Minimizing Intra-Edge Crossings in Wiring Diagrams and Public Transportation Maps

Marc Benkert[1⋆], Martin Nöllenburg[1⋆], Takeaki Uno[2], and Alexander Wolff[1⋆]

[1] Department of Computer Science, Karlsruhe University, Germany.
WWW: http://i11www.iti.uka.de/algo/group
[2] National Institute of Informatics, Tokyo, Japan.
Email: uno@nii.jp

Abstract. In this paper we consider a new problem that occurs when drawing wiring diagrams or public transportation networks. Given an embedded graph $G = (V, E)$ (e.g., the streets served by a bus network) and a set L of paths in G (e.g., the bus lines), we want to draw the paths along the edges of G such that they cross each other as few times as possible. For esthetic reasons we insist that the relative order of the paths that traverse a node does not change within the area occupied by that node.
Our main contribution is an algorithm that minimizes the number of crossings on a single edge $\{u, v\} \in E$ if we are given the order of the incoming and outgoing paths. The difficulty is deciding the order of the paths that terminate in u or v with respect to the fixed order of the paths that do not end there. Our algorithm uses dynamic programming and takes $O(n^2)$ time, where n is the number of terminating paths.

1 Introduction

In wiring diagrams or public transportation networks many paths must be drawn on the same underlying graph, see Figures 1 and 2. In order to make the resulting layout as understandable as possible it is desirable to (a) avoid crossings wherever possible and (b) insist that the relative order of the lines that traverse a node does not change in that node. For example, note that the subway line 5, which passes under the main station of Cologne (Köln Hbf), crosses lines 16–19 south of the station in the clipping of the public transport map of Cologne in Figure 2. The crossing is not hidden under the rectangle that represents the station. This makes it easier to follow the subway lines visually.

We model the problem as follows. We assume that we are given an undirected connected graph $G = (V, E)$ together with an embedding in the plane. The graph represents the underlying structure of the wiring or the road/tracks in the case of a transportation network. We are also given a set L of *lines* in G. They represent the cables in a wiring diagram or the lines in a transportation network. A line $\ell \in L$ is an edge sequence $e_1 = \{v_0, v_1\}, e_2 = \{v_1, v_2\}, \ldots, e_k = \{v_{k-1}, v_k\} \in E$

⋆ Supported by grant WO 758/4-2 of the German Research Foundation (DFG).

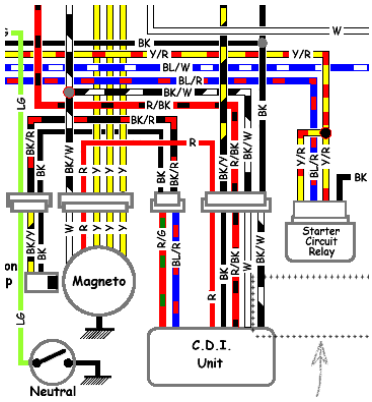


Fig. 1: Clipping of a wiring diagram.

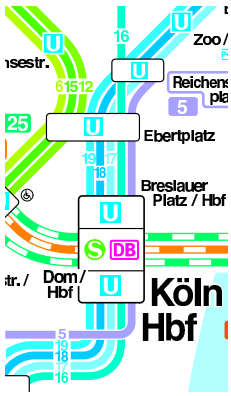


Fig. 2: Clipping of the public transport network of Cologne.

that forms a simple path in G. The stations v_0 and v_k are the *terminal stations* for line ℓ while $v_1, \ldots, v_{k-1}$ are *intermediate stations*. Our aim is to draw all lines in L such that the number of crossings among pairs of lines in L is minimized.

To get a flavor of the problem observe that the structure of G enforces certain crossings, see Figure 3a: lines ℓ_1 and ℓ_2 use exactly the path $\langle u, w_1, w_2, v\rangle$ together. The graph structure (indicated by the first and last line segments of each line) enforces that ℓ_1 enters station u above ℓ_2 while it leaves v below ℓ_2, thus ℓ_1 and ℓ_2 have to cross somewhere between u and v. However, fixing the location of the crossing of ℓ_1 and ℓ_2 determines crossings with other lines that have a terminal stop in w_1 or w_2. If there is a line ℓ that enters u between ℓ_1 and ℓ_2 and terminates at w_2 (see Figure 3b), the crossing between ℓ_1 and ℓ_2 should be placed between w_2 and v. Now suppose there is another line ℓ' that enters v between ℓ_2 and ℓ_1 from the right and terminates at w_2, see Figure 3c. Then at least one of the lines ℓ and ℓ' intersects one of the lines ℓ_1 or ℓ_2, no matter where the crossing between ℓ_1 and ℓ_2 is placed.

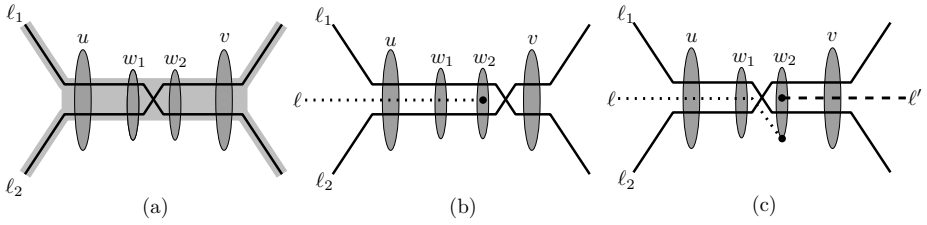


Fig. 3: Different placements of the necessary intersection between lines ℓ_1 and ℓ_2 on the path u, w_1, w_2, v. In (c) at least one of the lines ℓ and ℓ' has to intersect one of the lines ℓ_1 or ℓ_2.

We can abstract from geometry as follows. Given an edge $\{u, v\}$ in G and the orders of the lines in u and v that traverse u and v, respectively, we ask for the order of *all* lines that enter $\{u, v\}$ in u and for the order in which *all* lines leave $\{u, v\}$ in v. Then the number of crossings on $\{u, v\}$ is the number of transpositions needed to convert one order into the other. The difficulty is deciding the order of the lines that terminate in u or v with respect to the fixed order of the lines that traverse both u and v. This is the *one-edge layout* problem that we study in Section 2.

In contrast to the well-known NP-hard problem of minimizing crossings in a two-layer bipartite graph [3] the one-edge layout problem is polynomially solvable. The main reason is that there is an optimal layout of the lines that pass through edge $\{u, v\}$ such that no two lines that terminate in u intersect and no two lines that terminate in v intersect. This observation allows us to split the problem and to apply dynamic programming. It is then rather easy to come up with an $O(n^5)$-time solution and with some effort we could reduce the running time to $O(n^2)$, where n is the number of lines that do not terminate in u or v.

A solution of the general line layout problem, i.e., a simultaneous crossing-minimal solution for all edges of a graph, would be of interest as a second (and mostly independent) step for drawing bus or metro maps, a topic that has received some attention lately, see the work of Nöllenburg and Wolff [4] and the references therein. However, in that direction of research the focus has so far been exclusively on drawing the underlying graph nicely, and not on how to embed the bus or metro lines along the network. We give some hints in Section 3 why already the *two-edge layout* problem seems to be substantially harder than the one-edge layout.

A vaguely similar problem has been considered by Cortese et al. [1]. Given the drawing of a planar graph G they widen the edges and vertices of the drawing and ask if a given combinatorial cycle in G has a plane embedding in the widened drawing.

2 A Dynamic Program for One-Edge Layout

In this section we consider the following special case of the problem.

Problem 1. One-edge layout

We are given a graph $G = (V, E)$ and an edge $e = \{u, v\} \in E$. Let L_e be the set of lines that use e. We split L_e into three subgroups: L_{uv} is the set of lines that pass through u and v, i.e., neither u or v is a terminal station. L_u is the set of lines that pass through u and for which v is a terminal station and L_v is the set of lines that pass through v and for which u is a terminal station. We assume that there are no lines that exclusively use the edge $\{u, v\}$ as they could be placed top- or bottommost without causing any intersections. Furthermore we assume that the lines for which u is an intermediate station, i.e., $L_{uv} \cup L_u$, enter u in a predefined order S_u. Analogously, we assume that the lines for which v is an intermediate station, i.e., $L_{uv} \cup L_v$, enter v in a predefined order S_v. The task

is to find a layout of the lines in L_e such that the number of pairs of intersecting lines is minimized.

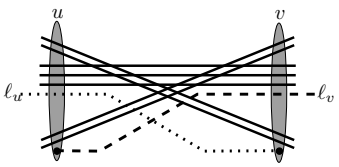


Fig. 4: The lines in L_{uv} are drawn solid. In an optimal solution $\ell_u \in L_u$ and $\ell_v \in L_v$ intersect.

Note that the number of crossings is determined by inserting the lines in L_u into the order S_v and by inserting the lines in L_v into S_u. The task is to find insertion orders that minimize the number of crossings. Observe that the orders S_u and S_v themselves already determine the number of crossings between pairs of lines in L_{uv} and that the insertion orders of L_u in S_v and of L_v in S_u do not change this number. Thus, we will not take crossings between lines in L_{uv} into account anymore. On the other hand, fixing an insertion order affects the number of crossings between lines in $L_u \cup L_v$ and L_{uv} and the number of crossings between lines in L_u and L_v in a non-trivial way. Figure 4 shows that a line $\ell_u \in L_u$ can indeed cross a line $\ell_v \in L_v$ in the unique optimal solution. Throughout the paper lines in L_{uv} are drawn solid, lines in L_u dotted and lines in L_v dashed. We will now show that no two lines in L_u (and analogously in L_v) cross in an optimal solution. This nice property is the key for solving the one-edge layout in polynomial time.

Lemma 1. *In any optimal solution for the one-edge layout problem no pair of lines in L_u and no pair of lines in L_v intersects.*

Proof. Assume to the contrary that there is an optimal solution σ with a pair of lines in L_u that intersects. Among all such pairs in σ let $\{\ell, \ell'\}$ be the one whose intersection point p is rightmost. W.l.o.g. ℓ is above ℓ' in u. Let ℓ_p and ℓ'_p be the parts of ℓ and ℓ' to the right of p, see Figure 5a. Since σ is crossing minimal, the courses of ℓ_p and ℓ'_p intersect the minimum number of lines in $L_{uv} \cup L_v$ in order to get from p to v. In particular, the number of crossings between ℓ_p and lines of $L_{uv} \cup L_v$ and between ℓ'_p and lines of $L_{uv} \cup L_v$ must be the same otherwise we could place ℓ_p parallel to ℓ'_p (or vice versa) which would reduce the number of crossings. However, since the number of crossings to the right is the same we can easily get rid of the crossing between ℓ and ℓ' by replacing ℓ_p by a copy of ℓ'_p infinitesimally close above ℓ'_p, see Figure 5b. The proof for L_v is analogous. □

From now on we assume that no two lines of L_u are consecutive in S_u and analogously no two lines of L_v are consecutive in S_v. The reason for this assumption is that a set of consecutive lines can simply be drawn parallelly in an optimal layout. Thus a single line suffices to determine the optimal course for the whole bundle. Technically, we can deal with this case by merging a bundle of k consecutive lines of L_u or L_v to one line and assigning a weight of k to it. The dynamic program will then run in a weighted fashion that counts $k \cdot k'$ crossings for a crossing of two lines with weights k and k'. For simplification we only explain the unweighted version of the problem in detail.

Let n, n_u and n_v be the number of lines in L_{uv}, L_u and L_v, respectively. Note that by the above assumption $n_u, n_v \leq n + 1$ holds.

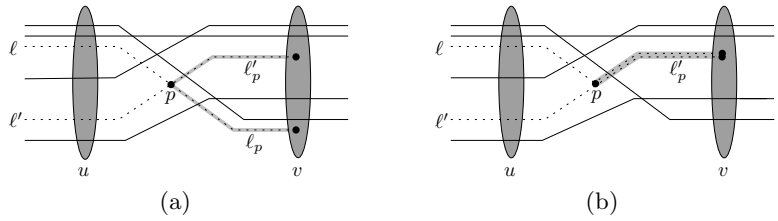


Fig. 5: Two lines of L_u do not intersect in an optimal solution.

Recall that by assumption the lines in $L_{uv} \cup L_u$ enter u in a predefined order and the lines in $L_{uv} \cup L_v$ leave v in a predefined order. Let $S_u = (s_1^u < \cdots < s_{n+n_u}^u)$ be the bottom-up order of lines in $L_{uv} \cup L_u$ in u and $S_v = (s_1^v < \cdots < s_{n+n_v}^v)$ be the bottom-up order of lines in $L_{uv} \cup L_v$ in v. A line ℓ in L_v can terminate below s_1^u, between two neighboring lines s_i^u, s_{i+1}^u, or above $s_{n+n_u}^u$. We denote the position of ℓ by the index of the lower line and by 0 if it is below s_1^u. Let $S_v|L_v = (s_{\pi(1)}^v < \ldots < s_{\pi(n_v)}^v)$ denote the order on L_v induced by S_v and let $S_u|L_u = (s_{\mu(1)}^u < \ldots < s_{\mu(n_u)}^u)$ denote the order on L_u induced by S_u. Here, π and μ are injective functions that filter the lines L_v out of all ordered lines $L_{uv} \cup L_v$ in S_v and the lines L_u out of all ordered lines $L_{uv} \cup L_u$ in S_u, see Figure 6.

Preprocessing. The orders S_u and S_v already determine the number of necessary crossings between pairs of lines in L_{uv}. Let cr_{uv} denote this number. Since cr_{uv} is fixed there is no need to consider the corresponding crossings in the minimization. We will now fix the course of a line in L_v. This line, say $\ell = s_{\pi(j)}^v$, has index $\pi(j)$ in S_v, and we fix the course of ℓ by choosing its terminal position i in the order S_u. We denote the number of crossings between ℓ and all lines in L_{uv} by $\mathrm{cr}_v(i, j)$. This number is determined as follows. The line ℓ crosses a line $\ell' \in L_{uv}$ with left index i' and right index j' if and only if either it holds that $i' \leq i$ and $j' > \pi(j)$ or it holds that $i' > i$ and $j' < \pi(j)$. The table cr_v for all lines in L_v has $(n + n_u + 1) \times n_v = O(n^2)$ entries. For fixed i we can compute the row $\mathrm{cr}_v(i, \cdot)$ as follows. We start with $j = 1$ and compute the number of lines in L_{uv} that intersect line ℓ with indices i and $\pi(j)$. Then, we increment j and obtain $\mathrm{cr}_v(i, j+1)$ by $\mathrm{cr}_v(i, j)$ minus the number of lines in L_{uv} that are no longer intersected plus the lines that are newly intersected. As any of the n lines in L_{uv} receives this status 'no longer' or 'newly' at most once and this status can easily be checked by looking at S_v, this takes $O(n)$ time per row. Thus, in total we can compute the matrix cr_v in $O(n^2)$ time.

We define $\mathrm{cr}_u(i, j)$ analogously to be the number of crossings of the lines in L_{uv} with a line in L_u that has index $\mu(i)$ in S_u and position j in S_v. Computing cr_u is analogous to cr_v.

Dynamic Program. Assume that we fix the destination of $s_{\pi(j)}^v$ to some $i \in \{0, \ldots, n + n_u\}$. Then we define $F(i, j)$ as the minimum number of crossings of

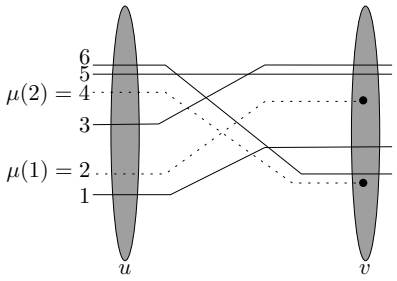


Fig. 6: The order S_u and the induced suborder $S_u|L_u = (s_2^u < s_4^u)$.

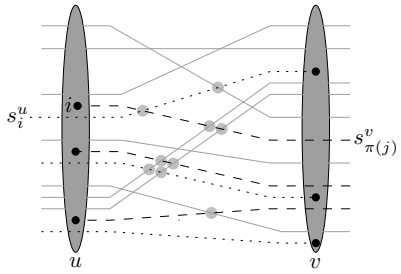


Fig. 7: Configuration corresponding to $F(i, j)$: line $s^v_{\pi(j)}$ terminates at position i in u.

(a) the lines in $\{s_1^u, \ldots, s_i^u\} \cap L_u$ with the lines in $L_{uv} \cup L_v$ and (b) the lines in $\{s_1^v, \ldots, s^v_{\pi(j)}\} \cap L_v$ with the lines in $L_{uv} \cup L_u$. This situation is depicted in Figure 7, where only the crossings indicated by gray disks are counted in $F(i, j)$. Then the values $F(i, j)$ define an $(n + n_u + 1) \times n_v$-matrix F.

Once the last column $F(\cdot, n_v)$ of the matrix F is computed, i.e., all lines in L_v are placed, we can determine the optimal solution for L_e as

$$F^* := \min\{F(i, n_v) + C(i, n + n_u, n_v + 1) \mid i = 0, \ldots, n\},$$

where $C(i, n + n_u, n_v + 1)$ is the remaining number of crossings of lines in $L_u \cap \{s_{i+1}^u, \ldots, s_{n+n_u}^u\}$ with $L_{uv} \cup L_v$, which are not yet counted in $F(i, n_v)$.

Before turning to the recursive computation of $F(i, j)$ we introduce another notation. Let us assume that $s^v_{\pi(j-1)}$ terminates at position k and $s^v_{\pi(j)}$ terminates at position i, where $0 \le k \le i \le n + n_u$ and $j \in \{1, \ldots, n_v\}$. Then let $C(k, i, j)$ denote the minimum number of crossings that the lines $L^u_{k,i} := \{s^u_{k+1}, \ldots, s^u_i\} \cap L_u$ cause with $L_{uv} \cup L_v$. In other words $C(k, i, j)$ counts the minimal number of crossings of all lines of L_u in the interval defined by the endpoints of the two lines $s^v_{\pi(j-1)}$ and $s^v_{\pi(j)}$. This situation is illustrated in Figure 8, where those crossings that are marked with gray disks are counted in the term $C(k, i, j)$. The following theorem gives the recursion for F and shows its correctness.

Theorem 1. *The values $F(i, j)$, $i = 0, \ldots, n + n_u$, $j = 1, \ldots, n_v$, can be computed recursively by*

$$F(i, j) = \begin{cases} \min_{k \le i}\{F(k, j-1) + C(k, i, j) + cr_v(i, j)\} & \text{if } i \ge 1, j \ge 2 \\ \sum_{l=1}^{j} cr_v(0, l) & \text{if } i = 0, j \ge 1 \\ C(0, i, 1) + cr_v(i, 1) & \text{if } i \ge 1, j = 1. \end{cases} \quad (1)$$

Proof. The base cases of Equation (1) consist of two parts. In the first row, an entry $F(0, j)$ means that all lines $s^v_{\pi(1)}, \ldots, s^v_{\pi(j)}$ terminate at position 0 in u and hence the required number of crossings is just the number of crossings of these lines with L_{uv}, which equals the sum given in Equation (1). In the first column,

an entry $F(i,1)$ reflects the situation that line $s^v_{\pi(1)}$ terminates at position i. The required number of crossings in this case is simply $\mathrm{cr}_v(i,1)$, the number of crossings of $s^v_{\pi(1)}$ with L_{uv}, plus $C(0,i,1)$, the number of crossings of $L^u_{0,i}$ with $L_{uv} \cup L_v$.

The general case of Equation (1) means that the value $F(i,j)$ can be composed of the optimal placement $F(k,j-1)$ of the lines below and including $s^v_{\pi(j-1)}$ (which itself terminates at some position k below i), the number $C(k,i,j)$ of crossings of lines in L_u in the interval between k and i, and the number of crossings $\mathrm{cr}_v(i,j)$ of $s^v_{\pi(j)}$ at position i.

Due to Lemma 1 we know that $s^v_{\pi(j)}$ cannot terminate below $s^v_{\pi(j-1)}$ in an optimal solution. Hence, for $s^v_{\pi(j)}$ terminating at position i, we know that $s^v_{\pi(j-1)}$ terminates at some position $k \leq i$. For each k we know by the induction hypothesis that $F(k,j-1)$ is the correct minimum number of crossings as defined above. In order to extend the configuration corresponding to $F(k,j-1)$ with the next line $s^v_{\pi(j)}$ in L_v we need to add two terms: (a) the number of crossings of $L^u_{k,i}$ with $L_{uv} \cup L_v$, which is exactly $C(k,i,j)$, and (b) the number $\mathrm{cr}_v(i,j)$ of crossings that the line $s^v_{\pi(j)}$ (terminating at position i) has with L_{uv}. Note that potential crossings of $s^v_{\pi(j)}$ with lines in $L^u_{k,i}$ are already considered in the term $C(k,i,j)$. Figure 8 illustrates this recursion: $s^v_{\pi(j)}$ is placed at position i in the order S_u, and $s^v_{\pi(j-1)}$ terminates at position k. The crossings of the configuration corresponding to $F(i,j)$ that are not counted in $F(k,j-1)$, are the $C(k,i,j)$ crossings of the marked, dotted lines of L_u (indicated by gray disks) and the $\mathrm{cr}_v(i,j)$ encircled ones of $s^v_{\pi(j)}$ with L_{uv}.

Finally, we have to show that taking the minimum value of the sum in Equation (1) for all possible terminal positions k of line $s^v_{\pi(j-1)}$ yields an optimal solution for $F(i,j)$. Assume to the contrary that there is a better solution $F'(i,j)$. This solution induces a solution $F'(k,j-1)$, where k is the position of $s^v_{\pi(j-1)}$ in S_u. Lemma 1 restricts $k \leq i$ and hence $s^v_{\pi(j-1)}$ runs completely below $s^v_{\pi(j)}$. Therefore we have $F'(k,j-1) \leq F'(i,j) - C(k,i,j) - \mathrm{cr}_v(i,j) < F(i,j) - C(k,i,j) - \mathrm{cr}_v(i,j) \leq F(k,j-1)$. This contradicts the minimality of $F(k,j-1)$. □

If we store in each cell $F(i,j)$ a pointer to the corresponding predecessor cell $F(k,j-1)$ that minimizes Equation (1) we can reconstruct the optimal edge layout: starting at the cell $F(i,n_v)$ that minimizes F^*, we can reconstruct the genesis of the optimal solution using backtracking. Obviously, using the combinatorial solution to place all endpoints of L_e in the correct order and then connecting them with straight-line segments results in a layout that has exactly F^* crossings in addition to cr_{uv}, the invariable number of crossings of L_{uv}.

Now, we can give a first, naive approach: As mentioned earlier the tables cr_u and cr_v can be computed in $O(n^2)$ time. For the computation of one cell entry $C(k,i,j)$ we only have to look at the at most n lines $L^u_{k,i}$ and their possible $n+1$ terminal positions in v. Once we have fixed a terminal position of a line $\ell \in L^u_{k,i}$, we have to compute the number of crossings that ℓ has with $L_{uv} \cup L_v$. For the crossings with L_{uv} we simply have to look at the corresponding value of cr_u. For

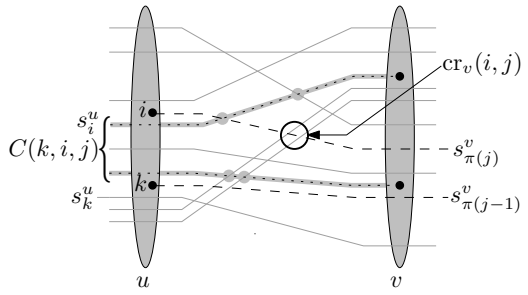


Fig. 8: The recursion for $F(i,j)$: lines $s^v_{\pi(j-1)}$ and $s^v_{\pi(j)}$ terminate at pos. k and i, resp.
$C(k,i,j)$ = min. # crossings of the marked dotted lines $L^u_{k,i}$ with $L_{uv} \cup L_v =^{\text{here}} 4$
$\mathrm{cr}_v(i,j)$ = number of crossings of $s^v_{\pi(j)}$ with $L_{uv} =^{\text{here}} 2$

the crossings with L_v it is sufficient to look at the index of the terminal position because we know that position k is the terminal position of $s^v_{\pi(j-1)}$ and position i is the terminal position of $s^v_{\pi(j)}$. Thus, computing one of the $O(n^3)$ cells of the table C requires $O(n^2)$ time, so in total we need $O(n^5)$ time for filling C. This dominates the computation of the table F. In the remainder of this section we show how to speed up the computation of F and C.

Improving the Running Time. Let us—for the moment—assume that the values $C(k,i,j)$ and $\mathrm{cr}_v(i,j)$ are available in constant time. Then the computation of the $(n+n_u+1) \times n_v$ matrix F still needs $O(n^3)$ time because the minimum in Equation (1) is over a set of $O(n)$ elements. The following series of lemmas shows how we could bring the running time down to $O(n^2)$. First we show a relation for the entries of the matrix C.

Lemma 2. *$C(k,i,j)$ is additive in the sense that $C(k,i,j) = C(k,l,j)+C(l,i,j)$ for $k \le l \le i$.*

Proof. Since $C(k,i,j)$ denotes the number of crossings of the lines in $L_u \cap \{s^u_{k+1}, \ldots, s^u_i\}$ and no two of these lines intersect each other (recall Lemma 1) we can split the layout corresponding to $C(k,i,j)$ at any position l, $k \le l \le i$ and get two (possibly non-optimal) configurations for the induced subproblems. This implies $C(k,i,j) \ge C(k,l,j) + C(l,i,j)$.

Conversely, we can get a configuration for $C(k,i,j)$ by putting together the optimal solutions of the subproblems. W.l.o.g. this introduces no additional crossings (they could be removed as in the proof of Lemma 1). Hence we have $C(k,i,j) \le C(k,l,j) + C(l,i,j)$. □

Now we show that we do not need to compute all entries of C.

Lemma 3. *Given the matrix C, the matrix F can be computed in $O(n^2)$ time.*

Proof. Having computed entry $F(i-1,j)$ we can compute $F(i,j)$ in constant time as follows:

$$F(i,j) = \min \begin{cases} F(i-1,j) + C(i-1,i,j) - \mathrm{cr}_v(i-1,j) + \mathrm{cr}_v(i,j), \\ F(i,j-1) + \mathrm{cr}_v(i,j). \end{cases} \tag{2}$$

The correctness follows from Equation (1), Lemma 2, and the fact that $C(i,i,j)$ vanishes:

$$\begin{aligned} F(i,j) &\overset{(1)}{=} \min \begin{cases} \min\limits_{k<i}\{F(k,j-1) + C(k,i,j) + \mathrm{cr}_v(i,j)\}, \\ F(i,j-1) + C(i,i,j) + \mathrm{cr}_v(i,j) \end{cases} \\ &\overset{\text{L. 2}}{=} \min \begin{cases} \min\limits_{k\le i-1}\{F(k,j-1) + C(k,i-1,j) + C(i-1,i,j) + \mathrm{cr}_v(i,j)\}, \\ F(i,j-1) + \mathrm{cr}_v(i,j) \end{cases} \\ &\overset{(1)}{=} \min \begin{cases} F(i-1,j) - \mathrm{cr}_v(i-1,j) + C(i-1,i,j) + \mathrm{cr}_v(i,j), \\ F(i,j-1) + \mathrm{cr}_v(i,j) \end{cases} \end{aligned}$$

In the first column, we can reformulate the recursion for $i \geq 1$ as follows:

$$\begin{aligned} F(i,1) &\overset{(1)}{=} C(0,i,1) + \mathrm{cr}_v(i,1) \\ &\overset{\text{Lemma 2}}{=} C(0,i-1,1) + C(i-1,i,1) + \mathrm{cr}_v(i,1) \\ &\overset{(1)}{=} F(i-1,1) - \mathrm{cr}_v(i-1,1) + C(i-1,i,1) + \mathrm{cr}_v(i,1) \end{aligned}$$

Hence the whole matrix F can be computed in $O([n+n_u]\cdot n_v) = O(n^2)$ time. □

Observe that due to the reformulation in Lemma 3 we only need the values $C(i-1,i,j)$ explicitly in order to compute F. Now we will show that we can compute these relevant values in $O(n^2)$ time. For simplification we introduce the following notation: $C'(i,j) := C(i-1,i,j)$.

Lemma 4. *The values $C'(i,j)$ $(i = 1,\ldots,n+n_u,\ j = 1,\ldots,n_v)$ can be computed in $O(n^2)$ time.*

Proof. We compute $C'(i,j)$ row-wise, i.e., we fix i and increase j. Recall that $C(k,i,j)$ was defined as the minimal number of crossings of the lines in $L^u_{k,i} = \{s^u_{k+1},\ldots,s^u_i\} \cap L_u$ with the lines in $L_{uv} \cup L_v$ under the condition that $s^v_{\pi(j-1)}$ ends at position k and $s^v_{\pi(j)}$ ends at position i. For $C'(i,j) = C(i-1,i,j)$ this means we have to consider crossings of the set $L^u_{i-1,i} = \{s^u_i\} \cap L_u$. Hence, we distinguish two cases: either s^u_i is a line in L_{uv} and then $L^u_{i-1,i}$ is empty or $s^u_i \in L_u$ and we have to place the line s^u_i optimally. Clearly, in the first case we have $C'(i,j) = 0$ for all j as there is no line to place in $L^u_{i-1,i}$.

Now we consider the case that $s^u_i \in L_u$. For each j we split the set of candidate terminal positions for s^u_i into the intervals $[0,\pi(j-1))$, $[\pi(j-1),\pi(j))$, and $[\pi(j),n+n_v]$. Let $\mathrm{LM}(i,j)$, $\mathrm{MM}(i,j)$, and $\mathrm{UM}(i,j)$ denote the minimum number of crossings of s^u_i with $L_{uv} \cup L_v$ for terminal positions in $[0,\pi(j-1))$, $[\pi(j-1),\pi(j))$, and $[\pi(j),n+n_v]$, respectively. Now we have $C'(i,j) = \min\{\mathrm{LM}(i,j),\mathrm{MM}(i,j),\mathrm{UM}(i,j)\}$ as the minimum of the three distinct cases. The situation is illustrated in Figure 9.

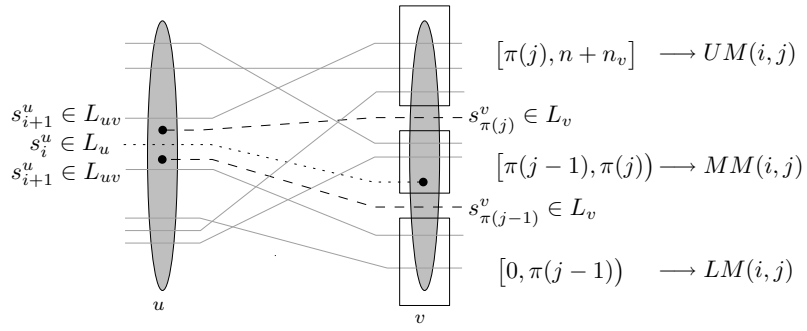


Fig. 9: Splitting the candidate terminal positions for s^u_i into three intervals w.r.t. $\pi(j-1)$ and $\pi(j)$.

Next, we have to show how to compute MM, LM, and UM. First, we consider MM. Recall that $\mathrm{cr}_u(i,j)$ was defined as the number of crossings of the lines in L_{uv} with the line $s^u_{\mu(i)}$ that terminates at position j in v. It follows that

$$\mathrm{MM}(i,j) = \min\{\mathrm{cr}_u(\mu^{-1}(i),k) \mid \pi(j-1) \le k < \pi(j)\}, \tag{3}$$

where $\pi(0)$ is defined as 0. Because of Lemma 1 there are no lines of L_v intersecting the tunnel between $s^v_{\pi(j-1)}$ and $s^v_{\pi(j)}$. Hence, if the line s^u_i terminates at position $k \in [\pi(j-1), \pi(j))$ it does not cross any line of L_v and Equation (3) is correct. We can calculate $\mathrm{MM}(i,j)$ by a straight-forward minimum computation through $j = 1, \ldots, n_v$ which takes $O(n+n_v) = O(n)$ time for each value of i.

Secondly, we consider LM. Initially, in the case that $j = 1$ there is no line $s^v_{\pi(j-1)}$ and the corresponding interval is empty. Hence we set $\mathrm{LM}(i,1) = \infty$. Then we recursively compute

$$\mathrm{LM}(i,j+1) = \min\{\mathrm{LM}(i,j)+1, \mathrm{MM}(i,j)+1\}. \tag{4}$$

Observe that for $\mathrm{LM}(i,j+1)$ we merge the previous intervals corresponding to $\mathrm{LM}(i,j)$ and $\mathrm{MM}(i,j)$. Moreover the line $s^v_{\pi(j)}$, which previously ended at position i, now terminates at position $i-1$. Hence, in order to reach its terminal position in the interval $[0, \pi(j))$, the line s^u_i has to cross $s^v_{\pi(j)}$ in addition to the crossings counted before by $\mathrm{MM}(i,j)$ and $\mathrm{LM}(i,j)$. This explains the recursion in Equation (4). The computation again requires $O(n)$ time for each value of i.

Finally, we initialize $\mathrm{UM}(i,n_v) = 1 + \min\{\mathrm{cr}_u(\mu^{-1}(i),k) \mid \pi(n_v) \le k \le n+n_v\}$ as for $j = n_v$ the line s^u_i crosses the line $s^v_{\pi(n_v)}$ but no other line of L_v. In decreasing order we compute

$$\mathrm{UM}(i,j-1) = \min\{\mathrm{UM}(i,j)+1, \mathrm{MM}(i,j)+1\} \tag{5}$$

analogously to LM, which again requires $O(n)$ time. As the whole procedure needs linear time for each $i = 1, \ldots, n+n_u$ the total running time is $O(n^2)$. □

Putting the intermediate results in Lemmas 3 and 4 together, we conclude:

Theorem 2. *The one-edge layout problem can be solved in $O(n^2)$ time.*

So far, the algorithm requires $O(n^2)$ space to store the tables F, C', cr_v, and cr_u. If we are only interested in the minimum *number* of crossings this can easily be reduced to $O(n)$ space as all tables can be computed row-wise: in F we need only two consecutive rows at a time and we can discard previous rows; in the other tables the rows are independent and can be computed on demand. This does not affect the time complexity. However, to restore the optimal placement we need the pointers in F to do backtracking and hence we cannot easily discard rows of the matrix. But we can still reduce the required space to $O(n)$ with a method similar to a divide-and-conquer version of the Needleman-Wunsch algorithm for biological sequence alignment [2] adding a factor of 2 to the time complexity. Basically, the idea is to keep only the pointers in one column of the matrix to reconstruct the optimal position of the corresponding line. This line cuts the problem into two smaller subproblems which are solved recursively.

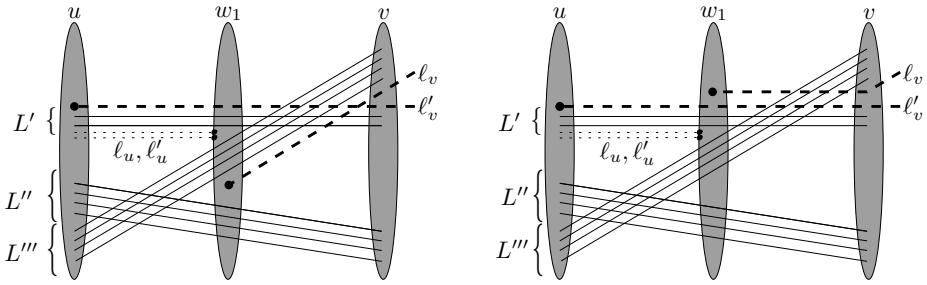
3 Generalization to a Path

Surely it is desirable to draw lines on a more general fraction of the graph than only on a single edge. However, the problem seems to become significantly harder even for two edges. Let us first define the problem on a path.

Problem 2. Path layout

Given a graph $G = (V, E)$ and a simple path $P = \langle u, w_1, \ldots, w_m, v\rangle$ in G. Let L_P be the set of lines that use at least one edge in P. We split L_P into three subgroups: L^P_{uv} is the set of lines that have no terminal station in $\{u, w_1, \ldots, w_m, v\}$, L^P_u is the set of lines for which u is an intermediate station and that have a terminal station in $\{w_1, \ldots, w_m, v\}$, and L^P_v is the set of lines for which v is an intermediate station and that have a terminal station in $\{u, w_1, \ldots, w_m\}$. We assume that there are no lines having both terminal stations in $\{u, w_1, \ldots, v\}$ as these could be placed top- or bottommost causing the minimum number of crossings with lines in L^P_{uv}. We also assume that any two lines ℓ_1, ℓ_2 that use exactly the subpath $x, \ldots, y \subseteq P$ together and for which neither x nor y is a terminal station enter P in x in a predefined order and leave P in y in a predefined order. The task is to find a layout of the lines in L_P such that the number of pairs of intersecting lines is minimized.

We tried to apply the same dynamic-programming approach as for the one-edge case. However, the dilemma is that the generalized version of Lemma 1 does not hold, namely that no two lines in L^P_v intersect. Thus, the problem instance cannot be separated into two independent subproblems, which seems to forbid dynamic programming. In Figure 10a we give an instance where two lines of L^P_v cross in the optimal solution. Here, the lines in L^P_{uv} are drawn solid. Recall that we do not have to take the intersections of lines in L^P_{uv} into account as these are again given by the orders in S_u and S_v. The two lines ℓ_u and $\ell'_u \subseteq L^P_u$ are only needed to force the bundle crossings between the bundles $L' \subset L^P_{uv}$ and



(a) Optimal solution (7 crossings). (b) Manipulating ℓ_v (8 crossings).

Fig. 10: The two lines ℓ_v,$\ell'_v \in L_v^P$ cross in the optimal solution.

$L''' \subset L_{uv}^P$ to be on the edge $\{w_1, v\}$. In the optimal solution the lines $\ell_v, \ell'_v \in L_v^P$ intersect causing a total number of 7 crossings between lines in L_v^P with lines in $L_v^P \cup L_{uv}^P$. We have to argue that any solution in which ℓ_v and ℓ'_v do not cross produces more than 7 crossings. We look at the optimal solution and argue that getting rid of the crossing between ℓ_v and ℓ'_v by manipulating the course of either ℓ_v or ℓ'_v produces at least 8 crossings. First, we consider manipulating the course of ℓ_v, see Figure 10b. However then, as ℓ_v has to be above ℓ'_v, it has to cross the 4 lines in the bundle L''' resulting in a total number of 8 crossings. Similarly, manipulating the course of ℓ'_v would also result in at least 8 crossings.

Concluding Remarks

Clearly our work is only a first step in exploring the layout of lines in graphs. What is the complexity of the problem if two edges of the underlying graph are considered, what about longer paths, trees and finally, general plane graphs? A variant of the problem where lines must terminate bottom- or topmost in their terminal stations is also interesting. This requirement prevents gaps in the course of continuing lines.

References

1. P. F. Cortese, G. D. Battista, M. Patrignani, and M. Pizzonia. On embedding a cycle in a plane graph. In P. Healy and N. S. Nikolov, editors, *Proc. 13th Int. Symp. Graph Drawing (GD'05)*, volume 3843 of *LNCS*, pages 49–60. Springer-Verlag, 2006.
2. R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis.* Cambridge University Press, 1998.
3. M. R. Garey and D. S. Johnson. Crossing number is NP-complete. *SIAM J. Alg. Disc. Meth.*, 4:312–316, 1983.
4. M. Nöllenburg and A. Wolff. A mixed-integer program for drawing high-quality metro maps. In P. Healy and N. S. Nikolov, editors, *Proc. 13th Int. Symp. Graph Drawing (GD'05)*, volume 3843 of *LNCS*, pages 321–333. Springer-Verlag, 2006.