufmann, M., Wagner, D. (eds.) GD 2006. LNCS, vol. 4372, pp. 270–281. Springer, Heidelberg (2007)
6. Fink, M., Pupyrev, S.: Metro-line crossing minimization: Hardness, approximations, and tractable cases. ArXiv e-print abs/1306.2079 (2013)
7. Fink, M., Pupyrev, S.: Ordering metro lines by block crossings. In: Chatterjee, K., Sgall, J. (eds.) MFCS 2013. LNCS, vol. 8087, pp. 397–408. Springer, Heidelberg (2013)
8. Groeneveld, P.: Wire ordering for detailed routing. IEEE Des. Test 6(6), 6–17 (1989)
9. Grötschel, M., Pulleyblank, W.: Weakly bipartite graphs and the Max-Cut problem. Operations Research Letters 1(1), 23–27 (1981)
10. Marek-Sadowska, M., Sarrafzadeh, M.: The crossing distribution problem. IEEE Transactions on CAD of Integrated Circuits and Systems 14(4), 423–433 (1995)
11. Nöllenburg, M.: Network Visualization: Algorithms, Applications, and Complexity. Ph.D. thesis, Fakultät für Informatik, Universität Karlsruhe (TH) (2009)
12. Nöllenburg, M.: An improved algorithm for the metro-line crossing minimization problem. In: Eppstein, D., Gansner, E.R. (eds.) GD 2009. LNCS, vol. 5849, pp. 381–392. Springer, Heidelberg (2010)
13. Okamoto, Y., Tatsu, Y., Uno, Y.: Exact and fixed-parameter algorithms for metro-line crossing minimization problems. ArXiv e-print abs/1306.3538 (2013)
14. Pupyrev, S., Nachmanson, L., Bereg, S., Holroyd, A.E.: Edge routing with ordered bundles. In: van Kreveld, M.J., Speckmann, B. (eds.) GD 2011. LNCS, vol. 7034, pp. 136–147. Springer, Heidelberg (2012)
15. Razgon, I., O'Sullivan, B.: Almost 2-SAT is fixed-parameter tractable. Journal of Computer and System Sciences 75(8), 435–450 (2009)