Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 27, no. 6, pp. 489–522 (2023)
DOI: 10.7155/jgaa.00631

Inserting Multiple Edges into a Planar Graph

Markus Chimani[1] *Petr Hliněný*[2]

[1]Theoretical Computer Science, University Osnabrück, Germany
[2]Faculty of Informatics, Masaryk University, Brno, Czech Republic

Submitted: May 2022 Reviewed: October 2022 Revised: December 2022
Accepted: April 2023 Final: April 2023 Published: July 2023
Article type: Regular paper
Communicated by: G. Da Lozzo and P. Kindermann

Abstract. Let G be a connected planar (but not yet embedded) graph and F a set of edges with ends in $V(G)$ and not belonging to $E(G)$. The *multiple edge insertion* problem (MEI) asks for a drawing of $G + F$ with the minimum number of pairwise edge crossings, such that the subdrawing of G is plane. A solution to this problem is known to approximate the crossing number of the graph $G + F$, but unfortunately, finding an exact solution to MEI is NP-hard for general F. The MEI problem is linear-time solvable for the special case of $|F| = 1$ (SODA 01 and Algorithmica), and there is a polynomial-time solvable extension in which all edges of F are incident to a common vertex which is newly introduced into G (SODA 09). The complexity for general F but with constant $k = |F|$ was open, but algorithms both with relative and absolute approximation guarantees have been presented (SODA 11, ICALP 11 and JoCO). We present a fixed-parameter algorithm for the MEI problem in the case that G is biconnected, which is extended to also cover the case of connected G with cut vertices of bounded degree. These are the first exact algorithms for the general MEI problem, and they run in time $O(|V(G)|)$ for any constant k.

1 Introduction

The crossing number $\mathrm{cr}(G)$ of a graph G is the minimum number of pairwise edge crossings in a drawing of G in the plane. Finding the crossing number of a graph is one of the most prominent, difficult optimization problems in graph theory [22] and is NP-hard already in very restricted cases, e.g., even when considering a planar graph with one additional edge [8] (such graphs are

Special Issue on Parameterized and Approximation Algorithms in Graph Drawing

A short conference version of this research appeared at SoCG 2016; this is the full enhanced version.

E-mail addresses: markus.chimani@uni-osnabrueck.de (Markus Chimani) hlineny@fi.muni.cz (Petr Hliněný)

called *almost-planar* or near-planar). The problem has been intensely investigated for over 60 years, but there is still surprisingly little known about it; see e.g. Schaefer [39, 40] for extensive references. There exists a $c > 1$ such that $\mathrm{cr}(G)$ cannot be approximated within a factor c in polynomial time [6], but we do not know whether $\mathrm{cr}(G)$ is approximable within some constant ratio for general G.

One notable positive algorithmic aspect is that there are fixed-parameter (FPT) algorithms for the crossing number problem parameterized by the solution value. Grohe [24] showed that $\mathrm{cr}(G) \leq k$ can be decided in quadratic time for any fixed k, and Kawarabayashi and Reed [33] improved that to linear time. Neither algorithm is practical, since they rely on parts of the graph-minor machinery and Courcelle's theorem. Over time, several involved approximation algorithms have been found for graphs of bounded maximum degree. There is the (nowadays classical) algorithm by Even, Guha and Schieber that approximates the quantity $n + \mathrm{cr}(G)$ instead of $\mathrm{cr}(G)$, giving an $\mathcal{O}(\log^2 n)$-approximation [3, 21]. This, however, gives only an $\tilde{\mathcal{O}}(n)$-approximation of $\mathrm{cr}(G)$ in the worst case. Algorithms achieving a sublinear approximation ratio have been given in [17, 34], and this line of research has seen a quite recent major breakthrough in the paper of Chuzhoy and Tan [19] giving the first randomized algorithm achieving a subpolynomial approximation for graphs of bounded degree.

Better approximation results can be obtained for more restricted graph classes. We know polynomial-time constant factor approximations of the crossing number for bounded-degree graphs that are embeddable in some surface of higher genus [15, 23, 29], and for graphs of bounded path-width [4]. For graphs of bounded-size vertex cover, we can even compute the crossing number exactly in FPT time [31]. Another case of polynomial-time constant factor approximations occurs when we have a small set of graph elements (in general vertices and edges) whose deletion leaves a planar graph—removing and re-inserting these elements can give strong approximation bounds such as [7, 14, 18, 30].

Here we study the *Multiple Edge Insertion* problem $\mathrm{MEI}(G, F)$ (see Definition 2.1). Intuitively, given a planar graph G we ask for a planar drawing of G such that inserting the edges of F (which are assumed to have end vertices in $V(G)$) into the drawing minimizes the number of crossings in $G + F$, the combined drawing of G and F. Note that finding a suitable planar drawing of G is a (nontrivial, in fact) part of the problem solution. It also turns out useful to study the *rigid* setting in which we require that the planar drawing of G remains as given.

The MEI problem is linear-time solvable for $|F| = 1$ [26] (unlike the crossing number problem of almost-planar graphs), and there is a polynomial-time solvable extension in which all edges of F are incident to a common vertex which may be newly introduced into G [12]. Note that, with unrestricted G (namely, G with no edges), the $\mathrm{MEI}(G, F)$ problem is obviously at least as hard as the ordinary crossing number problem.

An exact or at least approximate MEI solution constitutes an approximation for the crossing number of the graph $G+F$ [14]; see Theorem 2.2. Considering the cardinality $k := |F|$ as a general integer parameter, there have been two different polynomial-time approximation approaches by Chuzhoy, Makarychev and Sidiropoulos [18] and by the authors [13]; the former one directly targets the crossing number and achieves a multiplicative approximation guarantee for MEI; the latter one first attains an approximation of MEI with only an additive error term, and then uses [14] to deduce a multiplicative crossing number approximation. Both approaches assume bounded degrees. While the former is not directly practical, the algorithm from [13] is one of the best choices to obtain strong upper bounds on the crossing number in practice [11]; this still holds true when combining solving MEI with a post-processing step based on reinserting edge sets F with a common vertex [16].

In this paper we develop an exact, linear-time algorithm for MEI, for every fixed k, making

some mild connectivity assumptions. This has been an open problem since [26] even for $k = 2$.

Theorem 1.1. *Let G be a connected planar graph and F a set of $k \geq 1$ edges to insert, with ends in $V(G)$. Assume that all cut vertices of G have degree bounded from above by $2^{p(k)}$ where p is a polynomial (in particular, this assumption is void if G is biconnected). Then there is a polynomial function q such that the problem* MEI(G, F) *is solvable to optimality in time $\mathcal{O}(2^{q(k)} \cdot |V(G)|)$.*

In terms of parameterized complexity, we get that the problem MEI(G, F) is linear-time FPT with the parameter $k = |F|$.

Our high-level approach to a proof of Theorem 1.1 is a standard idea in this area, using dynamic programming on a decomposition (known as SP(Q)R-tree) of the planar graph; see Section 4. Similar ideas have been used, e.g., in [12,13,18,26]. However, this time it turns out that the most interesting and difficult case is the basic one of rigid components. The corresponding problem *Rigid MEI*, i.e., MEI under the restriction that a planar drawing of G is fixed, is NP-hard in general, even when G is 2-connected [43].

An FPT algorithm for Rigid MEI is given in Section 3. On an informal level, the algorithm for Rigid MEI simultaneously searches for shortest dual paths corresponding to the edges of F in rigid G, while keeping track of their mutual crossings. Although this task looks similar, in the dual, to the difficult problem of shortest disjoint paths in planar graphs [20,36], there is the crucial difference that our paths may share common subpaths as long as they do not cross. Our algorithm utilizes the concept of *path homotopy* in the plane with obstacles, and a special structure which we call a *trinet*, to represent and search for shortest dual paths of a given homotopy with the standard geometric funnel algorithm [9,37].

Closely related to the Rigid MEI problem is the problem recently named *partially predrawn crossing number* in [27] (while that problem was first suggested in [38] under yet another name, it did not receive more attention until recently). In a nutshell, [27] can be seen as solving, again in FPT time, the Rigid MEI with respect to an arbitrary set F of inserted edges (and vertices), and G need not be connected. The parameter is $\mathrm{cr}(G + F)$ which makes the FPT-algorithm of [27] incomparable with Theorem 1.1, because $\mathrm{cr}(G + F)$ may be small even with very large F and, on the other hand, planar G with just one inserted edge may have unbounded crossing number. The algorithm in [27] is also very different from the one here in Section 3.

Organization. After some basic definitions in Section 2, we show how to solve the *Rigid MEI* problem in Section 3, as stated in Theorem 3.16. We use that solution in a dynamic programming procedure for the general MEI problem in Section 4, stated in Theorem 4.3, which proves Theorem 1.1.

2 Preliminaries

We use the standard terminology of graph theory. By default, we use the term *graph* to refer to a multigraph, i.e., we allow parallel edges (self-loops are allowed but they can be safely ignored in the context of crossing numbers in the plane). If there is no danger of confusion, we denote an edge with the ends u and v by uv.

A problem featuring an arbitrary integer parameter k as a (separate or implicit) part of the input is *fixed-parameter tractable*, or FPT, if it allows an algorithm running in *FPT time*, meaning runtime $\mathcal{O}(f(k) \cdot n^c)$ where n is the input size, c a constant, k the parameter value, and f an arbitrary computable function.

Drawing a graph. A *drawing* of a graph $G = (V, E)$ is a mapping of the vertices V to distinct points on a surface Σ, and of the edges E to simple curves on Σ, connecting their respective endpoints but not containing any other vertex point. Unless explicitly specified, we will always assume Σ to be the plane (or, equivalently, the sphere). To keep things simple, we will moreover assume each curve to be a polygonal curve, which is a union of finitely many line segments (or circular-arc segments in the case of a sphere). A *crossing* is a common point of two distinct edge curves, other than a common endpoint. A drawing is *plane* if there are no crossings. *Plane embeddings* form equivalence classes over plane drawings of connected graphs, in that they only define the cyclic order of the edges around their incident vertices (and, if desired, the choice of the outer, infinite face). A *planar* graph is one that admits a plane embedding. A *plane* graph is an embedded graph, i.e., a planar graph together with a plane embedding.

Given a plane embedding G_0 of G, we define its geometric *dual* G_0^* as the embedded multigraph that has a (dual) vertex for each face in G_0; (dual) vertices are joined by a (dual) edge for each (primal) edge shared by their respective (primal) faces. The cyclic order of the (dual) edges around any common incident (dual) vertex v^*, is induced by the cyclic order of the (primal) edges around the (primal) face corresponding to v^*.

We refer to a path/walk in G_0^* as a *dual path/walk* in G_0, and we speak about a *dual path/walk* π *in* G_0 *between vertices* u, v if π starts in a face incident to u and ends in a face incident to v. We shortly say a *route from* u *to* v (a u–v *route*) to refer to a dual walk between vertices u, v (recall that a walk, unlike a path, may repeat vertices and edges).

Crossing numbers and edge insertion. Given a drawing D of G, let $\mathrm{cr}(D)$ denote the number of pairwise edge crossings in D. The *crossing number* problem asks for a drawing D° of a given graph G with the least possible number $\mathrm{cr}(D^\circ) =: \mathrm{cr}(G)$. By saying "pairwise edge crossings" we emphasize that we count a crossing point x separately for every pair of edges meeting in x (e.g., ℓ edges meeting in x give $\binom{\ell}{2}$ crossings).

It is well established that the search for optimal solutions to the crossing number problems can be restricted to so-called *good* drawings: any pair of edges crosses at most once, adjacent edges do not cross, and there is no point that is a crossing of three or more edges.

In this paper we especially consider the following variant of the crossing number problem:

Definition 2.1 (Multiple edge insertion, Rigid MEI and MEI)**.**
Consider a planar, connected graph G and a set F of edges with ends in $V(G)$ (a multiset of vertex pairs, in fact). We denote by $G + F$ the graph on the vertex set $V(G)$ and the edge set $E(G) \dot\cup F$, and say that we *insert* the (new) edges F to G.

Let G_0 be a plane embedding of G. The *Rigid Multiple Edge Insertion* problem r-MEI(G_0, F) is to find a drawing D of the graph $G + F$ with minimal $\mathrm{cr}(D)$ such that the restriction of D to G is the plane embedding G_0. The attained number of crossings is denoted by r-ins(G_0, F).

The *Multiple Edge Insertion* problem MEI(G, F) is to find an embedding G_1 of G (together with the subsequent drawing D as above), for which r-MEI(G_1, F) attains the minimum number of crossings. The latter value is denoted by $\mathrm{ins}(G, F)$. ◇

As mentioned above, a solution of a MEI(G, F) instance—which is a trivial upper bound on $\mathrm{cr}(G + F)$, readily gives an approximate solution of the crossing number problem of $G + F$ by the following inequality:

Theorem 2.2 (see [14, Theorem 7])**.** *Consider a planar graph* G *and a multiset of edges* F *with ends in* $V(G)$*. Then* $\mathrm{ins}(G, F) \le |F| \cdot \Delta(G) \cdot \mathrm{cr}(G + F) + \binom{|F|}{2}$*, where* $\Delta(G)$ *is the maximum degree in* G*.*

3 Rigid MEI

We first give an FPT algorithm for solving the rigid version r-MEI(G, F), parameterized by $k = |F|$. Throughout this section, G is a plane graph with a fixed embedding. Recall that the r-MEI(G, F) problem is NP-hard [43] for unrestricted k.

We first illustrate the simple cases. Solving r-MEI$(G, \{uv\})$, the fixed embedding edge insertion problem with $k = 1$, is trivial. Augment the dual G^* with edges of length 0 between the terminals u, v (which are added as new vertices into G^*) and their respective incident faces (vertices in G^*), to suit the definition of a u–v route in G. Then, simply compute the shortest u–v route in this graph. *Realizing* a route for uv means to draw uv along it within G. If the shortest route has length ℓ, realizing it attains r-ins$(G, \{vw\}) = \ell$, the smallest number of crossings in the Rigid MEI setting.

For $k \geq 2$, the situation starts to be more interesting: not every collection of shortest routes gives rise to an optimal solution of r-MEI(G, F) since there might arise crossings between edges of F. While for $k = 2$, the only question is whether some pair of shortest routes of the two edges in F can avoid crossing each other, for larger values of k we can encounter situations in which all the optimal solutions of r-MEI(G, F) draw some edges of F quite far from their individual shortest routes (in order to avoid crossings with other edges of F), and a more clever approach is needed.

On a very high level, our approach to finding a drawing D of $G + F$ that is an optimal solution to r-MEI(G, F), can be described as follows:

(I) We guess, for each pair $f, f' \in F$, whether f and f' will cross each other in D. Since $k = |F|$ is a parameter, all the possibilities can be enumerated in FPT time.

(II) Let $X \subseteq \binom{F}{2}$ be a (guessed) set of pairs of edges of F. We find a collection of shortest routes for the edges of F in G under the restriction that exactly the pairs in X cross; Drawing D_X is obtained by inserting the edges of F along their computed routes. As we will see, we may restrict our attention to routes pairwise crossing at most once.

(III) We select $D := D_X$ which minimizes the sum of $|X|$ and of the lengths of the routes.

3.1 Handling path homotopy of routes

Obviously, the core task of the scheme (I)–(III) is to solve the point (II) of finding a collection of shortest routes under the restriction that every route avoids crossing certain other routes (note; none of these routes are fixed in advance). The key to this is the concept of *path homotopy* in the plane with point obstacles.

In a brief and rather informal topological view, consider the sphere with a finite set of point obstacles. Two simple curves α, α' with the same endpoints are *homotopic* if there exists a homeomorhpism (a continuous deformation) of α to α' that fixes the endpoints and otherwise avoids all the obstacles. For example, if α, α' are disjoint except at the common ends, then they are homotopic if and only if one of the two open regions bounded by $\alpha \cup \alpha'$ is obstacle-free. In our case, the *obstacles* are the ends $V(F)$ of the edges of F (as given by the fixed embedding of G), where each endpoint is "blown up" into a small open disc. Then, given the homotopy classes $hom(\alpha), hom(\beta)$ of two curves α, β, one can decide whether α and β are "forced to cross"—although, α and β may cross if they are not forced to, such unforced crossings can be avoided in our case.

Instead of the above classical algebraic-topology setting of homotopies, in this paper we choose to deal with path homotopy in a combinatorial setting of T-sequences as in Definition 3.2. This

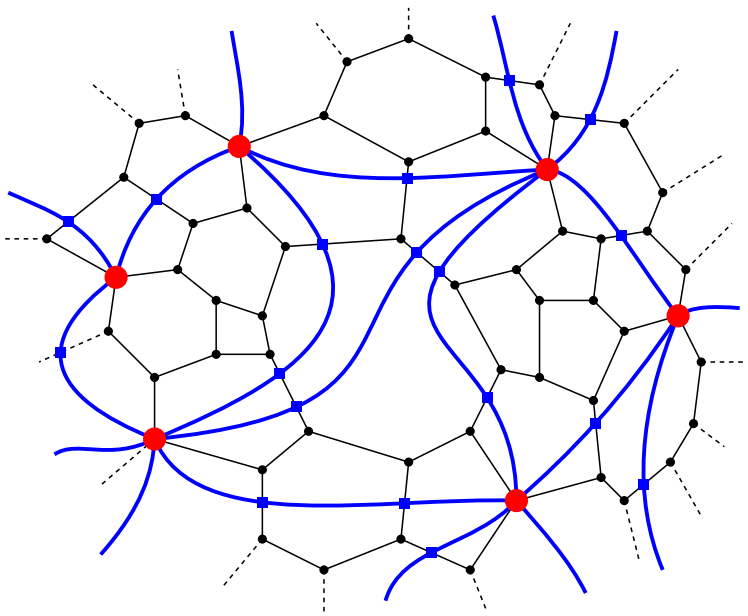


Figure 1: An example of a trinet of a plane graph G (see Definition 3.1): underlying G is in thin black, the trinodes in red, and the triedges in thick blue with square blue nodes subdividing edges of G.

new setting is closely inspired by the discrete-geometry view of boundary-triangulated 2-manifolds by Hershberger and Snoeyink [28]:

First, we "triangulate" the point set $V(F)$ (our obstacles) using transversing paths in the embedding G. A *transversing path* between vertices x, y of G is a path whose ends are x, y and whose internal vertices subdivide some edges of G. Let T be the union of these transversing paths and G' denote the corresponding subdivision of G. In order to avoid a terminology clash with graph triangulations, we will call T in the pair (G', T) a *trinet of* G. See Figure 1 for an illustration of a trinet of a plane graph. We will apply the following formal definition with $V(F) = N$:

Definition 3.1 (Trinet)**.** Let G be a connected plane graph and $N \subseteq V(G)$, $|N| \geq 4$. A plane graph T such that $V(T) \cap V(G) = N$ is called a *trinet* of G if the following holds:

a) T is a subdivision of a 3-connected plane triangulation on the vertex set N (in particular, every face of T is incident to precisely three distinct vertices of N), and

b) there exists a subdivision G' of G such that $V(G') \setminus V(G) = V(T) \setminus N$, $E(G') \cap E(T) = \emptyset$ and the union $G' \cup T$ is a plane embedding.

The pair (G', T) is a *full trinet* of G. The vertices in $N(T) := N$ are called *trinodes* of T, the maximal paths in T internally disjoint from N are *triedges* and their set is denoted by $I(T)$, and the faces of T are *tricells*. Note that the triedges of T are transversing paths of G. ◇

We need to introduce terms related to (and describing) a path homotopy in a full trinet (G', T) of a plane graph G. See Figure 2 for an illustration of the definition.

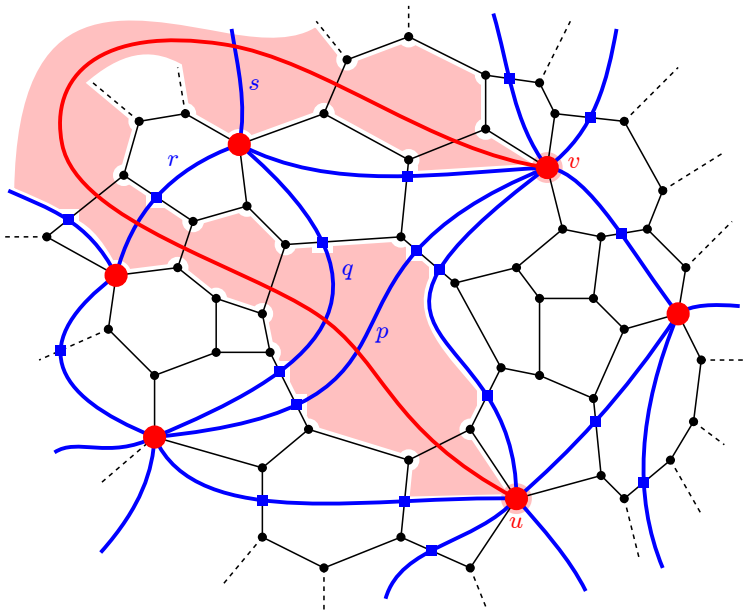


Figure 2: An illustrating example (see Definition 3.2): the T-sequence of the u–v route depicted in red is (p, q, r, s) from u to v. It is a proper T-sequence from u to v (Definition 3.6); its corresponding alley is shaded in light red.

Definition 3.2 (Alley and T-sequence)**.** Let (G', T) be a full trinet of a plane graph G. Consider a route π between $u, v \in V(G)$ in the graph $G' \cup T$. Then $V(\pi) = \{\phi_0, \phi_1, \dots, \phi_m\}$ where each dual vertex ϕ_i of π is an open face of $G' \cup T$. Let these faces $(\phi_0, \phi_1, \dots, \phi_m)$ be ordered along π such that ϕ_0 is incident to u and ϕ_m incident to v. Let $(e_1, e_2, \dots, e_m) \subseteq E(G' \cup T)$ be the sequence of the primal edges of the dual edges of π, ordered from ϕ_0 to ϕ_m. As a point set, each edge e_i is considered without the endpoints.

a) The union $\{u, v\} \cup \bigcup_{i=0}^{m} \phi_i \cup \bigcup_{i=1}^{m} e_i$ is called the *alley of* π (or, an *alley between* u, v).

b) Let $(e'_1, \dots, e'_\ell) \subseteq (e_1, e_2, \dots, e_m)$ be the restriction to $E(T)$, and let $(p_1, p_2, \dots, p_\ell) \subseteq I(T)$ be the sequence of triedges such that p_i contains the edge e'_i for $i = 1, \dots, \ell$. Then $(p_1, p_2, \dots, p_\ell)$ is called the *T-sequence of* π from u to v (or, of the corresponding alley from u to v). ◇

The purpose of introducing an alley is to describe a topological corridor for all u–v arcs of a similar kind (and same number of crossings) in the embedding $G' \cup T$. The correspondence is clear: a route π *crosses* a triedge p if the alley of π contains at least one of the G'-edges forming p. The T-sequence of π hence describes the unique order (with repetition) in which π crosses the triedges of T. Usually, we shall consider only the case of $u, v \in N(T)$.

A route may, in general, cross the same triedge many times, but we aim to prove that for "reasonable" routes there is an explicit upper bound; see Lemma 3.8.

Some situations of multiple crossings of the same triedge are resolved quite easily. This resolution can be formalized by the notion of reducing a T-sequence as follows: if $S = (p_1, p_2, \dots, p_\ell)$ is a T-sequence such that $p_i = p_{i+1}$ for some $1 \le i < \ell$, then the subsequence $S' = (p_1, \dots, p_{i-1}, p_{i+2}, \dots, p_\ell)$ is called a one-step reduction of S. A subsequence $S^* \subseteq S$ is a *reduction* of S (or S

reduces to S^*) if S^* results from a sequence of one-step reductions of S.

It comes as no surprise that our T-sequences are closely related to the path homotopy concept via reductions:

Remark 3.3. Consider a trinet T in the sphere. One can show that two arcs with the same fixed endpoints are path-homotopic (in the sphere with the obstacles formed by the trinodes of T) if, and only if, their T-sequences can be reduced to the same subsequence. However, since we are not going to directly use this fact, we refrain from giving it as a formal statement in the paper.

3.2 Refined approach to Rigid MEI

We slightly generalize the shortest route setting to allow for a connected plane graph G with *integer edge weights* $w\colon E(G) \to \mathbb{N}_+ \cup \{\infty\}$. For a full trinet (G', T) of G, we define the edge weights of $G' \cup T$ as follows: $w'(p) := 0$ for all $p \in E(T)$ and $w'(e') := w(e)$ where $e' \in E(G')$ is obtained by subdividing $e \in E(G)$. We call w' the *weights induced by* w in the trinet (G', T). We give the same weights w' also to the edges of the geometric dual of $G' \cup T$. This way we rigorously define the length of any route π between vertices x, y in $G' \cup T$ (recall that a route π between vertices x, y is a dual path between a face incident to u and a face incident to v), as the sum of the w'-weights of the dual edges of π. If α is the alley of a route π between vertices x, y in $G' \cup T$, then the *length of* α equals the length of π.

For any weighted graph G' with $w\colon E(G') \to \mathbb{N}_+ \cup \{\infty\}$, as above, we correspondingly generalize the notion of a crossing number to the *weighted crossing number* as follows: a crossing between two edges $e_1, e_2 \in E(G')$ contributes the amount of $w(e_1) \cdot w(e_2)$ to the crossing number $\mathrm{cr}(G')$. For MEI(G, F) problem variants with weighted G, we shall always assume that the weight of each edge $f \in F$ is $w(f) = 1$, and so the (sum of) weighted crossings of f in a drawing of $G + f$ are naturally determined by the weights of the edges crossed by f.

With the help of the framework developed in the previous section, we can now give an (again informal) high-level refinement of our solution steps of r-MEI(G, F) as follows:

(IV) Consider a trinet T of G on the trinodes $V(F)$. If we fix a (realizable) T-sequence S, then we can use established tools, namely an adaptation of the idea of the funnel algorithm [9, 37], to efficiently compute a shortest alley among those having the same T-sequence S. For $uv \in F$, if we compute an alley α between u, v of length ℓ, then we can easily draw the inserted edge uv as an arc in α with ℓ weighted crossings in G. This will be handled precisely in Subsection 3.4.

(V) Suppose that, for $i = 1, 2$, α_i is a shortest alley between x_i and y_i having the T-sequence S_i. Then, as detailed later in Subsection 3.5, in Lemma 3.15 and Claim 3.18, we can decide from *only* S_1, S_2 whether there exist arcs from x_1 to y_1 in α_1 and from x_2 to y_2 in α_2, which do not cross (note that $\alpha_1 \cap \alpha_2$ may be nonempty and yet there may exist such a pair of non-crossing arcs). Moreover, if the two arcs cross then it should be only once.

(VI) Consequently, it will be enough to loop through all "suitable" T-sequences for every edge of F and independently perform the steps (IV), (V) for each combination of such T-sequences, in order to get an optimal solution of r-MEI(G, F) as in (III). The point is to bound the number of considered T-sequences in terms of only the parameter $k = |F|$. This will be done next, in Subsection 3.3.

3.3 T-sequences of potential shortest routes

Considering the outline (IV)–(VI), we first resolve the last point which is a purely mathematical question. In order to achieve the goal, we shall build a special trinet of G along shortest dual paths between the trinodes in G (Definition 3.4), and then we will be able to restrict our attention to special T-sequences (Definition 3.6) of bounded length. The latter is formulated in Lemma 3.8 whose proof presents the main piece of technical work in this paper.

Definition 3.4 (Shortest-spanning trinet)**.** Let (G', T) be a full trinet of a connected plane graph G, and let the weights w' in (G', T) be induced by weights w in G. For a triedge $q \in I(T)$, every internal vertex t of q is incident with two edges e, e' of G' of weight $w'(e) = w'(e')$ which we call the weight of t. The *transversing weight of* q equals the sum of the weights of the internal vertices of q.

A triedge $q \in I(T)$ between trinodes x, y is *locally-shortest* if the transversing weight of q is equal to the length of a shortest dual path π in $G' \cup T$ between x, y, such that π is contained in(!) the union of the two tricells incident to q (including the points of q itself). Similarly, q is *globally-shortest* if the transversing weight of q is equal to the dual distance between x, y in $G' \cup T$.

We say that T has the *shortest-spanning property* if every triedge in $I(T)$ is locally-shortest, and there exists a subset of triedges $J \subseteq I(T)$ forming a connected subgraph of T spanning all the trinodes such that every triedge in J is globally-shortest. ◇

The following observation is easy; it constitutes the initial part of Algorithm 1.

Claim 3.5. Let G be a connected plane graph and $N \subseteq V(G)$ be such that $|N| \geq 4$. Then there exists a trinet T of G with $N(T) = N$, such that the corresponding full trinet (G', T) has the shortest-spanning property, and (G', T) can be computed in linear time assuming constant $|N|$.

Proof. The claim is a consequence of the following straightforward procedure:

(a) Pick any trinode $x \in N$ and compute a shortest-dual-path tree in G from x to all other trinodes $N \setminus \{x\}$, using a linear-time shortest path algorithm. Shared sections of the computed dual paths from x can be naturally split into "parallel dual paths" in G as can be seen, e.g., in Figure 1. This produces a collection of globally-shortest triedges from x to all other trinodes $N \setminus \{x\}$.

(b) Choose any plane triangulation T_0 on the vertex set N such that $E(T_0)$ contains all edges from x to $N \setminus \{x\}$. For each edge $yz \in E(T_0 - x)$, greedily in any order, compute a shortest dual path in G from y to z not crossing any of the previously computed dual paths, which is done by calling a linear-time shortest path algorithm with suitably modified infinite weights. This step produces the remaining, locally-shortest triedges between the trinodes $N \setminus \{x\}$.

It is obvious that the procedure results in a trinet with the desired shortest-spanning property. Concerning runtime, we use $1 + 2(|N| - 1) - 3 = 2|N| - 4$ calls to a linear-time shortest path algorithm, and altogether spend time $\mathcal{O}(|N| \cdot |V(G)|)$ modifying G into G' of the trinet. This gives an easy upper bound of $\mathcal{O}(|N|^2 \cdot |V(G)|)$ for the total runtime. □

Definition 3.6 (Proper T-sequence)**.** Consider a trinet T and trinodes $u \neq v \in N(T)$. A *nonempty* sequence $S = (p_1, p_2, \ldots, p_m) \subseteq I(T)$ of triedges of T (repetition allowed) is a *proper T-sequence from u to v* if the following holds: u is disjoint from p_1 but there exists a tricell θ_0 incident to both u and p_1, v is disjoint from p_m but there exists a tricell θ_m incident to both v and p_m, and each two consecutive triedges p_i, p_{i+1} are distinct and incident to a common tricell θ_i for $1 \leq i < m$.

Finally, the *empty* sequence $S = \emptyset$ is considered a proper T-sequence from u to v if u, v are incident to a common tricell θ_0. ◇

Since T is a subdivision of a graph triangulation and u, v are nodes of this triangulation, we immediately obtain the following fact complementing Definition 3.6:

Claim 3.7. For every proper nonempty T-sequence S, the sequence of tricells $(\theta_0, \theta_1, \ldots, \theta_m)$ as in Definition 3.6 is uniquely determined by T and S. If $S = \emptyset$, then uv is a triedge of T and there are two choices of θ_0 incident to uv; we simply make an arbitrary deterministic choice of θ_0 among those two in each case. □

Now the main technical finding, necessary for our algorithm, comes in:

Lemma 3.8. *Consider an instance* r-MEI(G, F) *where G is a connected plane graph. Let (G', T) be a full trinet of G having the shortest-spanning property. There exists a set $\{\pi_f : f \in F\}$ where π_f for $f = uv$ is a route in $G' \cup T$ between the trinodes u, v, such that the following hold:*

a) There exists an optimal drawing D of $G + F$ with r-ins(G, F) *crossings such that each edge $f \in F$ is drawn in the alley of π_f, and no two edges of F cross each other more than once.*

b) The T-sequence S_f of each π_f is a proper T-sequence, and no triedge occurs in S_f more than $8k^4$ times where $k = |F|$.

Before giving the proof, we establish two simple technical claims which restrict how "complicated" drawings of the edges of F may appear in an optimal solution of a MEI(G, F) or r-MEI(G, F) instance. We also recall that although we allow edge-weighted graphs G in MEI instances, the weights of the edges of F are always 1. For any drawing D, let $\mathrm{cr}_D(X, Y)$ denote the number of crossings between edges of X and edges of Y in D, and let $\mathrm{cr}_D(X) := \mathrm{cr}_D(X, X)$.

Claim 3.9. In any optimal solution of an r-MEI(G, F) instance, any two edges of F cross at most once, and two edges of F have no crossing if they share a common end vertex. Moreover, if there exists a drawing D of $G + F$ such that $\mathrm{cr}_D(E(G), F) = c$, then r-ins$(G, F) \leq c + \binom{|F|}{2}$. The same holds also for (non-rigid) MEI(G, F) instances.

Proof. Let D be an optimal solution of r-MEI(G, F) or MEI(G, F). For the first part, we employ the folklore "arc exchange" argument from crossing numbers theory. If two edges $f, f' \in F$ are drawn in D such that they share two points x and y in common (one may be their end vertex), then we denote by b and b' the arcs of the drawings of f and f' (resp.) between x and y. Up to symmetry, we may assume that the interior of b carries not more crossings of D that the interior of b'. We redraw f' in the section between x, y closely along b (which is easily possible with polygonal arcs), and the new drawing D_1 saves at least one of the crossings between f and f'. This contradicts optimality of the starting solution D.

For the second part, we observe that since the edge weights in F are all 1, we have $\mathrm{cr}_D(F, F) \leq \binom{|F|}{2}$. □

Claim 3.10. For an instance r-MEI(G, F) and $f \in F$, assume that D_1 and D_2 are two drawings of $G + F$ such that $D_1 - f$ is identical to $D_2 - f$, and that $\mathrm{cr}_{D_1}(E(G), \{f\}) - \mathrm{cr}_{D_2}(E(G), \{f\}) > \binom{|F|}{2}$. Then $\mathrm{cr}(D_1) >$ r-ins(G, F), i.e., D_1 is not an optimal solution of r-MEI(G, F).

Proof. Let $E = E(G)$, $k = |F|$ and $F' = F \setminus \{f\}$. Using Claim 3.9, we estimate

$$\begin{aligned} \text{r-ins}(G,F) &\leq \operatorname{cr}_{D_2}(E,F) + \binom{k}{2} = \operatorname{cr}_{D_2}(E,F') + \operatorname{cr}_{D_2}(E,\{f\}) + \binom{k}{2} \\ &= [\operatorname{cr}_{D_1}(E,F') + \operatorname{cr}_{D_1}(E,\{f\})] - \operatorname{cr}_{D_1}(E,\{f\}) + \operatorname{cr}_{D_2}(E,\{f\}) + \binom{k}{2} \\ &< \operatorname{cr}_{D_1}(E,F) + 0 \leq \operatorname{cr}(D_1)\,. \end{aligned}$$

□

In the upcoming proof of Lemma 3.8, we shall use the following special terminology and notation. Let r-MEI(G,F) be an instance and (G',T) be a full trinet of G. For simplicity, we use the symbol f both for an edge $f \in F$ and for the arc representing f in a specific drawing of $G+F$ (more generally, of $(G'+F)\cup T$). We similarly consider a triedge $p \in I(T)$ also as the arc representing p in G'. If x, y are two points on any arc b, then let $b[x,y]$ denote the section of the arc from x to y.

Proof of Lemma 3.8. Consider the given shortest-spanning trinet and the corresponding plane embedded graph $G' \cup T$ with edge weights w' induced by the given weights w of G. We will implicitly assume that every arc a drawn in $G' \cup T$ avoids vertices and intersects $G' \cup T$ in finitely many points. In particular, the embedding and the arcs may be restricted to polygonal lines. For any arc b with ends u, v, we define the *T-sequence of b* from u to v as the sequence (with repetition) in which b intersects the triedges of T. We define the *transversing weight of b*, shortly *t-weight*, as the sum of the w'-weights of the edges of $G' \cup T$ crossed by b, and denote it by $t_{w'}(b)$.

We choose a drawing D among the optimal drawings of $G' + F$ such that D minimizes the combined length of the T-sequences of the edges of F, i.e., the number of crossings between F and the trinet T. Recall from Claim 3.9 that any two edges of F cross at most once, and they have no crossing if they share a common end vertex. For $f \in F$, let $S_f = (p_f^1, \ldots, p_f^{m_f})$ be the T-sequence of f and let $x_f^1, \ldots, x_f^{m_f}$, respectively, denote the points at which the arc of f intersects the triedges of T. The first task is to prove that each S_f is a proper T-sequence.

We prove a stronger technical claim: if, for some $f \in F$ and $j > i$, we have $p_f^i = p_f^j$ and the simple loop $a := p_f^i[x_f^i, x_f^j] \cup f[x_f^i, x_f^j]$ is contractible (i.e., with no trinode inside), then we get a contradiction to the choice of D above. Indeed, we may assume that f and $j > i$ are chosen such that a encloses minimal area in the drawing D. By the minimality of a, no triedge crosses the interior of $f[x_f^i, x_f^j]$ twice (all $p_f^{i+1}, \ldots, p_f^{j-1}$ are distinct). However, since the interior enclosed by a contains no trinode, the previous implies that no triedge other than p_f^i may intersect a, and so $j = i+1$. Consequently, since the triedge p_f^i is locally shortest in T, the t-weights of the considered section satisfy $t_{w'}\big(p_f^i[x_f^i, x_f^j]\big) \leq t_{w'}\big(f[x_f^i, x_f^j]\big)$. If we re-route f closely along $p_f^i[x_f^i, x_f^j]$ (without crossing p_f^i), then this change does not increase the crossing number by the inequality of t-weights, but the T-sequence of f gets shorter (see in Figure 3). Hence, it contradicts our choice of D.

We return to S_f being a proper T-sequence. If S_f is empty, then the statement is trivial. If S_f contained a consecutive repeated triedge $p_f^i = p_f^{i+1}$ for some $1 \leq i < m_f$, then the loop a from the previous paragraph is always contractible, and so we would get the claimed contradiction to our choice of D. Assume that $f = uv$ and the triedge p_f^1 would be incident to the starting trinode u. Then we again have the contractible loop $a := p_f^1[u, x_f^1] \cup f[u, x_f^1]$, and we would again obtain the same contradiction to our choice of D. The remaining properties of proper T-sequences follow trivially.

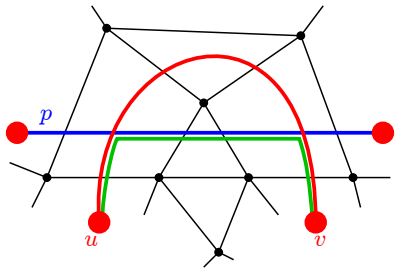


Figure 3: Two consecutive crossings of the arc of $f = uv$ with a triedge p (which is locally-shortest) determine a contractible loop, and so f can be re-routed partly along p without inducing more crossings with G or with other edges of F.

The last and most difficult step is to prove that no triedge repeats in S_f too many times, for each $f \in F$. If $p_f^i = p_f^j = p'$ for $i \neq j$, then the simple loop $a' := p'[x_f^i, x_f^j] \cup f[x_f^i, x_f^j]$ is non-contractible, so it separates some pair of trinodes of T. Since G' is connected, this implies that $t_{w'}(a') \geq 1$. Since at most $2k-1$ globally-shortest triedges of T span all the trinodes by Definition 3.4, a' (and consequently $f[x_f^i, x_f^j]$) must cross at least one of them. Therefore, if a triedge p' repeats in S_f at least $8k^4$ times, then there is a globally-shortest triedge p of T such that p repeats in S_f at least $\ell \geq 8k^4/(2k-1) > 4k^3$ times.

Let $Y = (y_1, \ldots, y_\ell)$ (a subsequence of $(x_f^1, \ldots, x_f^{m_f})$) be the ordered sequence of points in which the arc of f intersects the arc of the triedge p. We say that an index $i \in \{2, \ldots, \ell-1\}$ is a *switchback* of Y if y_{i-1}, y_{i+1} both lie on the same side of y_i on p (see Figure 4 for an example). Up to symmetry, let the points on p be ordered such that y_{i+1} lies between y_{i-1}, y_i. Since p is globally-shortest in T, we get (now regardless of contractibility of the induced loops)

$$t_{w'}\big(p[y_{i-1}, y_i]\big) \leq t_{w'}\big(f[y_{i-1}, y_i]\big),$$

and then

$$\begin{aligned} t_{w'}\big(f[y_{i-1}, y_{i+1}]\big) &\geq t_{w'}\big(p[y_{i-1}, y_i]\big) + t_{w'}\big(f[y_i, y_{i+1}]\big) \\ &= t_{w'}\big(p[y_{i-1}, y_{i+1}]\big) + t_{w'}\big(p[y_{i+1}, y_i]\big) + t_{w'}\big(f[y_i, y_{i+1}]\big) \\ &\geq t_{w'}\big(p[y_{i-1}, y_{i+1}]\big) + 1. \end{aligned}$$

Hence, if we locally re-route f along $p[y_{i-1}, y_{i+1}]$, then we save the amount of at least 1 in the crossings of f with $E(G)$. Note that this is not a contradiction to our choice of optimal drawing D yet since the change may introduce many new crossings of f with the rest of F. However, we cannot have more than $\binom{k}{2}$ switchbacks in Y or we get a contradiction using Claim 3.10.

Since $\ell \geq 4k^3$, there is a consecutive subsequence $Y' \subseteq Y$ of length $\ell' \geq \ell/\binom{k}{2} - 1 > 8k$ without switchbacks. Without loss of generality, we assume $Y' = (y_1, \ldots, y_{\ell'})$. Let $g_i := f[y_i, y_{i+1}]$ and $g_i^\circ := g_i \cup p[y_i, y_{i+1}]$, for $i \in \{1, \ldots, \ell'-1\}$ (see Figure 5). As argued before, each g_i° is a simple loop separating some pair of trinodes of T. Since no two edges of F cross more than once, there are at most $k-1$ indices $i \in \{1, \ldots, \ell'-1\}$ such that g_i is crossed by another edge(s) of F.

Let x, y be the ends of the triedge p. Assume that we have $i \neq j \in \{1, \ldots, \ell'-1\}$ such that neither of g_i°, g_j° separates x from y. Let $Z_i \neq \emptyset$ denote the set of trinodes of T that are separated by g_i° from x, y, and let Z_j be defined analogously. We claim that $Z_i \cap Z_j = \emptyset$. If not, then—up to symmetry—g_j° is separated from x, y by g_i°, except a possibly shared section of $p[y_i, y_{i+1}]$.

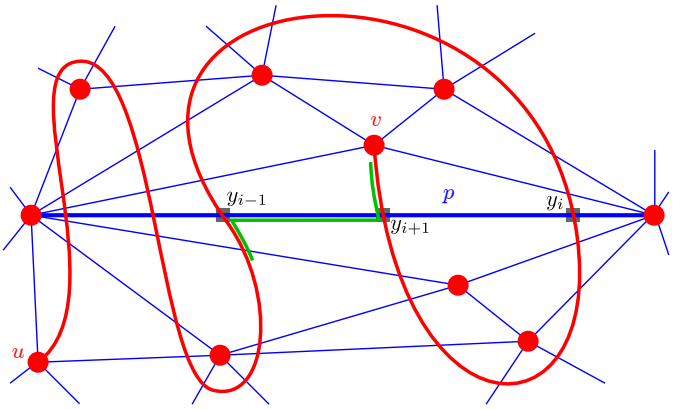


Figure 4: Example of a switchback. The blue edges are the trinet. The red arc $f = uv$ crosses the thick triedge p multiple times, with a switchback at point y_i. The green line depicts a local rerouting for f.

The former is impossible by the Jordan curve theorem and the latter would mean that there is a switchback between i and j, which is again a contradiction. Since there are at most $2k-2$ pairwise disjoint nonempty possibilities (e.g., singleton trinodes other than x, y) for the sets Z_i, Z_j, there are at most $2k-2$ indices $i \in \{1, \ldots, \ell'-1\}$ such that g_i° does not separate x from y.

Since $\ell' \geq 8k$, there exists a set of indices $J \subseteq \{1, \ldots, \ell'-2\}$, $|J| \geq \ell' - 2(k-1+2k-2) - 2 > 2k$, such that for every $j \in J$ both the arcs g_j, g_{j+1} are not crossed by other edges of F and both g_j°, g_{j+1}° separate x from y. Let $f_0 := f[y_1, y_{\ell'}]$ and $p_0 := p[y_1, y_{\ell'}]$; we get $Y' \subseteq p_0$ since there are no switchbacks in Y'. Observe also that $g_j^\circ \cap g_{j+1}^\circ = \{y_{j+1}\}$ since f is not self-intersecting and there is no switchback in Y'. Hence, up to symmetry, g_j° separates x from g_{j+1}°, and g_{j+1}° separates g_j° from y. It easily follows that $g_j^\circ \cup g_{j+1}^\circ$ forms the boundary of an arc-connected region of $\mathbb{R}^2 \setminus (f_0 \cup p_0)$ (a *face of* $f_0 \cup p_0$). Since at most $2k-2$ of the faces of $f_0 \cup p_0$ may contain a trinode of T other than x, y, there exists $j \in J$ such that, in addition to the above properties of J, the face σ bounded by $g_j^\circ \cup g_{j+1}^\circ$ contains no trinode (see in Figure 6).

Our goal now is to re-route f along $p[y_j, y_{j+1}]$ (i.e., "replacing" the part $g_j \subset f$). Again, since p is globally-shortest in T, this move does not increase the number of crossings of f with $E(G)$, and the T-sequence of f gets shorter. It remains to argue that we can avoid new crossings of f with $F \setminus \{f\}$. If any $f' \in F$ crosses $p[y_j, y_{j+1}]$ then, since σ contains no trinode, f' has to leave σ as well, and the only possibility is across $p[y_{j+1}, y_{j+2}]$ by the previous assumptions. Consequently, such f' can be re-routed along $p[y_j, y_{j+2}]$, similarly to f, and no crossing with f is required (see again in Figure 6). Note, moreover, that even if two such edges $f', f'' \in F$ cross each other in σ, there is no problem and they will cross in their new routing in the same way. We have again reached a contradiction to our choice of D. □

3.4 Shortest routes in a sleeve

Next, we consider point (IV) of the outline in Subsection 3.2. To recapitulate, for trinodes u, v of a trinet T of G and a given proper T-sequence S from u to v, the task is to find a shortest route from u to v among those having the same T-sequence S (and independently of other routes considered in the problem). Since we cannot, in general, avoid repeating triedges in S and tricells

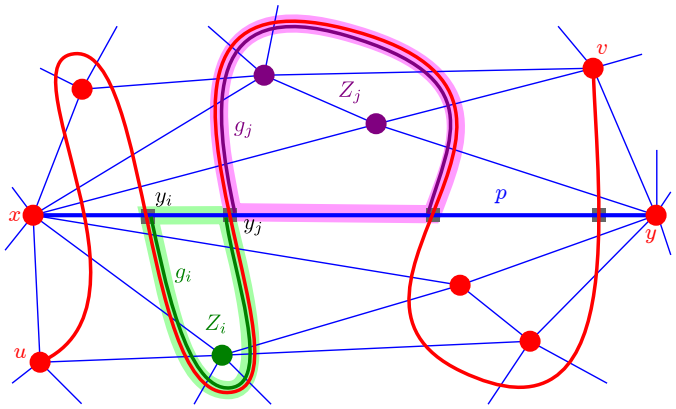


Figure 5: The blue edges are the trinet and the thick triedge p (which connects x and y) is crossed by the red arc $f = uv$ multiple times without switchbacks. The segment g_i of f between a crossing at y_i and its succeeding crossing is depicted in dark green, and gives rise to the loop g_i° in light green, which separates the (green) trinodes Z_i from x, y. Similarly, we have g_j, g_j° and Z_j in violet, and argue that $Z_i \cap Z_j = \emptyset$.

in the sequence $(\theta_0, \theta_1, \ldots, \theta_m)$ in Definition 3.6, we use a similar workaround as in [28]; "lifting" the respective sequence of tricells into a universal cover as follows.

Definition 3.11 (Sleeve of a T-sequence)**.** Let (G', T) be a full trinet of a plane graph G, and consider a proper T-sequence $S = (p_1, p_2, \ldots, p_m)$ from u to v determining the sequence of tricells $(\theta_0, \theta_1, \ldots, \theta_m)$ by Claim 3.7. For $i = 0, 1, \ldots, m$, let L_i be a disjoint copy of the embedded subgraph of $G' \cup T$ induced by θ_i. Construct a plane graph L from the union $L_0 \cup \cdots \cup L_m$ by identifying, for $j = 1, \ldots, m$, the copy of the triedge p_j in L_{j-1} with the copy of p_j in L_j. We call L the *sleeve of* S in the trinet (G', T), and we identify u and v with their copies in L_0 and L_m, respectively. We make the unique face of L that is not covered by a copy of any tricell of T the *outer face* of L. ◇

Observe that every route from u to v in $G' \cup T$ having its T-sequence equal to S can be easily lifted into a corresponding u–v route in the sleeve L of S. Conversely, any u–v route in L avoiding the outer face and crossing the copies of triedges in L at most once each, can be obviously projected down to $G' \cup T$ to make a route with the T-sequence equal to S. In fact, we can routinely prove that some shortest u–v route in L must be of the latter kind, under the shortest-spanning property (cf. Definition 3.4).

Lemma 3.12. *Let (G', T) be a shortest-spanning full trinet of an edge-weighted plane graph G, S a proper T-sequence between trinodes u, v of T, and let L be the sleeve of S. Let ℓ be the length of a shortest route from u to v among those having the T-sequence S. Then, ℓ is equal to the dual distance from u to v in L without the outer face, and at least one of the u–v routes of length ℓ in L crosses the copy of each triedge from S in L exactly once.*

Proof. Let π be any shortest route from u to v in $G' \cup T$ with the given T-sequence S. The copies of the faces and dual edges of π lifted into the sleeve L give a route π_L from u to v which avoids the outer face of L. Obviously, the length of π_L equals the length of π.

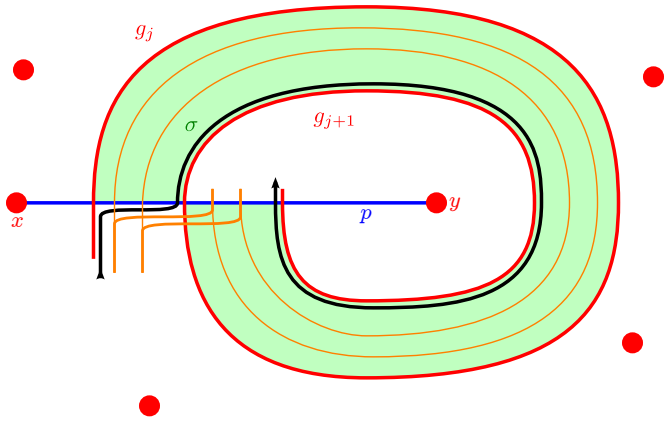


Figure 6: The face σ bounded by $g_j^\circ \cup g_{j+1}^\circ$ ($\subseteq f \cup p$) depicted in green: since there is no trinode in this face and neither g_j, g_{j+1} are crossed by other edges of F, it is possible to re-route f partly along p such that it avoids g_j. Other possible F-edges entering the green face through a section of p must leave at the other end, and hence can be re-routed similarly to f.

Conversely, we aim to show that some shortest u–v route crosses the copy of each triedge of S in L exactly once. Assume a shortest route π_1 of length ℓ from u to v in L without the outer face. Recall from Definition 3.11 that $L = L_0 \cup \cdots \cup L_m$. For $S = (p_1, p_2, \ldots, p_m)$, let $(p'_1, p'_2, \ldots, p'_m)$ be the sequence of corresponding copies of the triedges of S in L, and let $p'_0 := u$, $p'_{m+1} := v$. Note that each p'_i, $i \in \{1, \ldots, m\}$, connects two vertices of the outer face of L, and so p'_i separates p'_{i-1} from p'_{i+1}. In particular, every u–v route in L which avoids the outer face must cross each of $p'_1, \ldots, p'_m$.

Let i be the maximum index such that p'_i is crossed by π_1 more than once. Then there is a subpath $\sigma_1 \subseteq \pi_1$ stretching between two consecutive crossings of π_1 with p'_i and contained in L_i. We turn π_1 into π_2 by re-routing the subpath σ_1 along the boundary p'_i in L_{i-1}. Since p_i is locally-shortest in the trinet T, the length of π_2 equals the length of π_1. By induction on the number of excess crossings of π_2 with copies of the triedges, we can then get a u–v route π_0 of length ℓ such that π_0 crosses the copy of each triedge from S in L exactly once.

Finally, the route π_0 projects down to a route of length ℓ from u to v in $G' \cup T$ having the T-sequence S. □

Corollary 3.13. *Let (G', T) be a shortest-spanning full trinet of a plane graph G with integer edge weights, and S a proper T-sequence between trinodes u, v of T. A shortest u–v route among those having the T-sequence S can be found in $\mathcal{O}(|S| \cdot |N(T)| \cdot |V(G)|)$ time.*

Observe that in our case, by Lemma 3.8, we have $|S| \cdot |N(T)| \leq (6k \cdot 8k^4) \cdot 2k = 96k^6$.

Proof. By Definition 3.1, the size of the full trinet (G', T) is $\mathcal{O}(|N(T)| \cdot |V(G)|)$, and the sleeve is composed of $|S|$ copies of subgraphs of $G' \cup T$, and so $\mathcal{O}(|S| \cdot |N(T)| \cdot |V(G)|)$ bounds the size of the sleeve L. Therefore, any linear time shortest path algorithm applicable in our situation, such as the algorithm of Klein et al. [35] since L is planar, or Thorup's algorithm [41] since we have integral weights, suffices. □

We also remark that, as we are going to use Corollary 3.13 in the subsequent proofs, we are fine with edge weights of G given in unary. More precisely, every unit of finite weight on the edges of

G could be counted towards some edge of the full input graph from Section 4 without repetition, and so a simple adaptation of BFS (treating infinite weights separately) in Corollary 3.13 suffices for our purposes.

3.5 Crossing of routes

Finally, it remains to address point (V). Consider a 4-tuple of distinct trinodes u, v, u', v'. Let π be a u–v route and π' be a u'–v' route. We say that an *arc b follows the route π* if b is contained in the alley of π and b intersects the faces forming the alley exactly in the order given by π (recall that a route is technically a dual walk and hence, possibly, some face might repeat in π). We say that the pair of routes π, π' is *non-crossing*, if there exist a u–v arc b following π and a u'–v' arc b' following π' such that $b \cap b' = \emptyset$. In order to characterize possible non-crossing pairs of routes in terms of their T-sequences, we introduce the notion of a crossing certificate (see also Figure 7):

Definition 3.14 (Crossing certificate)**.** Let (G', T) be a full trinet of a plane graph G, and let π be a route from u to v and π' be a route from u' to v' in $G' \cup T$, where u, v, u', v' are distinct trinodes of T. Assume the T-sequences $S = (p_1, \ldots, p_n)$ of π and $S' = (p'_1, \ldots, p'_\ell)$ of π' are proper and let $(\theta_0, \ldots, \theta_n)$ and $(\theta'_0, \ldots, \theta'_\ell)$ be their tricell sequences (unique by Claim 3.7). For technical reasons, let $p_0 := u$, $p_{n+1} := v$ and $p'_0 := u'$, $p'_{\ell+1} := v'$.

A *crossing certificate for S, S'* is a triple of indices (c, d, m) where $c, d, m \geq 0$, $c + m \leq n$, $d + m \leq \ell$, such that the following holds:

a) $\theta_{c+j} = \theta'_{d+j}$ for $0 \leq j \leq m$, but $p_c \neq p'_d$ and $p_{c+m+1} \neq p'_{d+m+1}$,

b) the triple p_c, p'_d, p_{c+1} occurs around the tricell $\theta_c = \theta'_d$ in the same cyclic orientation as the triple $p_{c+m+1}, p'_{d+m+1}, p_{c+m}$ occurs around $\theta_{c+m} = \theta'_{d+m}$.

Referring to point a), we call the tricells $\theta_{c+j} = \theta'_{d+j}$ for $0 \leq j \leq m$ the *central tricells of the crossing certificate* (c, d, m).

Furthermore, a crossing certificate for the same sequence S and the reversal of S' from v' to u' is also called a crossing certificate for S, S'. ◇

Definition 3.14 deserves a closer explanation. Assume that a crossing certificate satisfies $0 < c < n$ and $0 < d < \ell$. Then all four elements $p_c, p'_d, p_{c+1}, p'_{d+1}$ are triedges of the same tricell $\theta_c = \theta'_d$, and since $p_{c+1} \neq p_c \neq p'_d \neq p'_{d+1}$, we get $p_{c+1} = p'_{d+1}$. Hence $m > 0$ and the situation is such that S and S' "merge" at θ_c where (up to symmetry) S comes on the right of S', and they again "split" at θ_{c+m} where S leaves on the left of S', thereby "crossing it". This is the case illustrated in Figure 7. The full definition, though, covers also the boundary cases of crossing certificates for which $c \in \{0, n\}$ or $d \in \{0, \ell\}$ (or both), and when S and S' may have no triedge in common; those can be easily examined case by case.

Lemma 3.15. *Let (G', T) be a full trinet of an edge-weighted plane graph G, and u_i, v_i, $i = 1, 2$, be four distinct trinodes. Assume that S_i from u_i to v_i are proper T-sequences. In $G' \cup T$, for $i = 1, 2$, there exist routes π_i from u_i to v_i having the the T-sequence S_i, such that π_1, π_2* are non-crossing*, if and only if there exists no crossing certificate for S_1, S_2.*

Proof. Let L_i, $i = 1, 2$, be the sleeves of S_i in (G', T). Assume that (c, d, m) is a crossing certificate for S_1, S_2, and let $R = (\theta_c, \ldots, \theta_{c+m})$ be the sequence of the central tricells of this certificate. Let $K_i \subseteq L_i$, $i = 1, 2$, be the plane subgraphs consisting of the copies of the tricells from R in the sleeve L_i. Note that R may repeat the same tricell several times, but in L_i we have got independent

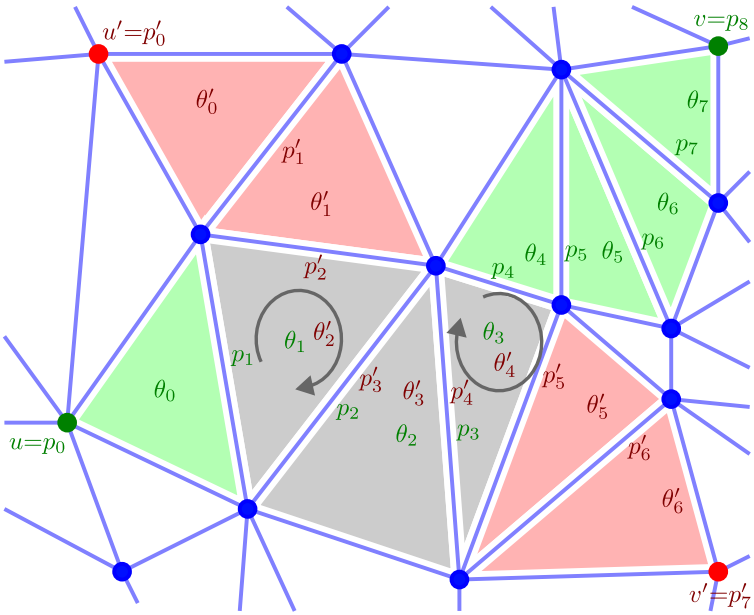


Figure 7: An example illustrating a crossing certificate (see Definition 3.14): The blue graph depicts a trinet T. In green, we depict the route π (with $n = 7$) from $u = p_0$ to $v = p_8$ with T-sequence $S = (p_1, ..., p_7)$ and tricell sequence $(\theta_0, ..., \theta_7)$. In red, we depict the route π' (with $\ell = 7$) from $u' = p'_0$ to $v' = p_7$ with T-sequence $S' = (p'_1, ..., p'_6)$ and tricell sequence $(\theta'_0, ..., \theta'_6)$. Both routes overlap at three tricells $\theta_1 = \theta'_2$, $\theta_2 = \theta'_3$, and $\theta_3 = \theta'_4$ (depicted in gray) in such a way that a crossing between the two routes is forced. We thus have a crossing certificate (c, d, m) with $c = 1$, $d = 2$, $m = 3$: both the triple $p_1, p'_2, p_2 = p'_3$ (in $\theta_1 = \theta'_2$) and the triple $p_4, p'_5, p_3 = p'_4$ (in $\theta_3 = \theta'_4$) appear in the same orientation, namely, clockwise.

copies of the possibly repeated tricells. We may also assume that $K_1 = K_2$ since they are both made of copies of the same sequence R of tricells.

In the above view, Definition 3.14 says that (c, d, m) is a crossing certificate iff the elements $p_c, p'_d, p_{c+m+1}, p'_{d+m+1}$ appear on the outer face of $K_1 = K_2$ in this cyclic order. Hence, by Jordan's curve theorem, if there is a crossing certificate for S_1, S_2, then π_1, π_2 cannot be non-crossing.

Conversely, we show how to build non-crossing π_1, π_2 if there is no crossing certificate for S_1, S_2. For each tricell θ of T, bounded by triedges q_1, q_2, q_3, we choose three arbitrary edges e_j from q_j, $j = 1, 2, 3$, and three arbitrary internally disjoint dual paths $\sigma_{1,2}, \sigma_{1,3}, \sigma_{2,3}$ contained in θ such that $\sigma_{i,j}$ is a dual path in $G' \cup T$ connecting the face incident to e_i to the face incident to e_j. Furthermore, we denote by x_j the trinode of θ opposite to q_j and we choose another three arbitrary dual paths ρ_1, ρ_2, ρ_3 contained in θ such that ρ_j is a dual path in $G' \cup T$ connecting a face incident to x_j to the face incident to e_j. We call chosen $\sigma_{1,2}, \sigma_{1,3}, \sigma_{2,3}, \rho_1, \rho_2, \rho_3$ the representative dual paths of the tricell θ.

For the proper T-sequence S_i, $i = 1, 2$, we simply compose the route π_i of the appropriate representative dual paths of the tricells determined by S_i. It is routine to verify that these π_1, π_2 are non-crossing, if and only if there exists no crossing certificate for S_1, S_2. □

In: a plane graph G, edge weights $w\colon E(G) \to \mathbb{N}_+ \cup \{\infty\}$, an edge set F to insert, s.t. $w(f) = 1$ for $f \in F$.

Out: an optimal solution to (w-weighted) r-MEI(G, F).

(1) Compute a full trinet (G', T), $N(T) := V(F)$,with the shortest-spanning property of T, as given by Claim 3.5.

(2) For each $f = uv \in F$:

  (a) Compute $\mathcal{S}_f$ as the set of all its possible and relevant proper T-sequences from u to v. The size of $\mathcal{S}_f$ is bounded due to Lemma 3.8(b) by $2^{s(|F|)}$ where $s(k) = \mathcal{O}(k^5 \log k)$.

  (b) For each $S \in \mathcal{S}_f$, compute a shortest u–v route π_S in $G' \cup T$ among those having the T-sequence S (where the length function is induced by w), using Corollary 3.13.

(3) For each possible system of representatives $\mathcal{P} = \{S_f\}_{f\in F}$ with $S_f \in \mathcal{S}_f$:

  (a) Check, for each pair $f, f' \in F$, whether there exists a crossing certificate for $S_f, S_{f'}$ (e.g., using brute force by Def. 3.14).
  Let $X_\mathcal{P}$ be the set of pairs $\{f, f'\}$ for which such a certificate has been found.

  (b) If any pair $\{f, f'\} \in X_\mathcal{P}$ requires more than a single crossing (which can be found by checking again for two "independent" crossing certificates of $S_f, S_{f'}$), let $\mathrm{cr}_\mathcal{P} := \infty$.

  (c) Otherwise, let $\mathrm{cr}_\mathcal{P} := |X_\mathcal{P}| + \sum_{f\in F} \mathrm{len}_w(\pi_{S_f})$, where π_{S_f} is the shortest route for f and S_f computed in step (2) and $\mathrm{len}_w(\pi_{S_f})$ is the length.

(4) Among all $\mathcal{P}$ considered in (3), pick one with smallest $\mathrm{cr}_\mathcal{P} < \infty$. Let this be $\mathcal{P}^\circ = \{S^\circ_f\}_{f\in F}$.

(5) In the plane graph G, realize each edge $f \in F$ following its respective route $\pi_{S^\circ_f}$, such that the overall resulting weighted number of crossings is $\mathrm{cr}_{\mathcal{P}^\circ}$:

  (a) By the minimality setup in (4), no $\pi_{S^\circ_f}$ is self-intersecting.

  (b) A standard postprocessing argument—removing consecutive crossings between any pair f, f' (within a section of $S^\circ_f \cap S^\circ_{f'}$) by re-routing f' partially along f—prevents multiple crossings in the pairs from $X_\mathcal{P}$ and makes the pairs of F not in $X_\mathcal{P}$ crossing-free.

Algorithm 1: Algorithm to solve the (weighted) Rigid MEI problem.

3.6 Summary of the r-MEI algorithm

We can now summarize the overall algorithm to solve Rigid MEI, see Algorithm 1. Based thereon, together with Lemmas 3.8, 3.12, Corollary 3.13, and Lemma 3.15 we obtain:

Theorem 3.16. *Let G be a connected plane graph with edge weights $w\colon E(G) \to \mathbb{N}_+ \cup \{\infty\}$, and F a set of $k \geq 1$ edges to insert, such that $w(f) = 1$ for all $f \in F$. Algorithm 1 finds an optimal solution to the w-weighted* r-MEI(G, F) *problem, or determines that there is no finite solution, in time $\mathcal{O}\big(2^{p(k)} \cdot |V(G)|\big)$, where $p(k)$ is a polynomial function in k. In fact, $p(k) = \mathcal{O}(k^6 \log k)$.*

Before giving the proof, we need to develop a deeper understanding of the concept of non-crossing routes and crossing certificates, and a detailed specification of step (3b) of Algorithm 1. By adapting the arguments of Lemma 3.15, one can actually get the following slight strengthening:

Claim 3.17. Let (G', T) be a full trinet of an edge-weighted plane graph G, and u_i, v_i, $i = 1, 2$, be four distinct trinodes. Let π_i, $i = 1, 2$, be a u_i–v_i route in $G' \cup T$. If there exist simple u_i–v_i arcs b_i, $i = 1, 2$, following the route π_i, such that b_1 intersects b_2 in exactly one point x and they properly cross in x, then there exists a crossing certificate for the T-sequences of π_1 and π_2. □

We say that there exist *two independent crossing certificates* for the T-sequences S, S' if there are crossing certificates (c, d, m) and (c', d', m') for S, S' (each one up to possible reversal of S' as in Definition 3.14), such that the set of central tricells of (c, d, m) is disjoint from the set of central tricells of (c', d', m'). The following can then be straightforwardly obtained from Lemma 3.15:

Claim 3.18. Let (G', T) be a full trinet of an edge-weighted plane graph G, and u_i, v_i, $i = 1, 2$, be four distinct trinodes. Assume that S_i from u_i to v_i are proper T-sequences. In $G' \cup T$, there exist simple arcs b_i from u_i to v_i such that, for $i = 1, 2$,

- b_i is contained in the alley of a u_i–v_i route having the T-sequence S_i, and
- b_1 intersects b_2 in at most one point,

if and only if there exists no two independent crossing certificates for S_1, S_2. □

The implementation of step (3b), using Claim 3.18, hence simply checks by brute force for the existence of two independent crossing certificates for $S_f, S_{f'}$.

Proof of Theorem 3.16. In this proof, we will use some of the terminology and notation from the proof of Lemma 3.8, and refer to the notation of Algorithm 1.

Consider arbitrary $\mathcal{P} = \{S_f\}_{f \in F}$ as in step (3). The value of $\mathrm{cr}_{\mathcal{P}}$ computed in step (3c), provided that $\mathrm{cr}_{\mathcal{P}^\circ} < \infty$, is a lower bound on the number of crossings of any feasible solution of r-MEI(G, F) such that, for each $f \in F$, the T-sequence of the arc f is exactly S_f. This fact follows directly from Lemma 3.15 (for the part $|X_{\mathcal{P}}|$) and from Lemma 3.12 (for the part $\sum_{f \in F} \mathrm{len}_w(\pi_{S_f})$).

By Lemma 3.8, there is an optimal feasible solution D to r-MEI(G, F) such that, for every $f \in F$, the arc of f in D has its T-sequence (with respect to the trinet T from step (1)) equal to some proper $S_f \in \mathcal{S}_f$ as computed in (2), and step (3b) does not apply to these values by Claim 3.18. Consequently, $\mathrm{cr}_{\mathcal{P}^\circ} \leq$ r-ins(G, F) by the lower-bound argument from the previous paragraph. Hence if we can prove that step (5) can compute a drawing D° of $G + F$ with $\mathrm{cr}_{\mathcal{P}^\circ}$ weighted crossings, provided $\mathrm{cr}_{\mathcal{P}^\circ} < \infty$, then we complete the proof of Theorem 3.16.

For $f \in F$, let b_f denote a realization of the edge f as an arc following the route of $\pi_{S^\circ_f}$ (such that the T-sequence of b_f is S°_f), before the postprocessing step (5b). By the minimality choice in step (4), we can be sure that b_f does not cross itself: the self-crossing would induce a non-contractible loop with at least one crossing over G', but then there exists a T-sequence $S'_f \subset S^\circ_f$—the sequence S°_f without the triedges forcing the loop. Replacing these two sequences in $\mathcal{P}^\circ$ results in a smaller number of crossings on f while not increasing the number of crossings at any term in the summation considered in step (3c).

Let D_1 denote the drawing of $G+F$ made of G and $\{b_f : f \in F\}$. Observe that $\mathrm{cr}_{D_1}(E(G), F) = \sum_{f \in F} \mathrm{len}_w(\pi_{S^\circ_f})$.

Fix some $f, f' \in F$, $f = uv$ and $f' = u'v'$, such that $b_f, b_{f'}$ cross each other (properly) more than once. Consider the point set $p := b_f \cup b_{f'}$ with the outer face incident to, say, the trinode u. If any of the bounded faces of p contained a trinode (which cannot be u) of T, then we could "split" the T-sequences S°_f and $S^\circ_{f'}$ into S^1_f, S^2_f and $S^1_{f'}, S^2_{f'}$ each, such that for each of the pairs $S^1_f, S^1_{f'}$ and $S^2_f, S^2_{f'}$ there would exist a crossing certificate by Claim 3.17. This would, in turn, provide two independent crossing certificates for the T-sequences $S^\circ_f, S^\circ_{f'} \in \mathcal{P}^\circ$ and by the step (3b), it would contradict $\mathrm{cr}_{\mathcal{P}^\circ} < \infty$.

Consequently, all the bounded faces of $p = b_f \cup b_{f'}$ are free of trinodes of T. This, in particular, means that if we construct b^1_f from b_f by re-routing it along a section of $b_{f'}$ between two consecutive shared points of $b_f \cap b_{f'}$, then the T-sequence of b^1_f would again be S°_f. Moreover, since $b_f, b_{f'}$ have

been chosen from their respective shortest routes, the t-weight of b_f^1 would be equal to $\mathrm{len}_w(\pi_{S_f^\circ})$ (which is the t-weight of the original b_f). Iterating the process, we arrive at a drawing D_2 of $G+F$ satisfying the following:

i. no two edges of F in D_2 cross more than once,

ii. $\mathrm{cr}_{D_2}(E(G),F) \le \mathrm{cr}_{D_1}(E(G),F)$.

What remains is to observe that two edges $f, f' \in F$ properly cross each other in D_2 only if $\{f,f'\} \in X_{\mathcal{P}^\circ}$. Indeed, if f, f' properly cross in D_2, then the crossing of f, f' is the only one and there exists a crossing certificate for $\pi_{S_f^\circ}, \pi_{S_{f'}^\circ}$ by Claim 3.17, and then $\{f,f'\} \in X_{\mathcal{P}^\circ}$ due to step (3a).

To summarize, we get

$$\begin{aligned} \mathrm{cr}_{\mathcal{P}^\circ} &\le \text{r-ins}(G,F) \le \mathrm{cr}(D_2) = \mathrm{cr}_{D_2}(E(G),F) + \mathrm{cr}_{D_2}(F,F) \\ &\le \mathrm{cr}_{D_1}(E(G),F) + |X_{\mathcal{P}^\circ}| = \mathrm{cr}_{\mathcal{P}^\circ} \end{aligned}$$

which proves optimality of the solution computed by Algorithm 1.

Finally, we discuss the runtime bound of Algorithm 1. Let $k = |F|$. Step (1) is performed in time $\mathcal{O}(k^2 \cdot |V(G)|)$ by Claim 3.5. By Lemma 3.8, $|\mathcal{S}_f| \le 2^{s(k)}$ where $s(k) \le \log\left((6k)^{48k^5}\right) = 48k^5 \log 6k$. Step (2) hence takes time $\mathcal{O}(k \cdot 2^{s(k)} \cdot k^6 \cdot |V(G)|) < \mathcal{O}(2^{s(k)\cdot k} \cdot |V(G)|)$ by Corollary 3.13. Step (3) is iterated $\mathcal{O}(2^{s(k)\cdot k})$ times, and each iteration takes time polynomial in k (independently of G) even by brute force. Step (4) takes only time $\mathcal{O}(2^{s(k)\cdot k})$. Finally, step (5) performs k computations in $\mathcal{O}(|V(G)|)$ time, to realize each $f \in F$ in G, and then a number of concurrent re-routings which can be bounded by an amortized analysis: each of the k routes is of length $\mathcal{O}(|V(G)|)$ and each element of it could be re-routed at most once towards each of the $k-1$ remaining routes, summing to $\mathcal{O}(k^2 \cdot |V(G)|)$.

The above analysis sums up to overall $\mathcal{O}\big(2^{s(k)\cdot k} \cdot |V(G)|\big) = \mathcal{O}\big(2^{p(k)} \cdot |V(G)|\big)$ time where $p(k) = \mathcal{O}(k^6 \log k)$. □

4 General MEI

Now, we turn our attention to the general MEI(G,F) problem, in which the embedding of the planar graph G is not pre-specified. Recall that triconnected planar graphs have a unique embedding (up to mirroring), but already biconnected graphs can have an exponential number of embeddings in general. As it is commonly done in insertion problems since [26], we will use the *SPR-tree* datastructure (sometimes also known as SPQR-tree) to encode and work with all these possible embeddings. The structure was first defined in a slightly different form in [2], based on prior work of [5, 42]. It can be constructed in linear time [25, 32] and only requires linear space.

Definition 4.1 (SPR-tree, based on [10])**.** Let G be a biconnected graph with at least three vertices. The *SPR-tree* $\mathcal{T}$ of G is the unique smallest tree satisfying the following properties:

a) Each node ν in $\mathcal{T}$ holds a specific graph $S_\nu = (V_\nu, E_\nu)$, with $V_\nu \subseteq V(G)$, called a *skeleton*. Each edge e of E_ν is either a *real* edge $e \in E(G)$, or a *virtual* edge $e = xy \notin E(G)$ (while still, $x, y \in V(G)$).

b) $\mathcal{T}$ has three different node types with the following skeleton structures: **(S)** S_ν is a simple cycle; **(P)** S_ν consists of two vertices and at least three multiple edges between them; **(R)** S_ν is a simple triconnected graph on at least four vertices.

c) For every edge $\nu\mu$ in $\mathcal{T}$ we have $|V_\nu \cap V_\mu| = 2$. These two common vertices, say x, y, form a vertex 2-cut in G. The skeleton S_ν contains a specific virtual edge $e_\mu \in E(S_\nu)$ that represents the node μ and, symmetrically, some specific $e_\nu \in E(S_\mu)$ represents ν. Both e_ν, e_μ have the ends x, y, and each one of e_ν, e_μ is called the *twin* of the other.

d) For an edge $\nu\mu \in E(\mathcal{T})$, let $e_\mu \in E_\nu$, $e_\nu \in E_\mu$ be the pair of virtual edges as in (c) connecting the same x, y. The operation of *merging* at $\nu\mu$ creates a graph $(S_\nu \cup S_\mu) - \{e_\mu, e_\nu\}$ obtained by gluing the two skeletons S_ν, S_μ at x, y and removing e_μ, e_ν.

Consider the tree $\mathcal{T}$ rooted at any node. For $\nu \in V(\mathcal{T})$ we define the *pertinent graph* P_ν of ν by recursively merging the skeletons at every tree-edge of the subtree rooted at ν, and removing the parent virtual edge of ν (if not the root itself). Then the pertinent graph of the root of $\mathcal{T}$ is G. ◇

4.1 High-level procedure for general MEI

We again start with an illustration of the "simple" case of $|F| = 1$. The central theorem of [26] states that an optimal solution of MEI$(G, \{uv\})$ for biconnected G can be obtained by looking only at the shortest path in the SPR-tree $\mathcal{T}$ of G between a node whose skeleton contains u and a node whose skeleton contains v. For each skeleton S_ν along this path, one simply finds an optimal embedding and a shortest partial route in this embedding between the virtual edge representing u (or u itself) and the virtual edge representing v (or v itself). In the case of S- and P-nodes this route requires no crossings. For an R-node ν, one needs a shortest route in the rigid setting. To this end, for each edge $\nu\mu$ in $\mathcal{T}$, the virtual edge e_μ in S_ν with ends x, y is assigned its *pertinent weight* $w(e_\mu)$: the size of a minimum x–y edge-cut in the pertinent graph P_μ. See [26] for more details.

Biconnected case. We turn to the general MEI problem for biconnected planar graphs G. Considering an arbitrarily rooted SPR-tree $\mathcal{T}$ of G, we devise a dynamic programming procedure to solve MEI bottom-up over $\mathcal{T}$. The core of the procedure is to describe which subproblems we are going to solve at each node ν of $\mathcal{T}$, assuming we know the solutions of the corresponding subproblems at the child nodes of ν. For better understanding, this task is illustrated and described in Figure 8. We say a virtual edge e_μ in S_ν representing a child node μ of ν, is *dirty* if its pertinent graph P_μ contains an end vertex of $f \in F$ which is not one of the ends of the twin e_ν.

At a high level, we describe our procedure as follows:

(A) We compute the pertinent weights $w(e_\mu)$ for all non-dirty virtual edges that appear in skeletons used in the following step (B).

(B) For each dirty child virtual edge e_μ in the skeleton S_ν, we assume to know the optimal number of crossings for every "reasonable" scheme of routing edges of F to, from, or across e_μ (we stress that we speak about *all* edges of F, including those not having an end in P_μ). We exhaustively process, over all possible embeddings of S_ν in case of a P-node, all the subproblems stated by every admissible combination of routing schemes for the dirty child virtual edges in S_ν and for the parent virtual edge of S_ν. For each "reasonable" routing scheme of the parent virtual edge, we then select and store the best obtained solution in terms of minimal overall crossing number.

(C) Each particular subproblem of the exhaustive processing in (B) can be formulated as a weighted r-MEI(H, F'') instance. The plane graph H is constructed from S_ν by inserting

special gadgets at the dirty virtual edges (see the colored digons in Figure 8), and the new edges F'' are suitable segments of the edges of F (their ends are either an end of the original F-edge or a vertex on such a gadget). All non-virtual edges of S_ν and all of F'' have weight 1. Each non-dirty virtual edge e_μ gets weight $w(e_\mu)$, computed in (A), and all dirty virtual edges get weight ∞. See Figure 8 for closer details.
This r-MEI(H, F'') instance can then be solved using Theorem 3.16.

(D) If ν is the root node of $\mathcal{T}$, then there is no parent virtual edge and the exhaustive processing in (B) selects a single optimal solution. Realizing this solution, together with the corresponding subsolutions at the descendants, gives an optimal solution to the original MEI(G, F) instance. ◇

We first give a more precise definition regarding step (B). For a dirty virtual edge $e_\mu = xy$ in the skeleton S_ν with its pertinent graph P_μ, let M be the set of those ends of the edges of F that belong to $V(P_\mu) \setminus \{x, y\}$. A *routing scheme* of e_μ is an arbitrary pair of disjoint sequences Q_1, Q_2, where $(Q_1 \cup Q_2) \cap V(G) = \emptyset$, together with a perfect matching on the set $M \cup Q_1 \cup Q_2$ (the cardinality of which must be even). The matching edges are meant to represent segments of edges of F drawn across the pertinent graph P_μ, but on this level we do not need to distinguish which segment belongs to which of the F-edges.

If φ is the parent node of ν then a routing scheme of e_φ in S_ν, as dealt with in step (B), is formally not the same as the corresponding routing scheme of the twin virtual edge e_ν in S_φ; these two are of course actually "complementary". We neglect this technical difference in a high-level description of our dynamic programming scheme.

Second, we informally outline three claims which make the above high-level procedure run in FPT time:

- A skeleton S_ν may contain more than k virtual edges, but only at most $2k$ of them may be dirty (containing an end vertex of F). The computation of the pertinent weights of non-dirty virtual edges can be done in linear overall time.

- A P-node skeleton S_ν may have an unbounded number of inequivalent embeddings (precisely $(q-1)!$ where q is the number of parallel edges in S_ν). However, if two *non-dirty* edges e_1, e_2 appear non-consecutively within the embedding of S_ν, there exists an alternative solution with the same number of crossings where e_2 is rerouted alongside e_1 (by optimality, both edges encounter the same number of crossings either way). Thus we can restrict our attention to only those at most $(2k)!$ embeddings in which all non-dirty virtual edges are consecutive (in any internal order) within the embedding of S_ν.

- In contrast to the single edge insertion, an edge of F may easily be forced to cross the same virtual edge e_μ multiple times (a number depending only on k). However, the complexity of any "reasonable" routing of the edges of F across e_μ is bounded by a function of $k = |F|$, as shown in Lemma 3.8.

As a corollary we see that the number of subproblems generated in step (B), as well as the amount of information stored at any SPR-tree node, is bounded by a function of k.

The solution of each single subproblem in step (C) can be obtained using the algorithm for r-MEI in Theorem 3.16 in linear time for constant k, and there are at most $\mathcal{O}(|V(G)|)$ nodes in the tree $\mathcal{T}$. Instead of the naïve quadratic runtime bound, we even achieve a linear overall runtime bound by observing that the union of all skeletons is still only of $\mathcal{O}(|V(G)|)$ size. We thus obtain, as given in the introduction:

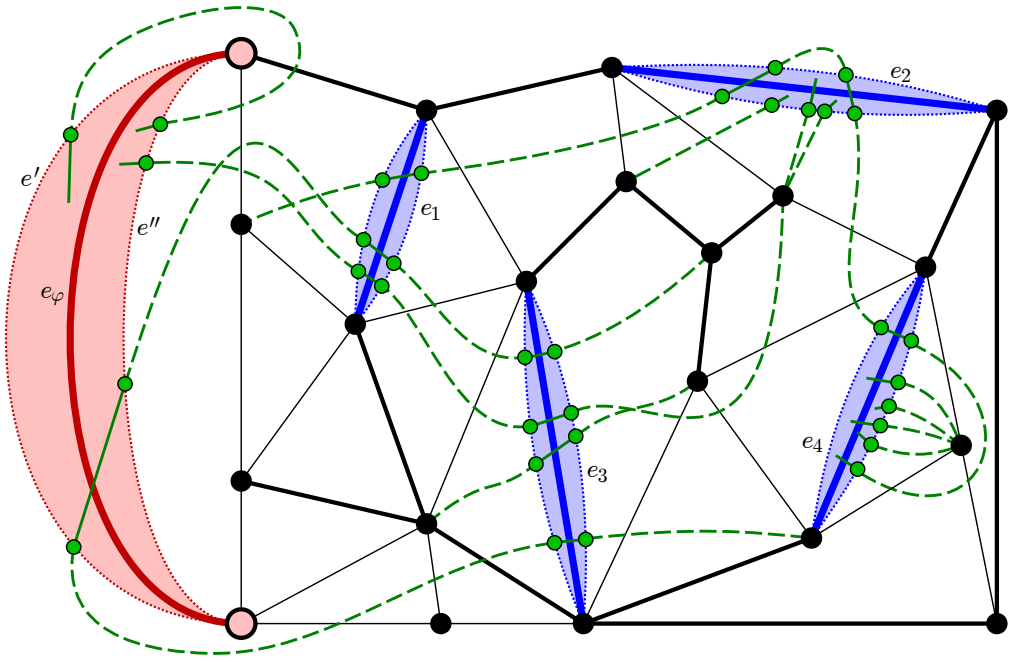


Figure 8: One of the possible r-MEI instances considered at a rigid R-node ν in step (C). The mostly black graph shows the skeleton S_ν, whereby the red virtual edge e_φ corresponds to ν's parent, the blue virtual edges $e_1, \ldots, e_4$ to ν's dirty children, and the thick black edges to the non-dirty children with their implicit pertinent weights. The inserted edges of F are depicted in green.

For each dirty virtual edge e_i, including e_φ, we consider a fixed (by the processing in step (B)) *routing scheme* that externally encodes at which points segments of edges of F enter and leave the pertinent subgraph of e_i. This is depicted with a blue or red digon of e_i and solid green segments of the F-edges within it. Consequently, we do not have to care for internal details of the digonal parts (they are, in fact, not part of our r-MEI instance and only visualized for the reader's context), as they are either already resolved in the subproblems of the children or are going to be resolved in the ancestor nodes. Likewise, for an r-MEI instance at ν, the only relevant information for any non-dirty virtual edge is its pertinent weight and no subembedding details are necessary.

The *gadget* (see step (C)) used to enforce the aforementioned routing schemes at every dirty virtual edge e_i, is composed of the edge e_i itself (solid blue or red), and two new edges e_i', e_i'' parallel to e_i (dotted) which are subdivided by the prescribed entry/exit *terminals* (green dots) of the segments of F-edges. The gadget edges are weighted ∞ and their embedding is rigid. Possible end vertices of F-edges in $V(S_\nu)$ are also counted as the terminals. By a combination of the routing schemes we mean an arbitrary perfect matching (the green dashed segments) on these terminals. It is an *admissible combination* (cf. step (B)) if the union of the prescribed solid green and the matching dashed green segments forms an actual "realization" of the inserted edges of F.

An admissible combination then gives rise to the depicted r-MEI instance, as described in step (C), which can be solved using Theorem 3.16.

Theorem 4.2 (The biconnected case of Theorem 1.1)**.** *Let G be a planar biconnected graph and F a set of $k \geq 1$ edges to insert. An optimal solution of the problem* MEI(G, F) *can be found in $\mathcal{O}(2^{q(k)} \cdot |V(G)|)$ time, where $q(k)$ is a polynomial function in k. In fact, $q(k) = \mathcal{O}(k^{30} \log k)$.*

Connected case. For essentially all known insertion algorithms (in particular the single edge insertion [26], the vertex insertion [12], and the MEI approximation [13]), one typically first resolves the case of biconnected graphs (using SPR-trees as above). Then, it is relatively straightforward to lift the algorithms to connected graphs, by considering BC-trees (see below). Interestingly, this step seems much more complicated in the case of exact MEI.

Consider the well-known *block-cut tree* (BC-tree) decomposing any connected graph into its blocks (biconnected components). Using analogous techniques as in [13], we extend our dynamic programming approach by amalgamating the BC-tree of G with respective SPR-trees of the blocks, to obtain a linear-sized combined tree (called con-tree in [13])—with an additional node type **C**, for the cut vertices. The procedure outline (A)–(D) from the previous case is completed, for the non-biconnected case, with the following:

(B$^+$) If ν is a C-node representing a cut vertex c of G, then we exhaustively process all possibilities to combine the dirty blocks of G incident to c (while non-dirty blocks can be safely ignored).

Unfortunately, although we can again bound the number of dirty blocks by $2k$, there is now no easy "external description of routing" with respect to a cut vertex available (analogous to a routing scheme of a virtual edge). Consequently, the number of possibilities to consider in (B$^+$) depends on k and the degree of the cut vertex c. We conclude:

Theorem 4.3 (The connected case of Theorem 1.1)**.** *Let G be a planar connected graph and F a set of $k \geq 1$ edges to insert. Let Δ_c be the maximum degree over cut vertices of G. Then an optimal solution of the problem* MEI(G, F) *can be found in $\mathcal{O}\big(2^{q'(k)} \cdot \Delta_c^k \cdot |V(G)|\big)$ time, where $q'(k)$ is a polynomial function in k.*

4.2 Decomposing into subproblems

We start with formally defining the subproblems to be solved and stored at each dirty non-root decomposition node. Let ν be such a node and let $e = xy \in E(S_\nu)$ be the virtual edge in the skeleton of ν corresponding to its parent node φ. Recall that the pertinent graph P_ν arises from S_ν by merging the skeletons of the subtree rooted at ν and removing the sole remaining virtual edge e. We consider the 3-partition of F into F_0, F_1, F_2, where F_0 are the edges without an end in $V(P_\nu) \setminus \{x, y\}$, F_1 are the edges with one end in $V(P_\nu) \setminus \{x, y\}$ and the other not in $V(P_\nu)$, and F_2 are the edges with one end in $V(P_\nu) \setminus \{x, y\}$ and the other in $V(P_\nu)$.

By definition, the graph $P_+ := P_\nu + e$ is planar, and e represents the "rest of the graph" disjoint from P_ν. We are, intuitively, interested in the best embedding P_+° of P_+ to

(a) route the edges of F_1 from a "side of e" to its end in $V(P_\nu) \setminus \{x, y\}$; observe that we may care from which side of e the new edge emanates.

But these are not the only routes to consider in an optimal solution:

(b) edges $uv \in F_2$ may be routed completely within P_ν, or drawn partially from u to some side of e (to the "rest of the graph") and from some side of e (from the "rest of the graph"; either from the other or even the same side!) to v;

(c) a segment of any edge of F may be routed through P_ν, i.e., from one side of e to the other side.

The task is illustrated in Figure 9.

Formally, we can define a single *routing query* (at the node ν) as a pair (s,t), where s and t are each either referencing a specific side of e or a vertex in $V(P_\nu) \setminus \{x,y\}$. We will use e', e'' to denote the two different *sides* of e. In such a routing query, we ask for a routing of a new edge between s and t in P_+, without crossing over e. Note that, with respect to the informal notation of Subsection 4.1, a combination of appropriate routing queries at ν forms a routing scheme of the virtual edge e_ν at the parent of ν, and vice versa.

Lemma 3.8 (which holds for every fixed embedding, and hence for each possible embedding) shows that a triedge of the trinet T of G is crossed by any given edge $f \in F$ at most $8k^4 = \mathcal{O}(k^4)$ times. When computing a shortest route (w.r.t. some T-sequence) between two succeeding triedges, we clearly have the property that any edge within the corresponding tricell is crossed at most once. Hence:

Corollary 4.4. *In an optimal solution to* MEI(G,F)*, each edge $f \in F$ crosses any edge of G at most $\xi = \mathcal{O}(k^4)$ times.*

Since this corollary also holds for virtual edges in a skeleton, we have the same upper bound for crossings through a biconnected component P_ν.[1]

In our dynamic programming scheme, we will hence—for each possible set of (reasonable) routing queries—store the minimum number of crossings necessary over all embeddings of P_+. A specific set of routing queries (to be described in details below) is called a *subproblem*, and the corresponding number of crossings (together with the embedding of P_+ and the corresponding routings, if desired) is called a *subsolution*. It remains to discuss the number of subproblems for ν.

Lemma 4.5. *Each subproblem specifies at most $r := \mathcal{O}(k^5)$ routing queries. The total number of subproblems to consider at any node ν is bounded by $\chi := 2^{\mathcal{O}(k^5 \log k)}$.*

Proof. Consider the routing query types (a)–(c) as above:

(a) For each edge $f = uv \in F_1$ with $u \in V(P_\nu) \setminus \{x,y\}$, we have to pick one of the two routing queries (e', u), (e'', u).

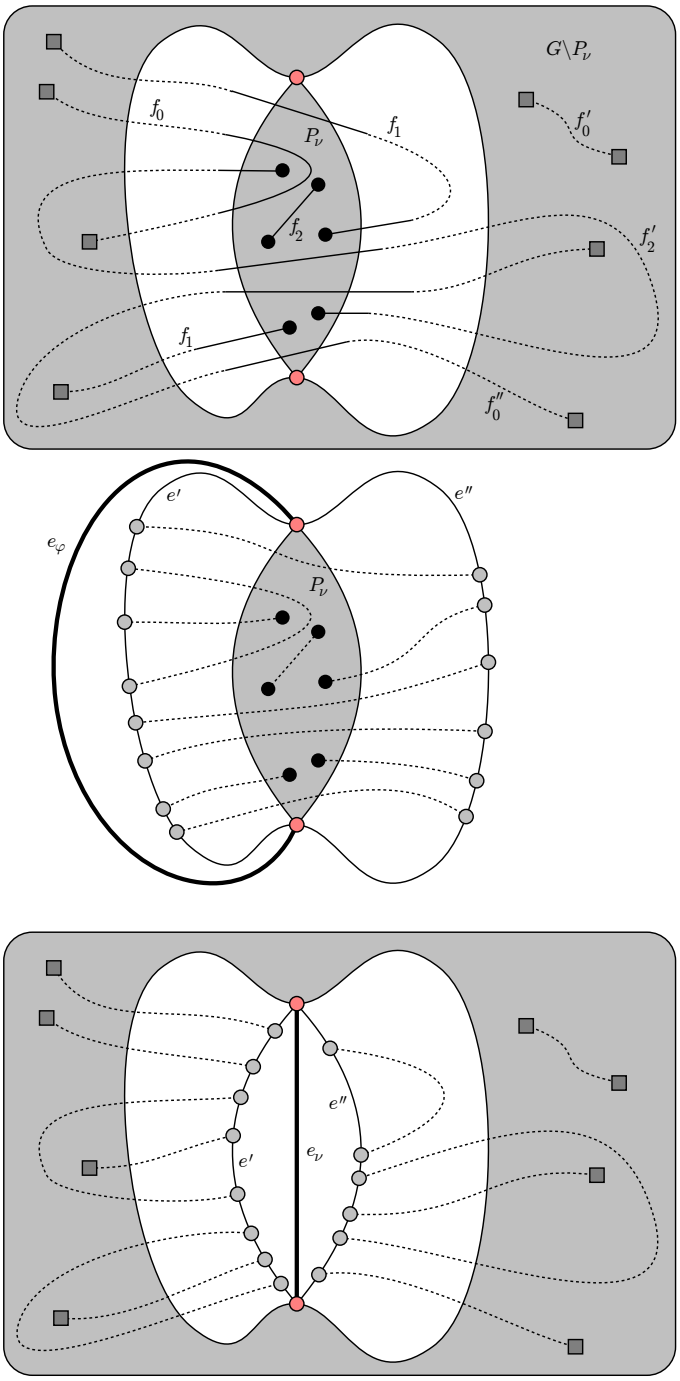
(b) For each edge $f = uv \in F_2$, we have to pick one out of five options: (i) a single routing query (u,v); (ii)–(v) two routing queries $(u, e^{(1)})$, $(e^{(2)}, v)$, with $e^{(1)}, e^{(2)} \in \{e', e''\}$.

(c) Finally, for each $f \in F$—except for those F_2-edges that picked option (i)—we have additional up to $\xi - 1$ (Corollary 4.4) routing queries. Each such additional query is of one of four types: $(e^{(1)}, e^{(2)})$, with $e^{(1)}, e^{(2)} \in \{e', e''\}$.

Overall, this sums up to $|F_1| + 2|F_2| + \xi|F| = \mathcal{O}(k^5) =: r$ routing queries.

The number of choices for such a set of routing queries is at most $2^{|F_1|} \cdot 5^{|F_2|} \cdot (4^\xi)^{|F|} = \mathcal{O}(2^{\mathcal{O}(k^5)})$. However, up to now we did not consider a crucial interplay of these individual routing queries—we need to take all possible orderings of the edges emanating from a side of e into account. Sides of e arise at most $2r$ times over all queries, and we hence have at most $(2r)!$ orderings to consider. Thus, we overall obtain at most $2^{\mathcal{O}(k^5)} \cdot 2^{\mathcal{O}(k^5 \log k)}$ subproblems. □

[1] Note that we can derive an even stronger quadratic bound here, directly using Claim 3.10, but that would not stregthen the overall runtime analysis.



Consider a drawing of G, where the pertinent subgraph P_ν (gray lense-shaped region) of some virtual edge e_ν is crossed as shown. The "rest of the graph" is visualized as the rectangular gray region. The inserted edges F are drawn as solid-and-dashed edges: we have $f_0, f_0', f_0'' \in F_0$, $f_1, f_1' \in F_1$, and $f_2, f_2' \in F_2$.

One part of the above situation is described as a subproblem at node ν. The subproblem is characterized by the order (and number) in which the edges of F enter (leave) P_ν. Its optimal solution is computed within $P_+ = P_\nu + e_\varphi$. Thereby, e_ϕ (thick edge) is the virtual edge representing ν's parent $\varphi = \varrho$. We set its weight to ∞ and add subdivided edges e', e'' left and right of e. The subdivision vertices (grey circles) are constructed according to the subproblem characterization.

If the above subproblem is solved, we can solve the problem on G without explicitly requiring P_ν: we replace P_ν in G by its virtual edge $e_n u$ (of weight ∞) and add side edges e', e'', that correspond to the considered subsolution.
If we perform this operation on all subsolutions at ν, we find the overall optimum solution in G.

Figure 9: Conceptual sketch of a subproblem at a node ν. For simplicity, assume that the SPR-tree consists of only two nodes: the root ϱ and ν.

4.3 Dynamic programming

Finally, we have to describe how to use these subproblems to efficiently compute MEI. The validity of this approach for non-dirty pertinent graphs was already established in [26]. As mentioned, we consider dirty nodes bottom-up.

Let ν be the considered SPR-tree node with skeleton S_ν. Let $e_\varphi \in E(S_\nu)$ be the virtual edge corresponding to ν's parent φ (if it exists), and $e_1, \dots, e_\ell$ ($e'_1, \dots, e'_{\ell'}$) the dirty (non-dirty) virtual edges in S_ν corresponding to the children $\mu_1, \dots, \mu_\ell$ ($\mu'_1, \dots, \mu'_{\ell'}$, respectively). We need to show that we can solve each subproblem at ν purely using S_ν and the solutions to the subproblems of the dirty children. In particular, we may not expand the skeleton to the pertinent graph (for which the subproblems at ν are actually defined).

Subproblems, embeddings, and the root. Assume, ν is a non-root node, then we have to solve up to χ many subproblems as claimed by Lemma 4.5. For each subproblem, we are given a set of up to r routing queries, and want to find the optimal solution over all embeddings of P_+. Recall that there are only a bounded number of embeddings for S_ν: 1, 2, and $(2k)!$ in case of an S-, R-, and P-node, respectively. Let $\chi' := (2k)! = 2^{\mathcal{O}(k \log k)}$. We hence can enumerate each embedding explicitly.

The routing queries of the considered subproblem give rise to the following gadget: Set the weight of e_φ to ∞, and introduce two edges e' and e'' parallel to e_φ, one directly to its left, one directly to its right. (We use the same identifiers e', e'' as for the sides of e, as they are their representatives within the gadget.) Now subdivide these two edges such that there is a vertex on e' (e'') if a routing query specifies the edge side e' (e''). Furthermore, these vertices are ordered according to the specification of the subproblem. Let S'_ν denote the embedded graph arising from this construction; we do not consider e_φ as a virtual edge in the following any more. Instead of considering the original edge set F, we will now consider the routing queries (s, t) as edges $st \in F'$ (a new set F') to be inserted.

If ν is the root node, we also have to consider all its skeleton's possible embeddings S'_ν individually, but there is no specific subproblem to consider and we simply set $F' := F$ without any gadget construction. From now on, we do the same steps, independent of whether considering the root node, or a specific subproblem at a non-root node.

Virtual edges. For each non-dirty virtual edge $e'_i = ab$, $1 \le i \le \ell'$, we set the weight of e'_i to the minimum-ab-cut in the pertinent graph of μ'_i. Note that these values can be computed in overall linear time in a preprocessing step.

Now, for each dirty virtual edge e_i, $1 \le i \le \ell$ ($\ell \le 2k$), we construct a gadget analogous to the gadget for e_φ: Edge e_i gets weight ∞, we add two edges e'_i, e''_i left and right of e_i, and subdivide them according to a subproblem at μ_i. To do this, we have to enumerate all possible choices of subproblems at all virtual edges. So this construction yields up to $\chi'' := \chi^{2k} = 2^{\mathcal{O}(k^6 \log k)}$ different choices, each of which we consider individually. Observe: If, for some edge $f' \in F$ that resides within some P_{μ_j}, we chose a routing query of type (b)(i), we do not consider subsolutions at any virtual edge where there are type (c) queries w.r.t. f'. We call such an edge f' a *local edge*.

We denote the so-modified plane graph by S''_μ, and now have to decide what happens to our new edge set F'. Each edge in F' corresponds to some edge in F. Furthermore, we add each non-local edge $f \in F$ to F' if it has no corresponding edge in F'. We observe that for each vertex $w \in P_\nu \setminus S_\nu$ that is an end in F', there is a unique *replacement vertex* $r(w)$ in S_ν—it arises from a query $(r, r(w))$ (unoriented) within a subproblem at some dirty virtual edge.

For each original edge $f = uv \in F$, we hence get a partial order of routing queries corresponding to it: either u (v) or its replacement vertex $r(u)$ ($r(v)$, respectively) is in S'_ν, so we start (end) there. There may or may not be a routing query starting at u (ending at v), which we would update to use $r(u)$ ($r(v)$) instead of u (v). Now, between this start and end, f may have to "visit" former queries of type (c) (whose ends are now represented by subdivision vertices at edges e'_j, e''_j, $1 \le j \le \ell$). While these former queries are totally ordered for each individual dirty virtual edge, it is unclear in which order f visits the different virtual edges. We will enumerate all possible orders to visit each of the $\mathcal{O}(k)$ dirty virtual edges up to $\mathcal{O}(\xi)$ times; there are hence $\chi''' := O\left(\frac{(k\,\xi)!}{k\,\xi!}\right) = \mathcal{O}(2^{\mathcal{O}(k^5 \log k)})$ different visit orderings for each edge of $f \in F$. Every visit order induces an unambiguous set T_f of (new) routing queries to draw part of f within S_ν: from f's start to the vertex representing the beginning of a former query, from the vertex representing the end of the last former query to the beginning of the next former query, and so on, until finally from the vertex representing the end of the last former query to f's end. Such a set T_f hence has size at most $\tau := \mathcal{O}(k\xi) = \mathcal{O}(k^5)$.

Finally, we obtain an instance r-MEI(S'_ν, F''), where $|V(S'_\nu)| = \mathcal{O}(|S_\nu| \cdot k\xi) = \mathcal{O}(|S_\nu| \cdot k^5)$ and F'' is the set of all routing queries (interpreted as unordered new edges) obtained from F' by considering each T_f (for all $f \in F$). We have $|F''| = \mathcal{O}(k \cdot \tau) = \mathcal{O}(k^6)$. The total cost of the considered subsolution (and also for the solution at the root node) is the minimum number of crossings over all possible r-MEI instances constructed as above *plus* the numbers of crossings given by the corresponding individual subsolutions realized at the dirty virtual edges. We have got:

Claim 4.6. Algorithm 2 settles the root node—and each specific subproblem at a non-root node—with $\mathcal{O}\big(\chi' \cdot \chi'' \cdot (\chi''')^k\big)$ calls to r-MEI. Each of the dirty non-root nodes is done with $\mathcal{O}\big(\chi \cdot \chi' \cdot \chi'' \cdot (\chi''')^k\big)$ calls to r-MEI.

4.4 Proving Theorem 4.2 (biconnected case of Theorem 1.1)

Proof. We refer to Algorithm 2. Let n the number of vertices in G, and let p be the polynomial function in k from the r-MEI runtime described in Theorem 3.16. In each of our subproblems, we have at most $\mathcal{O}(k^5)$ routing queries (Lemma 4.5), and thus consider an r-MEI instance for $k' = \mathcal{O}(k^5)$ inserted edges. Similarly, each of our $\mathcal{O}(k)$ gadgets in this instance has size at most $k \cdot \xi = \mathcal{O}(k^5)$ (Lemma 4.4). Hence, each individual r-MEI instance in our setting can be computed within $\mathcal{O}\big(|V(S_\nu)| \cdot k^6 \cdot 2^{p(k^5)}\big) = \mathcal{O}(|V(S_\nu)|) \cdot 2^{\mathcal{O}(p'(k))}$ time, for a polynomial function p'.

Furthermore, the sum of the amounts $|V(S_\nu)|$ over all recursively considered SPR-tree skeletons S_ν in the algorithm is still linear in n, as it follows from Definition 4.1. We hence obtain an upper bound on the overall runtime

$$\begin{aligned} &\mathcal{O}\big(\chi \cdot \chi' \cdot \chi'' \cdot (\chi''')^k\big) \cdot \mathcal{O}(n) \cdot 2^{\mathcal{O}(p'(k))} = \\ &= \mathcal{O}(n) \cdot 2^{\mathcal{O}(p'(k))} \cdot 2^{\mathcal{O}(k^5 \log k)} \cdot 2^{\mathcal{O}(k \log k)} \cdot 2^{\mathcal{O}(k^6 \log k)} \cdot 2^{\mathcal{O}(k^6 \log k)} = \\ &= \mathcal{O}(n \cdot 2^{q(k)})\,, \end{aligned}$$

where q is a polynomial function in k. In fact, $q(k) = \mathcal{O}(k^{30} \log k)$. □

We note in passing that the presented runtime bound can be improved to nearly match that of Theorem 3.16, by observing that Lemma 3.8 can be used to upper-bound the joint length of all the T-sequences of those edges of F'' which belong to the same edge of F. A suitable combined runtime analysis of Theorems 3.16 and 4.2 would then give a stronger upper bound on $q(k)$ above. Though, we skip the lengthy technical details here.

In: a planar biconnected graph G, an edge set F to insert.

Out: an optimal solution to MEI(G, F).

(1) Compute an SPR-tree $\mathcal{T}$ of G in linear time.

(2) Root $\mathcal{T}$ at an arbitrary (root) node $\varrho \in V(\mathcal{T})$

(3) Compute the pertinent weights of non-dirty virtual edges that are going to be considered later in the algorithm: Traverse $\mathcal{T}$ top-down, not descending into subtrees of non-dirty nodes. For each visited non-dirty node ν:

- Let a, b be the end vertices of the virtual edge representing ν's parent φ in S_ν. Compute the value c_{ab} of the minimum-ab-cut in P_ν.[a] In the following, ignore ν and its subtree; instead consider the edge representing ν in φ to be an original edge of weight c_{ab}.

(4) Traverse $\mathcal{T}$ bottom-up, except for the root ϱ. Let ν be the current node and φ its parent. All the children $\mu_1, \ldots, \mu_\ell$ of ν are dirty and already processed by the algorithm. They correspond to the virtual edges $e_1, \ldots, e_\ell$ in S_ν, while φ corresponds to a virtual edge e_φ.

- For each feasible set of routing queries $\mathcal{R}$ at ν (these are the subproblems at ν, and we have $\chi = 2^{\mathcal{O}(k^5 \log k)}$ many of them by Lemma 4.5):
  (a) Consider all combinations of:
    i. every non-equivalent embedding S'_ν of S_ν ($\chi' = 2^{\mathcal{O}(k \log k)}$ many),
    ii. each combination of subsolutions for all children μ_i, $i = 1, \ldots, \ell$, consistent with $\mathcal{R}$ ($\chi'' = 2^{\mathcal{O}(k^6 \log k)}$ many possibilities),
    iii. each combination (over all $f \in F$) of the orders in which each f visits the virtual edges of S_ν, including multiple visits (χ''' for each f, and so $(\chi''')^k = 2^{\mathcal{O}(k^6 \log k)}$ possibilities overall)
  (b) Fix the embedding S'_ν.
  (c) Replace e_φ with the gadget representing $\mathcal{R}$.
  (d) For $i = 1, \ldots, \ell$; replace each e_i with the gadget representing the currently considered subsolution at μ_i.
  (e) For $j = 1, \ldots, k$; deduce the set F'' from $\mathcal{R}$, all S_i, and the visit orders.
  (f) Let G' be the so-constructed plane graph (we have $|V(G')| = \mathcal{O}(|S_\nu| \cdot k^5)$ and $|F''| = \mathcal{O}(k^6)$). Solve $x :=$ r-MEI(G', F''). The solution value for the considered combination is $x + \sum_{i=1}^{\ell} x(S_i)$, where $x(S_i)$ is the solution value of S_i.
  (g) Store the minimum observed solution value over all cases described in (a) as the solution $x(\mathcal{R})$ for subproblem $\mathcal{R}$ at ν.

(5) Solve the root node ϱ. (It has no parent and thus needs no specific $\mathcal{R}$ as in step (4) above.) Perform the substeps (a), (b), (d)–(f) from above.
Return the thereby minimum observed solution (over all the cases described in (a)) as the overall optimum solution to MEI(G, F).

[a] For unweighted G, this can be done in linear time using BFS in the dual of any plane embedding of $P_\nu + ab$.

Algorithm 2: The dynamic programming to solve MEI (biconnected case).

4.5 Proving Theorem 4.3 (connected case of Theorem 1.1)

Proof. Until now, the proofs only considered biconnected G. If G is only connected, we can first decompose (in linear time) G into its biconnected components (blocks), and establish a BC-tree $\mathcal{B}$. The tree $\mathcal{B}$ has two types of nodes: For each block of G, we have a node of type **(B)**; for each cut vertex in G, we have a node of type **(C)**. We have an edge $\beta\gamma$ in $\mathcal{B}$ if, and only if, β is a B-node, γ is a C-node, and the block of β contains the cut vertex of γ. We may root $\mathcal{B}$ arbitrarily at any dirty block; we say a block is *dirty* if it contains at least one end of F (other than possibly its parent cut vertex). Clearly, we can iteratively prune non-dirty B-leaves.

Now, we can construct a combined tree $\mathcal{C}$: For each block B in G, we construct (and root) its SPR-tree $\mathcal{T}_B$. In $\mathcal{B}$, we replace each B-node with the root vertex of the block's corresponding SPR-tree. Now, we can run the dynamic programming algorithm over $\mathcal{C}$ instead of a single SPR-tree.

Let ν be a non-C-node whose parent is a C-node γ corresponding to cut vertex $c \in S_\nu$. We need to redefine the subproblems considered at ν: instead of considering routing queries that attach to one of the two sides of the parent virtual edge, our routing queries may now attach to c in a specified order and through specified faces incident to c. We therefore introduce the gadget—for each considered embedding S'_ν of S_ν—obtained by planarly replacing c by a simple cycle C. The c-incident edges are attached to C such that the contraction of C again gives S'_ν. When considering the routing queries, instead of the two choices of the side of the parent virtual edge, we now hence have a $\delta(c) = \mathcal{O}(\Delta_c)$-fold choice over the segment of C where to attach to, where Δ_c denotes the maximum degree over all cut vertices.

In our dynamic programming, we will perform no operation at C-nodes, but let ν now be a node with a C-child γ corresponding to cut vertex $c \in S_\nu$. Analogous to above—in each considered embedding S'_ν of S_ν—we planarly replace c by a cycle C. On C, we realize all subsolutions of all (at most $2k$) children of γ, in all possible combinations. Except for these modifications, the algorithm remains unchanged. □

5 Conclusion

In this paper, we have affirmatively answered the long-standing open question, floating around ever since [1, 26], whether there is a polynomial-time algorithm to insert a constant-sized set of edges into a planar graph in a crossing-minimal way. Previously, this has only been known for single edges; the problem with multiple edges could only be approximated. Our result also gives a slightly improved approximation algorithm (compared to [13]) for the general crossing number of graphs with bounded number of edges beyond planarity; we use the computed optimal solution to MEI(G, F) both as an upper bound on the crossing number, and for a lower bound on the crossing number via Theorem 2.2. However, the improvement is not very significant since we also trade polynomial runtime of [13] for FPT runtime.

While the original MEI problem was defined over unweighted graphs, we considered weighted graphs in our subproblems but only to a limited extent. It is thus natural to ask:

Open Problem 5.1. Is MEI fixed-parameter tractable if both G and F are weighted?

In fact, we can straightforwardly answer this question affirmatively for weighted G and unweighted F, as should be clear from the above proofs. We may also assume weighted F, if the weights are bounded by k, by simply adding multiple copies of an edge to F. For generally weighted

F, we observe that the dynamic programming part of solving MEI would be capable of achieving this feat. However, the required r-MEI subproblems are not, due to the technical rerouting argument in the proof of Lemma 3.8. It is not easy to circumvent this argument and the problem seems surprisingly related to that of decidability of string graphs—a connection that deserves future investigation.

Another, probably most interesting open question related to this paper is about the MEI problem in case of a generally disconnected planar graph G. In an extreme subcase, G has no edges and the problem MEI(G, F) becomes an ordinary crossing-number problem parameterized by the number of edges, which has a trivial brute-force solution. However, considering G with "many" non-trivial components adds another level of difficulty, to which our decomposition-based brute-force handling of all possible embeddings of G does not have an answer in FPT time. Similar difficulties with disconnected predrawn skeletons have been faced in the setting of partially predrawn crossing number, and a significant and complex part of the new contribution of [27] lies in allowing disconnected skeletons (unfortunately, as the approaches of [27] are not comparable with ours, we cannot make use of that solution here).

As a perhaps easier subcase of the previous question we suggest:

Open Problem 5.2. Is MEI fixed-parameter tractable for simply-connected graphs G with the parameter independent of Δ_c (unlike in Theorem 4.3)?

Such a dependency on Δ_c does not show up when inserting a single edge or a star into planar G [12, 26]. On the other hand, even more restrictive degree dependencies are very common and seemingly unavoidable in the known general crossing number approximations.

Acknowledgments. We thank Sergio Cabello and Carsten Gutwenger for helpful discussions.

References

[1] C. Batini, M. Talamo, and R. Tamassia. Computer aided layout of entity relationship diagrams. *J. Syst. Softw.*, 4(2-3):163–173, 1984. doi:10.1016/0164-1212(84)90006-2.

[2] G. D. Battista and R. Tamassia. On-line planarity testing. *SIAM J. Comput.*, 25(5):956–997, 1996. doi:10.1137/S0097539794280736.

[3] S. N. Bhatt and F. T. Leighton. A framework for solving VLSI graph layout problems. *J. Comput. Syst. Sci.*, 28(2):300–343, 1984. doi:10.1016/0022-0000(84)90071-0.

[4] T. Biedl, M. Chimani, M. Derka, and P. Mutzel. Crossing number for graphs with bounded pathwidth. *Algorithmica*, 82(2):355–384, 2020. doi:10.1007/s00453-019-00653-x.

[5] D. Bienstock and C. L. Monma. On the complexity of embedding planar graphs to minimize certain distance measures. *Algorithmica*, 5(1):93–109, 1990. doi:10.1007/BF01840379.

[6] S. Cabello. Hardness of approximation for crossing number. *Discrete & Computational Geometry*, 49(2):348–358, 2013. doi:10.1007/s00454-012-9440-6.

[7] S. Cabello and B. Mohar. Crossing number and weighted crossing number of near-planar graphs. *Algorithmica*, 60(3):484–504, 2011. doi:10.1007/s00453-009-9357-5.

[8] S. Cabello and B. Mohar. Adding one edge to planar graphs makes crossing number and 1-planarity hard. *SIAM J. Comput.*, 42(5):1803–1829, 2013. doi:10.1137/120872310.

[9] B. Chazelle. A theorem on polygon cutting with applications. In *Proc. FOCS '82*, pages 339–349. IEEE Computer Society, 1982. doi:10.1109/SFCS.1982.58.

[10] M. Chimani. *Computing Crossing Numbers*. PhD thesis, TU Dortmund, Germany, 2008. http://hdl.handle.net/2003/25955.

[11] M. Chimani and C. Gutwenger. Advances in the planarization method: Effective multiple edge insertions. *J. Graph Algorithms Appl.*, 16(3):729–757, 2012. doi:10.7155/jgaa.00264.

[12] M. Chimani, C. Gutwenger, P. Mutzel, and C. Wolf. Inserting a vertex into a planar graph. In C. Mathieu, editor, *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 375–383. SIAM, 2009. URL: http://dl.acm.org/citation.cfm?id=1496770.1496812.

[13] M. Chimani and P. Hliněný. A tighter insertion-based approximation of the crossing number. In L. Aceto, M. Henzinger, and J. Sgall, editors, *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I*, volume 6755 of *Lecture Notes in Computer Science*, pages 122–134. Springer, 2011. doi:10.1007/978-3-642-22006-7_11.

[14] M. Chimani, P. Hliněný, and P. Mutzel. Vertex insertion approximates the crossing number of apex graphs. *Eur. J. Comb.*, 33(3):326–335, 2012. doi:10.1016/j.ejc.2011.09.009.

[15] M. Chimani, P. Hliněný, and G. Salazar. Toroidal grid minors and stretch in embedded graphs. *J. Comb. Theory, Ser. B*, 140:323–371, 2020. doi:10.1016/j.jctb.2019.05.009.

[16] M. Chimani, M. Ilsen, and T. Wiedera. Star-struck by fixed embeddings: Modern crossing number heuristics. In *Proc. GD '21*, volume 12868 of *LNCS*, pages 41–56. Springer, 2021. doi:10.1007/978-3-030-92931-2_3.

[17] J. Chuzhoy. An algorithm for the graph crossing number problem. In L. Fortnow and S. P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 303–312. ACM, 2011. doi:10.1145/1993636.1993678.

[18] J. Chuzhoy, Y. Makarychev, and A. Sidiropoulos. On graph crossing number and edge planarization. In D. Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1050–1069. SIAM, 2011. doi:10.1137/1.9781611973082.80.

[19] J. Chuzhoy and Z. Tan. A subpolynomial approximation algorithm for graph crossing number in low-degree graphs. In S. Leonardi and A. Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 303–316. ACM, 2022. doi:10.1145/3519935.3519984.

[20] É. Colin de Verdière and A. Schrijver. Shortest vertex-disjoint two-face paths in planar graphs. *ACM Transactions on Algorithms*, 7(2):19:1–19:12, 2011. doi:10.1145/1921659.1921665.

[21] G. Even, S. Guha, and B. Schieber. Improved approximations of crossings in graph drawings and VLSI layout areas. *SIAM J. Comput.*, 32(1):231–252, 2002. doi:10.1137/S0097539700373520.

[22] M. R. Garey and D. S. Johnson. Crossing number is NP-complete. *SIAM J. Alg. Discr. Meth.*, 4:312–316, 1983.

[23] I. Gitler, P. Hliněný, J. Leaños, and G. Salazar. The crossing number of a projective graph is quadratic in the face-width. *Electron. J. Comb.*, 15(1), 2008. URL: http://www.combinatorics.org/Volume_15/Abstracts/v15i1r46.html.

[24] M. Grohe. Computing crossing numbers in quadratic time. *J. Comput. Syst. Sci.*, 68(2):285–302, 2004. doi:10.1016/j.jcss.2003.07.008.

[25] C. Gutwenger and P. Mutzel. A linear time implementation of SPQR-trees. In J. Marks, editor, *Graph Drawing, 8th International Symposium, GD 2000, Colonial Williamsburg, VA, USA, September 20-23, 2000, Proceedings*, volume 1984 of *Lecture Notes in Computer Science*, pages 77–90. Springer, 2000. doi:10.1007/3-540-44541-2_8.

[26] C. Gutwenger, P. Mutzel, and R. Weiskircher. Inserting an edge into a planar graph. *Algorithmica*, 41(4):289–308, 2005. doi:10.1007/s00453-004-1128-8.

[27] T. Hamm and P. Hliněný. Parameterised partially-predrawn crossing number. In X. Goaoc and M. Kerber, editors, *38th International Symposium on Computational Geometry, SoCG 2022, June 7-10, 2022, Berlin, Germany*, volume 224 of *LIPIcs*, pages 46:1–46:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.SoCG.2022.46.

[28] J. Hershberger and J. Snoeyink. Computing minimum length paths of a given homotopy class. *Comput. Geom.*, 4:63–97, 1994. doi:10.1016/0925-7721(94)90010-8.

[29] P. Hliněný and M. Chimani. Approximating the crossing number of graphs embeddable in any orientable surface. In M. Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 918–927. SIAM, 2010. doi:10.1137/1.9781611973075.74.

[30] P. Hliněný and G. Salazar. On the crossing number of almost planar graphs. In M. Kaufmann and D. Wagner, editors, *Graph Drawing, 14th International Symposium, GD 2006, Karlsruhe, Germany, September 18-20, 2006. Revised Papers*, volume 4372 of *Lecture Notes in Computer Science*, pages 162–173. Springer, 2006. doi:10.1007/978-3-540-70904-6_17.

[31] P. Hliněný and A. Sankaran. Exact crossing number parameterized by vertex cover. In D. Archambault and C. D. Tóth, editors, *Graph Drawing and Network Visualization - 27th International Symposium, GD 2019, Prague, Czech Republic, September 17-20, 2019, Proceedings*, volume 11904 of *Lecture Notes in Computer Science*, pages 307–319. Springer, 2019. doi:10.1007/978-3-030-35802-0_24.

[32] J. E. Hopcroft and R. E. Tarjan. Dividing a graph into triconnected components. *SIAM J. Comput.*, 2(3):135–158, 1973. doi:10.1137/0202012.

[33] K. Kawarabayashi and B. A. Reed. Computing crossing number in linear time. In D. S. Johnson and U. Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 382–390. ACM, 2007. doi:10.1145/1250790.1250848.

[34] K. Kawarabayashi and A. Sidiropoulos. Polylogarithmic approximation for minimum planarization (almost). In C. Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 779–788. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.77.

[35] P. N. Klein, S. Rao, M. R. Henzinger, and S. Subramanian. Faster shortest-path algorithms for planar graphs. In F. T. Leighton and M. T. Goodrich, editors, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 27–37. ACM, 1994. doi:10.1145/195058.195092.

[36] Y. Kobayashi and C. Sommer. On shortest disjoint paths in planar graphs. *Discrete Optimization*, 7(4):234–245, 2010. doi:10.1016/j.disopt.2010.05.002.

[37] D. Lee and F. P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks*, 14(3):393–410, 1984. doi:10.1002/net.3230140304.

[38] P. Mutzel and T. Ziegler. The constrained crossing minimization problem. In J. Kratochvíl, editor, *Graph Drawing, 7th International Symposium, GD'99, Stirín Castle, Czech Republic, September 1999, Proceedings*, volume 1731 of *Lecture Notes in Computer Science*, pages 175–185. Springer, 1999. doi:10.1007/3-540-46648-7_18.

[39] M. Schaefer. The graph crossing number and its variants: A survey. *Electron. J. Comb.*, Dynamic Surveys (DS21), 2013–2022. Accessed in April 2022. doi:10.37236/2713.

[40] M. Schaefer. *Crossing Numbers of Graphs*. Discrete mathematics and its applications. CRC Press, Taylor & Francis Group, 2017. URL: https://books.google.de/books?id=suBMtAEACAAJ.

[41] M. Thorup. Undirected single-source shortest paths with positive integer weights in linear time. *J. ACM*, 46(3):362–394, 1999. doi:10.1145/316542.316548.

[42] W. T. Tutte. *Connectivity in graphs*, volume 15 of *Mathematical Expositions*. University of Toronto Press, 1966.

[43] T. Ziegler. *Crossing Minimization in Automatic Graph Drawing*. PhD thesis, Saarland University, Germany, 2001.