Shortest Non-Crossing Walks in the Plane*

Jeff Erickson
Department of Computer Science
University of Illinois at Urbana-Champaign
jeffe@cs.uiuc.edu

Amir Nayyeri
Department of Computer Science
University of Illinois at Urbana-Champaign
nayyeri2@cs.uiuc.edu

Abstract

Let G be an n-vertex plane graph with non-negative edge weights, and let k terminal pairs be specified on h face boundaries. We present an algorithm to find k non-crossing walks in G of minimum total length that connect all terminal pairs, if any such walks exist, in $2^{O(h^2)} n \log k$ time. The computed walks may overlap but may not cross each other or themselves. Our algorithm generalizes a result of Takahashi, Suzuki, and Nishizeki [*Algorithmica* 1996] for the special case $h \leq 2$. We also describe an algorithm for the corresponding geometric problem, where the terminal points lie on the boundary of h polygonal obstacles of total complexity n, again in $2^{O(h^2)} n$ time, generalizing an algorithm of Papadopoulou [*Int. J. Comput. Geom. Appl.* 1999] for the special case $h \leq 2$. In both settings, shortest non-crossing walks can have complexity exponential in h. We also describe algorithms to determine in $O(n)$ time whether the terminal pairs can be connected by *any* non-crossing walks.

1 Introduction

We consider the following extension of the classical geometric shortest path problem: Given a set of k pairs of terminal points (s_i, t_i) lying on a small number h of obstacles in the plane, find a set of non-crossing walks of minimum total length that connect the terminal pairs without intersecting the obstacles. The walks may be neither simple nor disjoint; however, they must not cross each other or themselves. The obstacles can either be formalized as a set of simple polygons in the plane, or as a subset of faces in an edge-weighted planar graph G. In the latter formulation, the output must be a set of walks in G. (We give a more formal statement of the problem in Section 2.)

Motivated by problems in VLSI design, Takahashi *et al.* [27] describe an algorithm that finds shortest non-crossing walks in a planar graph, when all terminals lie on at most two obstacle faces, in $O(n \log k)$ time.[1] They observed that when all the terminals lie on a single obstacle, the solution consists of shortest paths between the terminal pairs. For the case of two obstacles, they find 3 paths joining the obstacles, at least one of which is not crossed by the shortest walks; see Figure 1. Thus, by cutting along each of these paths in turn, they reduce two-obstacle problem to three instances of the single-obstacle case. The output walks could have complexity $\Omega(kn)$ in the worst case, however, their algorithm actually computes an implicit representation of complexity $O(n)$. The geometric formulation of the shortest non-crossing walks problem was proposed by Papadopoulou [22], who described a linear-time algorithm, again for the special case of at most two obstacles, using the same cutting strategy as Takahashi *et al.* to reduce two obstacles to one, and using a similar implicit output representation. In a followup paper, Takahashi *et al.* [28] describe an $O(n \log n)$-time algorithm for a rectilinear variant of the geometric problem, where the domain is a rectangle with many rectangular holes, and the terminals lie either on the outer boundary or on the boundary of one hole.

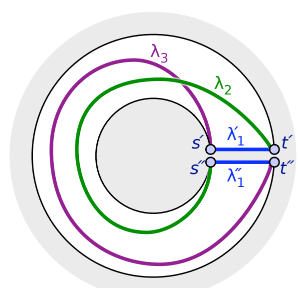

Figure 1. Reducing two obstacles to one, after Takahashi *et al.* [27].

At the other extreme, Bastert and Fekete proved that if the number of obstacles is allowed to be arbitrarily large, finding shortest non-crossing walks in planar graphs is NP-hard [1]. Polishchuk [23] proves that the minmax variant of the geometric problem, where the goal is to minimize the length of the longest path, is strongly NP-hard in general, and weakly NP-hard even when $k = 2$ (but the number of obstacles h is large). Motivated by problems in air-traffic control, Polishchuk

*This research was partially supported by NSF grant CCF 09-15519. See http://www.cs.uiuc.edu/~jeffe/pubs/noncrossing.html for the most recent version of this paper.

[1] Takahashi *et al.*report a running time of $O(n \log n)$, but the time bound can be improved to $O(n \log k)$ by using the linear-time shortest-path algorithm of Henzinger *et al.* [14] in place of Dijkstra's algorithm.

and Mitchell [24, 23] also considered a variant of the problem where the output is a set of 'thick paths'.

In this paper, we show that both formulations of the shortest non-crossing walks problem are fixed-parameter tractable with respect to the parameter h, the number of obstacles. Specifically, in Section 5, we describe an algorithm for the graph formulation that runs in time $2^{O(h^2)} n \log k$, and in Section 6, we describe an algorithm for the geometric formulation that runs in time $2^{O(h^2)} n$, generalizing previous results for the special case $h \leq 2$. Our key insight is the observation, in Section 4, that in the set of shortest non-crossing walks, each walk crosses any arbitrary shortest path at most $2^{O(h)}$ times. This crossing bound allows us to use algorithmic tools previously developed to find shortest cycles in combinatorial surfaces satisfying various topological properties [3, 4, 5, 18]. Like earlier algorithms for the case $h \leq 2$ [27, 22], our algorithms can be easily modified to find non-crossing walks that minimize any non-decreasing function of their lengths, such as the maximum length or the sum of squared lengths.

2 Preliminaries

2.1 Background

Curves in the plane. Let $\mathcal{S}$ be a compact subset of the plane. A ***curve*** in $\mathcal{S}$ is a continuous function $\alpha\colon [0,1] \to \mathcal{S}$. The *endpoints* of α are the points $\alpha(0)$ and $\alpha(1)$. We call a curve α ***simple*** if it is an injective function, and *closed* if $\alpha(0) = \alpha(1)$. The ***concatenation*** $\boldsymbol{\alpha \cdot \beta}$ of two curves α and β with $\alpha(1) = \beta(0)$ is the curve with $(\alpha \cdot \beta)(t) = \alpha(2t)$ if $t \leq 1/2$ and $(\alpha \cdot \beta)(t) = \beta(2t-1)$ if $t \geq 1/2$. The ***reversal*** $\mathbf{rev}(\boldsymbol{\alpha})$ of α is the curve $rev(\alpha)(t) = \alpha(1-t)$. To be consistent with standard graph nomenclature, we will refer to arbitrary curves as ***walks*** and simple curves as ***paths***. We frequently do not distinguish between a path and its image in $\mathcal{S}$.

Two paths α and β in $\mathcal{S}$ ***cross*** if and only if there is an open neighborhood $A \subset \mathcal{S}$ that is homeomorphic to an open disc, such that $A \cap \alpha$ and $A \cap \beta$ are nonempty subpaths of α and β, whose endpoints alternate around the boundary of A; see Figure 2. In other words, α and β cross if they cannot be perturbed within A to become disjoint. Two walks cross if they contain crossing subpaths; a walk is *self-crossing* if it contains two crossing subpaths.

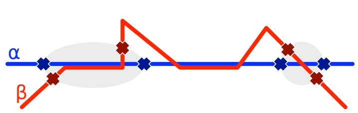

Figure 2. Two paths crossing twice.

A ***homotopy*** between two walks α and β is a continuous function $h\colon [0,1] \times [0,1] \to \mathcal{S}$ such that $h(0,\cdot) = \alpha$, $h(1,\cdot) = \beta$, $h(\cdot,0) = \alpha(0) = \beta(0)$, and $h(\cdot,1) = \alpha(1) = \beta(1)$. If such a homotopy exists, we say that α and β are ***homotopic***, or in the same ***homotopy class***.

Graph embeddings. A ***surface*** (or more formally a *2-manifold*) $\mathcal{S}$ is a compact Hausdorff space in which every point has an open neighborhood homeomorphic to the plane. An ***embedding*** of a graph G on a surface $\mathcal{S}$ is a function mapping the vertices of G to distinct points in $\mathcal{S}$ and the edges of G to paths in $\mathcal{S}$ that are disjoint except at common endpoints. The ***faces*** of the embedding are maximal subsets of $\mathcal{S}$ that are disjoint from the image of the graph. An embedding is ***cellular*** if each face is homeomorphic to an open disk. A ***plane graph*** is an embedding of a graph in either the plane or the sphere.

Any cellular embedding in an orientable surface can be encoded combinatorially by a ***rotation system***, which records the counterclockwise order of edges incident to each vertex. Conversely, every rotation system for a graph G is consistent with a cellular embedding of G in some orientable surface; in fact, we can recover the faces of an embedding from its rotation system in linear time [20]. A rotation system of a graph $G = (V,E)$ is ***planar*** if it is consistent with a planar embedding, or equivalently (by Euler's formula) if it has exactly $2-|V|+|E|$ faces.

A ***walk*** in a graph $G = (V,E)$ is an alternating sequence of vertices and edges whose consecutive elements are incident; the ***endpoints*** of a walk are its initial and final vertices. A walk is called a ***path*** if no vertex appears more than once. Any embedding maps walks in G to walks in $\mathcal{S}$, and paths in G to paths in $\mathcal{S}$; two walks in an embedded graph ***cross*** if their images in the embedding cross.

2.2 Problem Formulation

We consider two different variants of the shortest non-crossing walks problem: a *combinatorial* formulation proposed by Takahashi *et al.* [27], and a *geometric* formulation considered by Takahashi *et al.* [28] and Papadopoulou [22].

In the geometric formulation, the input consists of h disjoint simple polygons $P_1, P_2, \ldots, P_h$ in the plane, called ***obstacles***, together with two disjoint sets $S = \{s_1, \ldots, s_k\}$ and $T = \{t_1, \ldots, t_k\}$ of points on the boundaries of the obstacles, called ***terminals***. Formally, we consider the obstacles P_i to be *open* sets. To simplify our presentation, we assume without loss of generality that each terminal is a vertex of some obstacle; let n denote the number of obstacle vertices. A ***set of ST-walks*** is a set of walks

$\Omega = \{\omega_1, \omega_2, \ldots, \omega_k\}$ in the free space $\mathcal{S} := \mathbb{R}^2 \setminus (P_1 \cup P_2 \cup \cdots \cup P_k)$, where each walk ω_i joins the corresponding pair of terminals s_i and t_i. To make the definition of crossing precise, we implicitly extend each walk ω_i infinitesimally into the obstacles at their endpoints. Our goal is either to compute a set of *non-crossing* ST-walks of minimum total length, or to report correctly that no set of ST-walks exists.

The combinatorial formulation is similar. The input consists of an n-vertex plane graph $G = (V, E)$; a weight function $w\colon E \to \mathbb{R}^+$; a subset $H = \{f_1, f_2, \ldots, f_h\}$ of faces of G, called ***obstacles***; and two disjoint sets of vertices $S = \{s_1, \ldots, s_k\}$ and $T = \{t_1, \ldots, t_k\}$, called ***terminals***, where each terminal is incident to a single obstacle face. A ***set of ST-walks*** is a set of walks $\Omega = \{\omega_1, \ldots, \omega_k\}$ in G, where each walk ω_i connects s_i and t_i. To make the definition of crossing precise, we implicitly extend each walk ω_i infinitesimally into the obstacles at their endpoints. Equivalently, we assume without loss of generality that each terminal has degree 1 and each walk ω_i is forbidden to visit terminals s_j or t_j except at its endpoints. It is convenient to think of the obstacles as holes in the plane. Again, our goal is to compute a set of *non-crossing* ST-walks in G of minimum total length, or to report correctly that no such walks exist.

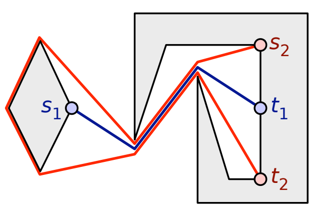

Figure 3. Shortest non-crossing walks.

When $h = 1$, shortest non-crossing ST-walks are actually *shortest paths* joining corresponding terminals. However, for any $h \geq 2$, there are inputs for which shortest non-crossing ST-walks must be non-simple. On the other hand, it is easy to prove that in any set of shortest non-crossing ST-walks, each walk is non-*self*-crossing.

2.3 Output Representation

Even when $h = 1$, the total complexity of the shortest non-crossing ST-walks is $\Omega(nk)$ in the worst case. To avoid worst-case quadratic running time, Takahashi *et al.* [27] and Papadopoulou [22] actually compute implicit representations of the shortest ST-paths of complexity $O(n)$. Our algorithms compute similar representations for all h, ultimately by reducing to the special case $h = 1$ and invoking the earlier algorithms [27, 22] as subroutines.

Both Takahashi *et al.* [27] and Papadopoulou [22] claim that their algorithms output a forest F, such that for each i, the unique path from s_i to t_i in F is the desired output path ω_i. However, this brief description is problematic in both settings. If the output walks are not paths, as in the example shown above, their union cannot be forest; moreover, Polishchuk and Mitchell [24] observed that the union of the output walks may contain cycles even when $h = 1$ and $k = 3$.

A close reading of Takahashi *et al.* [27] reveals that edges of G may appear multiple times in their output 'forest'; a single edge of G may appear in multiple trees, or even multiple times in the same tree. A more accurate description of their output structure is a ***non-crossing forest***: An abstract forest F, together with a homomorphism from F to G that maps edges to edges, leaves to terminal vertices, and terminal-to-terminal paths to non-crossing walks. A non-crossing forest represents a set of non-crossing ST-walks if and only if every walk ω_i is the image of some path in F; in particular, every pair of terminals s_i and t_i must lie in the image of the same component of F. The algorithm of Takahashi *et al.* for the single-obstacle problem ($h = 1$) computes a non-crossing forest with multiplicity at most two; that is, each edge of G is the image of at most two edges in F. For any constant h, our combinatorial algorithm computes a shortest non-crossing forest of complexity $O(n)$ that represents the shortest non-crossing ST-walks.

After some additional post-processing, an explicit representation of each walk can be extracted from the non-crossing forest in time proportional to its complexity; our reported running times suppress this output term. Specifically, to compute the walk ω_i from s_i to t_i, we first compute the path in F from s_i to t_i using fast least-common-ancestor queries [2]; then for each edge on the path in F in order, we report the corresponding edge of G. The time required to set up the least-common-ancestor data structure is dominated by the running time of our main algorithm.

Papadopoulou's algorithm for the geometric problem with one obstacle incorrectly assumes that the union of the shortest ST-paths is a forest [22, 24]; however, her algorithm can be modified to correctly compute a *geometric* non-crossing forest, in which the homomorphism maps edges of F to line segments in the free space $\mathcal{S}$. Alternately, one can adapt the output representation proposed by Polishchuk and Mitchell [24] for non-crossing *thick* paths to the original thin-path problem. Instead of a forest, their algorithm outputs the planar graph defined by the union of the shortest ST-paths, with additional information at the nodes that allow any shortest path ω_i to be extracted in time proportional to its complexity; we refer to Polishchuk and Mitchell [24] for further details.

2.4 Crossing and Cutting Non-Crossing Walks

Our results require reasoning carefully about crossings between different sets of non-crossing walks. To simplify our arguments, we implicitly treat any set of non-crossing walks as the limit of a sequence of well-behaved disjoint simple paths, which intersect the obstacles only at their endpoints.

Let $\mathcal{S}$ be a polygon with holes in the plane. A ***properly embedded arc*** in $\mathcal{S}$ is a simple path whose endpoints lie on the boundary of $\mathcal{S}$, and that is otherwise disjoint from the boundary. Following Cabello and Mohar [3], we let $\boldsymbol{\mathcal{S} ✁ \alpha}$ denote the surface obtained by ***cutting*** $\mathcal{S}$ along any properly embedded arc α; each point of α becomes a pair of boundary points in the new surface.[2] Topologically, $\mathcal{S} ✁ \alpha$ is the closure of $\mathcal{S} \setminus \alpha$; geometrically, $\mathcal{S} ✁ \alpha$ is a degenerate simple polygon with holes.

Now let ω be a non-self-crossing walk in $\mathcal{S}$ whose endpoints lie on $\partial\mathcal{S}$. We intuitively define $\mathcal{S} ✁ \omega$ as a space whose *topology* is consistent with cutting along a properly embedded arc close to ω, but whose *geometry* to be determined by ω itself. More formally, let $\tilde{\omega}$ be a properly embedded arc homotopic to ω, whose Hausdorff distance to ω is arbitrarily small. We define $\mathcal{S} ✁ \omega$ to be the topological space $\mathcal{S} ✁ \tilde{\omega}$ together with a continuous function $\phi : \mathcal{S} ✁ \tilde{\omega} \to \mathcal{S}$ that maps points on both copies of $\tilde{\omega}$ to the corresponding points in the original walk ω and is otherwise injective. The length of any walk ω' in $\mathcal{S} ✁ \tilde{\omega}$ is now defined to be the length of the projected walk $\phi(\omega')$ in the original space $\mathcal{S}$. At the risk of confusing the reader, we will use this formalism implicitly, without further comment, throughout the paper.

For the graph formulation, we implicitly work in the *combinatorial surface* model introduced by Colin de Verdière [8] and used by several other authors to formulate optimization problems for surface-embedded graphs [3, 4, 10, 9, 18]. For a simple path α in a plane graph G between two obstacle vertices, let $G ✁ \alpha$ denote the plane graph obtained by cutting G along α; each point of α becomes a pair of boundary points in $G ✁ \alpha$. If the endpoints of α lie on two different obstacles, those two faces are merged in $G ✁ \alpha$; otherwise, $G ✁ \alpha$ is disconnected. For a non-crossing walk ω between two vertices, $G ✁ \omega$ is obtained by duplicating the vertices and edges of ω with appropriate multiplicity.

Similarly, when we reason about crossing between different sets of walks, we implicitly perturb the walks into simple paths, so that every crossing becomes a single point of transverse intersection.

[2] We suggest the pronunciation "snip" for the symbol ✁.

3 The Decision Problem

In this section, we describe a linear-time algorithm to decide whether a given set of terminal pairs can be connected by *any* non-crossing walks. We describe our algorithm first for the combinatorial setting and then for the (easier) geometric setting.

Recall that in the combinatorial setting, our input consists of a plane graph G, together with $2k$ distinct vertices $s_1, t_2, \ldots, s_k, t_k$, each of degree 1. Call any face incident to a terminal vertex an *obstacle*. The obstacles and terminal pairs naturally define an undirected (multi-)graph C, called the *connection graph*, which has a node for each obstacle face f_i and k arcs $a_1, a_2, \ldots, a_k$, where each arc a_j joins the obstacles incident to the corresponding terminals s_j and t_j. The counterclockwise ordering of terminal vertices on each obstacle boundary defines a combinatorial embedding C_π of the connection graph.

Lemma 3.1. *Let $s_1, t_1, s_2, t_2, \ldots, s_k, t_k$ be vertices of degree 1 in a plane graph G, and let C_π the combinatorial embedding of their connection graph. G contains a set of non-crossing ST-walks if and only if C_π is a planar embedding.*

Proof: First suppose there are non-crossing walks $\omega_1, \omega_2, \ldots, \omega_k$ in G, where each walk ω_i connects terminals s_i and t_i. As discussed in Section 2, we can perturb these walks infinitesimally to obtain a set of disjoint simple paths $\tilde{\omega}_1, \tilde{\omega}_2, \ldots, \tilde{\omega}_k$ in the plane, where each path $\tilde{\omega}_j$ connects terminals s_j and t_j. Place a point v_i in the interior of each face f_i. For each obstacle face f_i, extend all the paths $\tilde{\omega}_j$ ending at a terminal incident to f_i to the point v_i. The extended paths define a planar geometric embedding of the connection graph C that is consistent with the combinatorial embedding C_π.

Conversely, suppose the combinatorial embedding C_π is planar. Fix an arbitrary sentinel point v_i inside each face f_i of G, including the outer face. Because C_π is planar, there is a geometric embedding of C that maps each node of C to the corresponding sentinel point and maps the arcs of C to disjoint simple paths $a_1, a_2, \ldots, a_k$ on the sphere S^2. Because the paths are disjoint, there is a disk δ_i of some small radius ε around each sentinel point v_i that intersects only the arcs ending at v_i. Within each disk δ_i, place a scaled copy $\tilde{f}_i$ of the face f_i around v_i. For each terminal vertex s_j (resp. t_j) on the boundary of f_i, let $\bar{s}_j$ (resp. $\bar{t}_j$) denote the corresponding point on the boundary of $\tilde{f}_i$. Because the arcs leaving v_i have the same counterclockwise order as the corresponding terminal vertices around f_i, we can continuously deform the arcs within each disk δ_i so that each arc a_j passes through the corresponding points $\bar{s}_j$ or $\bar{t}_j$ and their incident edges. By applying any continuous retraction from $S^2 \setminus (\tilde{f}_1 \cup \cdots \cup \tilde{f}_k)$ to G, we obtain a collection of non-crossing topological walks $\tilde{\omega}_1, \tilde{\omega}_2, \ldots, \tilde{\omega}_k$ in G connecting the terminal pairs. These are not walks in the graph-theoretic sense; they may double back many times in the interior of an edge. For each i, let ω_i be the graph-theoretic walk that visits the vertices of G in the same order as $\tilde{\omega}_i$. The walks $\omega_1, \omega_2, \ldots, \omega_k$ are homotopic to the non-crossing topological walks $\tilde{\omega}_1, \tilde{\omega}_2, \ldots, \tilde{\omega}_k$ and therefore do not cross. □

This lemma suggests a simple linear-time algorithm to determine if the terminal pairs can be connected by non-crossing walks in G. We compute the counterclockwise ordering of terminal vertices around each obstacle, by traversing each obstacle boundary once, after which it is easy to count the faces of C_π in $O(h+k) = O(n)$ additional time [20]. The connection graph C has h vertices and k edges, so by Euler's formula, the embedding C_π is planar if and only if it has exactly $k-h-2$ faces.

Theorem 3.2. *Let $s_1, t_1, s_2, t_2, \ldots, s_k, t_k$ be vertices of degree 1 in a plane graph G with n vertices. We can decide whether G contains a set of non-crossing ST-walks in $O(n)$ time.*

The algorithm and proof for the geometric setting are nearly identical. Here, the input consists of h disjoint closed polygonal obstacles $P_1, P_2, \ldots, P_k$ in the plane, of total complexity n, with $2k$ distinct terminal points $s_1, t_1, s_2, t_2, \ldots, s_k, t_k$ on their boundaries. The connection graph C has a node for each obstacle P_i and an arc for each terminal pair (s_j, t_j). The counterclockwise order of terminal points around each obstacle define a combinatorial embedding C_π. An easy modification of the proof of Lemma 3.1 implies that there is a set of non-crossing walks in $\mathbb{R}^2 \setminus (P_1 \cup \cdots \cup P_k)$ connecting the terminal pairs if and only if C_π is a planar embedding. (In fact, the proof is simpler.) Just as in the planar graph setting, we can construct C_π and determine whether it is planar in $O(n)$ time.

Theorem 3.3. *Let $s_1, t_1, s_2, t_2, \ldots, s_k, t_k$ be distinct terminal points on the boundary of h disjoint closed polygonal obstacles $P_1, P_2, \ldots, P_h$ of total complexity n in the plane. We can decide whether there is a set of non-crossing ST-walks in $\mathbb{R}^2 \setminus (P_1 \cup \cdots \cup P_k)$ in $O(n)$ time.*

4 Crossing Bounds

In this section, we prove that each walk in a minimum-length set of non-crossing walks crosses an arbitrary shortest path $2^{\Theta(h)}$ times in the worst case; this bound does not depend on the the number of terminal pairs (k) or the total complexity of the input (n).

Our upper bound proof (Section 4.1) uses an exchange argument, similar to arguments previously used to characterize shortest noncontractible and nonseparating cycles [3], shortest splitting cycles [4], and minimum cuts in surface-embedded graphs [5], as well as minimal realizations of string graphs (intersection graphs of simple curves in the plane) [21, 26]. We give an explicit upper bound proof only in the combinatorial setting, but our proof can be easily modified (in fact, simplified) to the geometric setting.

In particular, we use and refine an argument of Schaefer and Štefankovič [26, Theorem 3.2]. Let $G = (V, E)$ be an arbitrary graph, and let R be a set of pairs of edges of G. A drawing of G in the plane is a *weak realization* of the pair (G, R) if only pairs of edges in R are allowed (but not required) to cross in the drawing. Schaefer and Štefankovič prove that if the pair (G, R) has a weak realization, then it has a weak realization in which the total number of crossings along any edge is at most 2^m, where m is the number of edges in G. As we show below, their proof technique immediately implies that any walk in a set of shortest ST-walks crosses a shortest path at most 2^k times. However, further work is needed to reduce this crossing bound to a function of h (the number of obstacles).

Our lower bound proof (Section 4.2) uses an explicit construction inspired by a result of Hass *et al.* [13], but more similar in retrospect to an earlier construction of Kratchovíl and Matoušek [17].

4.1 Upper Bound

Fix an n-vertex plane graph $G = (V, E)$, a weight function $w\colon E \to \mathbb{R}_+$, and $2k$ distinct terminal vertices $s_1, t_1, \ldots, s_k, t_k \in V$, each with degree 1. In light of Theorems 3.2 and 3.3, we assume without loss of generality that there is a set of non-crossing ST-walks.

Fix a set $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_\ell\}$ of non-crossing shortest paths in G. Let $\Omega = \{\omega_1, \omega_2, \ldots, \omega_k\}$ be a set of non-crossing ST-walks in G that minimizes both the total length of the walks and the total number of crossings between walks ω_i and shortest paths σ_j. (In the geometric setting, minimizing length also minimizes the number of crossings, but the combinatorial setting is more subtle.) Our goal is to prove that each walk ω_i crosses each shortest path σ_j at most $2^{O(h)}$ times.

For each index j, the ***crossing sequence $X(\sigma_j, \Omega)$*** is a string over the alphabet $\{1, 2, \ldots, k\}$ that records the sequence of crossings between σ_j and walks in Ω, in order along σ_j. A substring is a contiguous sequence of symbols within a string. We call a substring of $X(\sigma_j, \Omega)$

even if any symbol appears an even number of times; for example, ELESSL is an even substring of the word SENSELESSLY.

The following key lemma follows directly from an argument of Schaefer and Štefankovič [26, Theorem 3.2].

Lemma 4.1. *For each j, the crossing sequence $X(\sigma_j, \Omega)$ contains no non-empty even substring.*

Proof: Because subpaths of shortest paths are themselves shortest paths, it suffices to prove that the entire crossing sequence $X(\sigma_j, \Omega)$ is not a non-empty even string. Suppose to the contrary that σ_j crosses each walk in Ω an even number of times, and crosses some walk in Ω at least once. We construct another set Ω' of non-crossing ST-walks that is no longer than Ω and has fewer crossings with Σ, contradicting the assumed optimality of Ω.

Following Schaefer and Štefankovič [26], consider a small 'window' W around σ_j that contains none of the obstacles. By an application of the Jordan-Schönflies theorem, we can assume without loss of generality that W is a long horizontal ellipse, σ_j is a horizontal line segment through the center of W, and for each walk ω_i, each component of $\omega_i \cap W$ is a vertical line segment. Removing these line segments partitions each walk ω_i into an odd number of subwalks, which we label alternately *even* and *odd*, with odd subwalks containing the endpoints of ω_i. To construct the new walk ω_i', we delete all subwalks within W, bring the even subwalks into W by a circular inversion, and then reflect the inverted subwalks across σ_j' to reconnect the odd subwalks. See Figure 4 for an example.

Because the original even subwalks did not cross outside W, their images inside W also do not cross. Thus, Ω' is indeed a set of non-crossing walks, with the same endpoints as Ω. Moreover, this surgery decreases the number of crossings on σ_j by at least a factor of 2, and it does not increase the number of crossings on any other path in Σ. Because W is homeomorphic to a disk and does not contain any obstacles, we can replace each transformed subwalk within W with a line segment without introducing any crossings. If we shrink the height of the ellipse W to zero, then in the limit, each transformed subwalk becomes a subpath of σ_j, and therefore a shortest path. Thus, the total length of Ω' is not greater than the total length of Ω. □

The next lemma now immediately implies that each crossing sequence $X(\sigma_j, \Omega)$ has length at most 2^k.

Lemma 4.2 ([26, Lemma 3.1]). *Any string of length at least 2^k with at most k distinct characters has a non-empty even substring.*

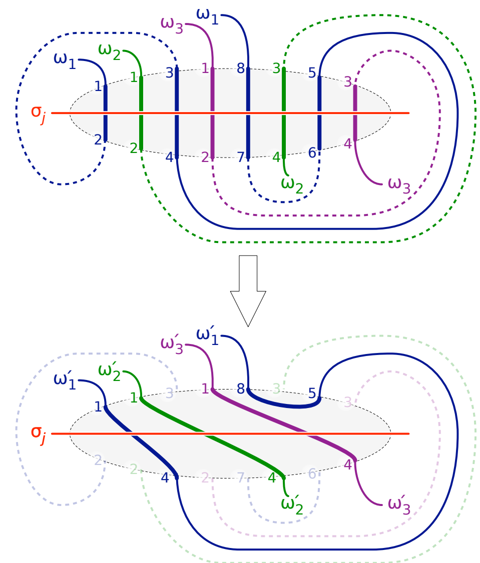

Figure 4. Shortening walks with an even crossing sequence, after Schaefer and Štefankovič [26]. Even subwalks are indicated by dashed lines.

Proof: Let A be a string of length $n = 2^k$ over the alphabet $\{x_1, \ldots, x_k\}$. We define a sequence of k-bit strings $B_1, B_2, \ldots, B_n$ as follows: $B_i[j] = 1$ if and only if x_j appears an *odd* number of times in the prefix $A[1\,..\,i]$, and 0 otherwise. If some string B_i is all zeros, the substring $A[1\,..\,i]$ is even. Otherwise, the pigeonhole principle implies that strings B_x and B_y are equal for some $x < y$, and the substring $A[x+1\,..\,y]$ is even. □

We improve this crossing bound by considering each walk in Ω individually and essentially bundling portions of the other $k-1$ walks into a small number of homotopy classes [11]. For each walk ω_i and each shortest path σ_j, we define an ***overlay graph H_{ij}***, whose vertices are the crossing points of ω_i and σ_j, and whose edges are the subwalks of ω_i and σ_j between consecutive crossing points. To simplify our following discussion, we color each edge of H_{ij} *blue* if it is a subwalk of ω_i and *red* if it is a subpath of σ_j. The graph H_{ij} has a natural planar embedding, and therefore a well-defined dual graph H_{ij}^*.

Each face of this embedding has an even number of sides, which alternate between red and blue. We call a face of H_{ij} ***empty*** if it does not contain any of the h obstacle faces and ***non-empty*** otherwise; clearly there are at most h non-empty faces. A face of H_{ij} is called a ***bigon*** if it has exactly two boundary edges, and a ***quadrilateral*** if it has exactly four boundary edges. The edges bounding any bigon or quadrilateral must alternate between red and blue. The following lemma mirrors a result of Pach and Tóth [21, Lemma 2.1] in the context of string graphs.

Lemma 4.3. *No bigon in H_{ij} is empty.*

Proof: Suppose H_{ij} has an empty bigon B, whose boundary is composed of a blue edge $b \subset \omega_i$ and a red edge $r \subset \sigma_j$. Every other walk in Ω that intersects B must cross r an even number of times, but cannot cross b. For each walk $\omega_x \in \Omega$, we define a new walk ω'_x by replacing any subwalk of ω_x inside B with the corresponding subpath of r. In particular, ω'_i is defined by replacing b with r in ω_i. See Figure 5.

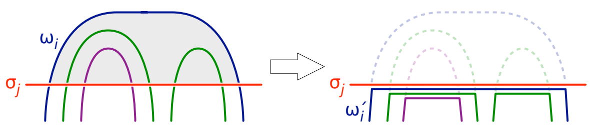

Figure 5. Removing an empty bigon.

Let $\Omega' = \{\omega'_1, \ldots, \omega'_k\}$. Because σ_j is a shortest path, each modified walk ω'_x is no longer than the original walk ω_x. Moreover, the walks in Ω' cross the shortest paths in Σ fewer times than Ω. To complete the proof, it remains only to show that the modified walks in Ω' do not cross each other.

Suppose two modified walks ω'_x and ω'_y cross, then they must cross at a subpath of r. That is, there must be subpaths $\pi'_x \subseteq \omega'_x \cap b$ and $\pi'_y \subseteq \omega'_y \cap b$ whose endpoints alternate along the red path r. But then the Jordan curve theorem implies that walks ω_x and ω_y must cross within B, which is impossible. $\square$

Let T_{ij} denote the subgraph of blue edges of H_{ij}, and let C_{ij} denote the subgraph of red edges. The subgraph T_{ij} is actually a spanning tree of H_{ij}; thus, the dual subgraph C^*_{ij} is a spanning tree of the dual graph H^*_{ij}. In other words, the pair (T_{ij}, C_{ij}) is a *tree-cotree decomposition* of H_{ij} [12]. Following Schaefer *et al.* [25], we call a vertex of C^*_{ij} ***good*** if the corresponding face of H_{ij} is an empty quadrilateral, and ***bad*** otherwise. The following lemma slightly improves a result of Schaefer *et al.* [25, Lemma 2.2].

Lemma 4.4. *The total degree of the bad vertices of C^*_{ij} is at most $4h - 4$.*

Proof: Let ℓ denote the number of leaves in C^*_{ij}. Each leaf of C^*_{ij} corresponds to a bigon in H_{ij}; thus, Lemma 4.3 implies that $\ell \le h$. It follows that at most $h - \ell$ bad vertices have degree 2. These vertices correspond to non-empty quadrilaterals and their total degree is at most $2h - 2\ell$.

Now smooth out the degree-2 vertices in C^*_{ij}, by replacing any path through degree-2 vertices with a single edge. The vertices of the resulting tree are the bad vertices whose degrees we have not already counted. Because this tree has ℓ leaves, it has at most $2\ell - 1$ vertices, and therefore has at most $2\ell - 2$ edges. Thus, the total degree of these vertices is at most $4\ell - 4$.

We conclude that the total degree of the bad vertices is at most $2h + 2\ell - 4 \le 4h - 4$. $\square$

The good vertices in C^*_{ij} induce a collection of paths in the dual; a good vertex has degree 2. We call the sequence of quadrilateral faces dual to each induced path a ***street***.[3] Because no street contains an obstacle, any walk in Ω that intersects a street must enter at one end, traverse the entire street, and exit at the other end; otherwise, it would either cross ω_i or define an empty bigon with σ_j.

We associate a unique label with each street, and then extend the street labeling to a labeling of the edges of C_{ij} (that is, the subpaths of σ_j) as follows. If an edge in C_{ij} intersects a street, either as one of the street's ends or by crossing through its interior, the edge inherits that street's label. Any edge that is adjacent to only bad faces is assigned a special label #. The edge labeling is well-defined, because no edge of C_{ij} is adjacent to more than one street. All edges of C_{ij} with the same label cross the same walks in Ω in the same order (up to reversal). We call the sequence of edge labels along σ_j the ***street sequence*** S_{ij}.

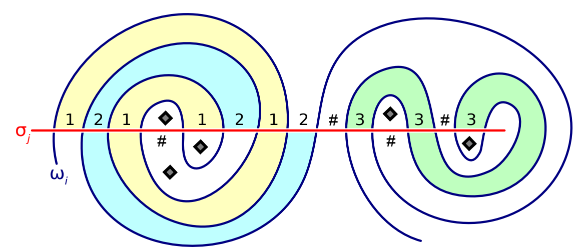

Figure 6. Three streets (shaded) defining the street sequence 121#1212#3#3#3. Black diamonds indicate obstacles.

Theorem 4.5. *Each walk ω_i crosses each shortest path σ_j at most 2^{2h-2} times.*

Proof: Let s denote the number of streets in the overlay graph H_{ij}, and let x denote the number of times the symbol # occurs in the street sequence S_{ij}. We claim that each walk ω_i crosses each shortest path σ_j at most $(x+1)2^s$ times.

For the sake of argument, suppose some walk $\omega_i \in \Omega$ crosses some shortest path $\sigma_j \in \Sigma$ more than $(x+1)2^s$ times. Then the street sequence S_{ij} has length greater than $(x+1)2^s$ and therefore contains a substring S' of

[3] Pach and Tóth [21] call this sequence of faces an *empty (e, f)-path of four-cells*.

length 2^s that avoids the symbol #. Lemma 4.2 implies that S' contains a nonempty even substring S''. Let σ'' be the subpath of σ_j that corresponds to S''. This subpath starts and ends at crossings with ω_i and crosses ω_i an odd number of times. Because any subwalk that enters a street at one end must exit at the other end, σ'' crosses any walk ω_x with $x \neq i$ an even number of times. Therefore, if we remove the last symbol of the crossing sequence $X(\sigma'', \Omega)$, we obtain a non-empty even substring of the crossing sequence $X(\sigma_j, \Omega)$, contradicting Lemma 4.1.

It remains only to prove an upper bound for the quantity $(x+1)2^s$. Every street in H_{ij} starts and ends at a red edge whose dual in C^*_{ij} is incident to exactly one bad vertex, and each occurrence of the symbol # in S_{ij} corresponds to an edge of between two bad vertices C^*_{ij}. Thus, Lemma 4.4 implies that $2s + 2x \leq 4h - 4$. We conclude that $(x+1)2^s \leq (x+1)2^{2h-2-x} \leq 2^{2h-2}$. □

4.2 Lower Bound

In this section, we prove by construction that shortest non-crossing walks can cross a shortest path $2^{\Omega(h)}$ times; thus, the total complexity of shortest non-crossing walks is exponential in h in the worst case. Our construction was inspired by the construction by Hass *et al.* [13] of an unknotted polygonal cycle in $\mathbb{R}^3$ such that any piecewise-linear spanning disk has exponential complexity. In retrospect, our example also resembles a construction of Kratchovíl and Matousek [17] of a graph $G = (V, E)$ and a set R of edge pairs, such that any weak representation of (G, R) has an exponential number of crossings.

Fix a positive integer n. Let G be a graph with vertices $\{s_1, t_1, \ldots, s_n, t_n, u, v, w\}$, with edges between v and every other vertex and a loop edge ℓ_i at each vertex s_i. We embed G in the plane so that the counterclockwise order of neighbors around v is $s_1, u, s_2, t_1, s_3, t_2, \ldots, t_{n-2}, s_n, t_{n-1}, t_n, w$, as shown in Figure 7. The loops ℓ_i enclose the obstacle faces. We weight the edges by setting $w(\ell_i) := 2^{in}$ for each i, setting $w(uv) = w(vw) = \infty$, and setting $w(e) = 0$ for every other edge e.

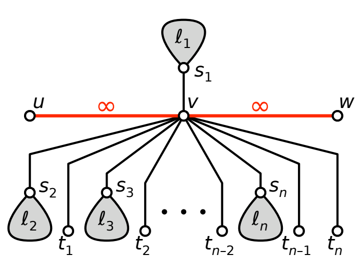

Figure 7. A plane graph in which the shortest non-crossing ST-walks have exponential complexity.

We inductively construct a set of ***canonical*** non-crossing ST-walks $\Omega^* = \{\omega^*_1, \omega^*_2, \ldots, \omega^*_n\}$ in G as follows. We first define a sequence $\alpha_1, \alpha_2, \ldots, \alpha_k$ of closed walks starting and ending at v. Specifically, we define α_1 to be the empty walk, and for each $i \geq 2$, we define

$$\alpha_i := rev(\alpha_{i-1}) \cdot (v, s_{i-1}) \cdot \ell_{i-1} \cdot (s_{i-1}, v) \cdot \alpha_{i-1}.$$

where $\cdot$ denotes the concatenation operator. Finally, for each i, we define $\omega^*_i := (s_i, v) \cdot \alpha_i \cdot (v, t_i)$. Our embedding ensures that the walks in Ω^* do not cross. Each walk ω^*_j traverses the loop ℓ_i exactly 2^{j-i-1} times if $i < j$, and does not traverse ℓ_i at all if $i \geq j$, so each loop ℓ_i is traversed $2^{n-i} - 1$ times altogether. Each walk ω^*_j crosses the shortest path σ from u to w exactly 2^{j-1} times; thus, σ is crossed $2^n - 1$ times altogether.

The following lemma implies that Ω^* is the unique minimum-length set of non-crossing walks connecting the terminals in G.

Lemma 4.6. *Let $\Omega = \{\omega_1, \omega_2, \ldots, \omega_n\}$ be a minimum-length set of non-crossing walks in G, such that each walk ω_i connects terminals s_i and t_i. For all i and j, walk ω_j traverses loop ℓ_i exactly $\lfloor 2^{j-i-1} \rfloor$ times.*

Proof: We prove the lemma by backward induction on i. The base case $i = n$ is trivial. The loop ℓ_n cannot be used in any optimal solution because it is longer than the total length of the canonical solution. Assume inductively that ℓ_{i+1} is traversed exactly $\lfloor 2^{j-i-2} \rfloor$ times by each ω_j.

Let δ_i be a path in the plane from t_i to s_i that crosses ℓ_i, but no other edges of G, so that ℓ_{i+1} lies in the interior of the cycle $vt_i \cdot \delta_i \cdot s_i v$. (See Figure 8.) Let ρ_i denote the closed walk $\omega_i \cdot \delta_i$. The induction hypothesis implies that ω_i does not traverse the loop ℓ_{i+1}; thus, ℓ_{i+1} lies completely inside ρ_i. On the other hand, for all $j > i+1$, the loop ℓ_j lies completely outside ρ_i.

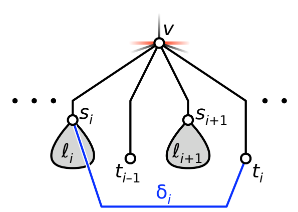

Figure 8. Defining the path δ_i.

Walk ω_{i+1} starts inside ρ_i and ends outside ρ_i, it must cross ℓ_i at least once. The Jordan Curve Theorem implies that the only way to cross ρ_i without crossing ω_i is by traversing ℓ_i. Thus, ω_{i+1} traverses ℓ_i at least once.

Fix an index $j > i+1$. The induction hypothesis implies that ω_j traverses ℓ_{i+1} at least 2^{j-i-2} times, and therefore must enter and then exit ρ_i at least 2^{j-i-2} times. It follows that ω_j must traverse ℓ_i twice for each traversal of ℓ_{i+1}, and therefore at least 2^{j-i-1} times altogether. We conclude that ℓ_i is traversed at least $2^{n-i}-1$ times by Ω.

Recall that each loop ℓ_j is traversed exactly $2^{n-j}-1$ times by the canonical walks Ω^*. Thus, the total length of all canonical traversals of loops $\ell_1, \ell_2, \ldots, \ell_{i-1}$ is

$$\sum_{j=1}^{i-1} 2^{nj}(2^{n-j}-1) \;<\; 2^n \sum_{j=1}^{i-1} (2^{n-1})^j \;<\; 2^{ni}.$$

Thus, if any walk ω_j traversed loop ℓ_i more than $\lfloor 2^{j-i-1} \rfloor$ times, Ω would have larger total length than the canonical walks Ω^* and would therefore not be optimal. We conclude that ω_j traverses loop ℓ_i exactly $\lfloor 2^{j-i-1} \rfloor$ times, as required. □

We can realize our lower bound example geometrically as follows. The terminals are evenly spaced points on a tiny circle, in cyclic order $s_1, s_2, t_1, s_3, t_2, \ldots, s_n, t_{n-1}, t_n$. Each terminal is at one end of a line segment (or a very skinny triangle) pointing directly away form the center of the circle; these segments are the obstacles. For each i, the segment attached to s_i has length 2^{in}, and the segment attached to t_i has infinite length. Figure 9 shows the canonical solution for $n = 3$.

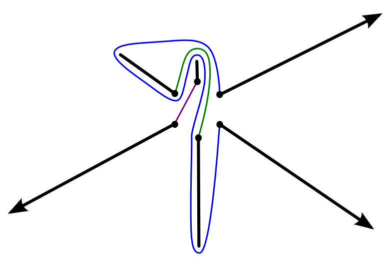

Figure 9. A geometric version of our exponential lower bound.

5 Planar Graph Algorithm

Now we describe our algorithm for the combinatorial version of the shortest non-crossing walks problem. As in the previous section, the input consists of an n-vertex plane graph $G = (V, E)$ with weighted edges, and two disjoint sets $S = \{s_1, s_2, \ldots, s_k\}$ and $T = \{t_1, t_2, \ldots, t_k\}$ of terminal vertices, each with degree 1. Let h denote the number of obstacles. Again, we assume that the input graph G contains at least one set of non-crossing ST-walks.

Our algorithm ultimately reduces to a special case already considered by Takahashi *et al.* [27], where for each i, the terminals s_i and t_i lie on the same obstacle (although different terminal pairs may lie on different obstacles). In this case, shortest ST-walks consist of non-crossing *shortest paths* joining the terminals. These shortest paths can be computed using the following naive algorithm: For each i from 1 to k, compute the shortest path σ_i in G connecting s_i and t_i, and then replace G with $G \not\mathrel{\&} \sigma_i$. Takahashi *et al.* describe a divide-and-conquer algorithm (called 'PATH2') to compute a non-crossing forest containing all the shortest paths between terminals on a *single* face in $O(n \log k)$ time.

Lemma 5.1. *Shortest non-crossing ST-walks in an n-vertex planar graph with k terminal pairs and h obstacles can be computed in $O(hn \log k)$ time, if for every index i, terminals s_i and t_i lie on the same obstacle.*

To solve the general problem, we adapt the approach of Takahashi *et al.* [27] for the special case $h = 2$; similar strategies have been used to find various optimal topologically interesting cycles in combinatorial surfaces [3, 4, 5, 18].

5.1 Spanning Walks

Recall from Section 3 that the obstacles and terminal pairs naturally define a *connection graph* C whose nodes correspond to the obstacles f_i and whose arcs correspond to the terminal pairs (s_j, t_j). Let F be an arbitrary maximal spanning forest of the connection graph. Without loss of generality, we can assume its edges correspond to the first m terminal pairs $(s_1, t_1), \ldots, (s_m, t_m)$; let $S' = \{s_1, \ldots, s_m\}$ and $T' = \{t_1, \ldots, t_m\}$. Note that $m \le h - 1$.

Say that a walk is ***tight*** if it is a shortest walk in its homotopy class. Results of Colin de Verdière and Lazarus [8, 10, 9] imply that in any minimum-length set of non-crossing ST-walks, every walk is tight; otherwise, we could make at least one walk shorter without introducing any crossings. Conversely, suppose Ω' is a set of tight non-crossing $S'T'$-walks; let $\tilde{\Omega}$ be a set of non-crossing ST-walks that includes Ω'; and let Ω be a set of ST-walks homotopic to $\tilde{\Omega}$. Then either the walks Ω are non-crossing, or there is a bigon whose removal decreases the total number of crossings, exactly as in Lemma 4.3.

Thus, given a set Ω' of tight non-crossing $S'T'$-walks each *in the correct homotopy class*, there is a minimum-length set Ω of non-crossing ST-walks that includes Ω'. Moreover, we can compute Ω by running the same-obstacle algorithm on the graph $G \not\mathrel{\&} \Omega'$, which has exactly one obstacle for each connected component of the connection graph C.

5.2 Enumerating Homotopy Classes

We enumerate all homotopy classes of non-crossing $S'T'$-walks that satisfy the crossing bound in Theorem 4.5 as follows. We first compute a set $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_{h-1}\}$ of non-crossing shortest paths that connect the obstacle faces $f_0, f_1, \ldots, f_{h-1}$. Specifically, we compute a shortest-path tree rooted at an arbitrary vertex v_0 on obstacle f_0 in $O(n)$ time [14], and then for each i, we define σ_i to be the shortest path from v_0 to any vertex on f_i that is not a terminal vertex. (This procedure may require subdividing some edges in G.)

Let $G \mathbin{\text{✂}} \Sigma$ denote the planar graph obtained by cutting G along every shortest path $\sigma_i \in \Sigma$. Following Chambers *et al.* [4], we represent $G \mathbin{\text{✂}} \Sigma$ compactly as an *abstract polygonal schema* Π, which is a convex polygon with $2h + 2m - 2 = O(h)$ vertices: $2h - 2$ *path vertices* corresponding to the copies of each shortest path σ_i in $G \mathbin{\text{✂}} \Sigma$, plus the $2m$ terminals $S' \cup T'$. The boundary edges of Π correspond to subpaths of the obstacle boundaries.

Call a set Ω' of non-crossing $S'T'$-walks *bigon free* if no walk in Ω' defines an empty bigon with any path in Σ; see Lemma 4.3. Any bigon-free set Ω' of non-crossing $S'T'$-walks in G can be represented by a weighted triangulation of Π whose edges correspond to certain subwalks of Ω'. Specifically, an edge between two path vertices represents a subwalk that consecutively crosses the corresponding pair of shortest paths in Σ; an edge between two terminals represents a walk between those terminals that does not cross any path in Σ; and an edge between a terminal and a path vertex represents a subwalk that starts at the terminal and immediately crosses the corresponding shortest path. The weight of each diagonal is the number of corresponding subwalks appearing in Ω'; if the walks in Ω' satisfy Theorem 4.5, then each diagonal has weight at most $2^{O(h)}$.

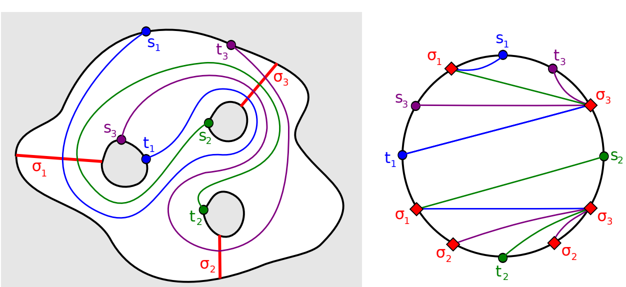

Figure 10. Representing non-crossing walks with an abstract polygonal schema.

Conversely, a weighted triangulation corresponds to a bigon-free set Ω' of non-crossing $S'T'$-walks if both vertices corresponding to any shortest path σ_i are incident to diagonals of equal total weight, and each terminal is incident to exactly one diagonal with weight 1. We call such a weighted triangulation *valid* if in addition every diagonal has weight at most $2^{O(h)}$. The polygon Π supports $2^{O(h)}$ unweighted triangulations, and therefore $2^{O(h^2)}$ valid weighted triangulations, which we can enumerate in $2^{O(h^2)}$ time.

5.3 Tight Spanning Walks

For each valid weighted triangulation Δ, we compute a corresponding collection of tight non-crossing $S'T'$-walks by adapting an algorithm of Kutz [18]. The *crossing sequences* of a bigon-free walk ω is the sequence of shortest paths in Σ that ω crosses, in order along the walk. Two bigon-free walks with the same endpoints have the same crossing sequences if and only if they are homotopic. We can easily extract the crossing sequences $X_1, X_2, \ldots, X_m$ of the m walks represented by Δ in $2^{O(h)}$ time, by brute force. For each index i, let $x_i \le 2^{O(h)}$ denote the length of crossing sequence X_i.

We can compute a shortest walk with a given crossing sequence X_i as follows. First, glue together x_i copies of $G \mathbin{\text{✂}} \Sigma$ along the copies of the shortest paths that ω crosses, to obtain a planar graph $\hat{G}$ of complexity $O(x_i n)$. Then compute a shortest path $\hat{\omega}_i$ in $\hat{G}$ between s_i in the initial copy of $G \mathbin{\text{✂}} \Sigma$ and t_i in the final copy of $G \mathbin{\text{✂}} \Sigma$, using the linear-time shortest path algorithm of Henzinger *et al.* [14]. Finally, project the path $\hat{\omega}_i$ back into G to obtain the walk ω_i.

Intuitively, we would like to run this shortest-walk algorithm independently for each crossing sequence X_i, but there is no guarantee that the resulting walks would not cross. Instead, we use a variant of the naive algorithm suggested by Takahashi *et al.* for the same obstacle case. Initially, let $H = G \mathbin{\text{✂}} \Sigma$. For each index i from 1 to m, compute the shortest walk ω_i in G with crossing sequence X_i by gluing together copies of H, replace G with $G \mathbin{\text{✂}} \omega_i$, and replace H with $H \mathbin{\text{✂}} \omega_i$. After the first iteration, the graph H may be disconnected, but it is easy to adapt the gluing algorithm to only glue together copies of the relevant components of $G \mathbin{\text{✂}} \Sigma$ to obtain the graph $\hat{G}$. Each iteration of this process increases the complexity of the graphs G and H by at most $2^{O(h)}n$. Thus, for each valid weighted triangulation Δ, we construct a minimum-length set Ω' of non-crossing $S'T'$-walks consistent with Δ in $2^{O(h)}n$ time.

5.4 Summing Up

Our algorithm spends $O(n)$ time computing the shortest paths Σ and constructing the abstract polygonal schema Π. For each of the $2^{O(h^2)}$ valid weighted triangulations Δ of Π, we compute a set Ω' of tight non-crossing walks consistent with Δ in time $2^{O(h)}n$. The graph $G \mathbin{\text{✂}} \Omega'$ has complexity at most $2^{O(h)}n$; thus, we can extend Ω' to a set of tight non-crossing ST-walks in time $O(2^{O(h)} n \log k)$ using Lemma 5.1. We conclude:

Theorem 5.2. *Shortest non-crossing ST-walks in an n-vertex planar graph with k terminal pairs and h obstacles can be computed in $2^{O(h^2)} n \log k$ time and $2^{O(h)} n$ space.*

6 Geometric Algorithm

Now we describe the geometric version of our shortest non-crossing ST-walk algorithm. The input consists of h disjoint simple polygonal obstacles $P_1, P_2, \ldots, P_h$ in the plane with total complexity n, along with two disjoint sets $S = \{s_1, \ldots, s_k\}$ and $T = \{t_1, \ldots, t_k\}$ of obstacle vertices. Our goal is to find a minimum-length set of non-crossing ST-walks in $\mathbb{R}^2 \setminus (P_1 \cup \cdots \cup P_h)$; we can clearly restrict our search to the smaller work space $W = \square \setminus (P_1 \cup \cdots \cup P_h)$, where $\square$ is a large rectangle containing all the obstacles. To simplify the algorithm, we assume the polygons are in general position, so there is a unique shortest path between any two vertices.

6.1 Same Obstacle Case

Consider the special case where each pair of matching terminals s_i and t_i lies on the same obstacle. In this case, the optimal set of non-crossing ST-walks consists of the *unique* globally shortest paths between the terminal pairs, which are unique because the polygons are in general position. These paths can be computed one at a time in $O(kn \log n)$ time using the shortest-path algorithm of Hershberger and Suri [16]. Here we describe an algorithm that runs in $O(n)$ time when h is constant, generalizing an algorithm of Papadopoulou [22] for the special case $h = 2$. The main difficulty is determining the homotopy class of each of the k shortest paths.

We first construct a set of disjoint line segments $\Sigma = \{\sigma_1, \ldots, \sigma_h\}$, where for each index i, σ_i is the vertical segment from the lowest vertex of P_i to the boundary of the next lower obstacle P_j or the bounding box $\square$. We can compute these segments in $O(hn)$ time by brute force. The space $W' = W \mathbin{\text{✂}} \Sigma$ is a topological disk with complexity $O(n)$, which we can compute in $O(n)$ time from Σ. We can compute a triangulation W' in $O(n)$ time [7].

We observe that the shortest path between any two points in W crosses each segment σ_i at most once. Our algorithm now considers all homotopy classes of walks that satisfy this crossing condition. There are $O(h!)$ valid crossing sequences, which we can enumerate in $O(h!)$ time.

For each crossing sequence X of length x, we glue together x copies of the disk W' along the crossed segments, to obtain a larger topological disk W^X of complexity $O(h!\, n)$. The disk W^X is not a simple polygon, but a *boundary-triangulated 2-manifold* [15], whose triangulation is inherited from the triangulation of W'. We compute the shortest paths in W^X from each terminal s_i in the first copy of W' to the corresponding terminal t_i in the last copy of W', using the linear-time single-obstacle algorithm of Papadopoulou [22]. Papadopoulou describes her algorithm only for simple polygons, but it actually works for arbitrary boundary-triangulated 2-manifolds, as it ultimately relies only on the standard funnel algorithm for computing shortest paths [6, 19, 15]. The output of Papadopoulou's algorithm is a geometric non-crossing forest F_X of complexity $O(h!\, n)$ containing the required shortest non-crossing paths in W^X.

We now have $O(h!)$ geometric non-crossing forests F_X, one for each crossing sequence X. For each index i, we can determine which forest F_X contains the shortest path from s_i to t_i. For each forest F_X, we extract the subforest F'_X containing the shortest ST-walks that have crossing sequence X. Finally, we return the union of all geometric non-crossing forests F'_X.

Lemma 6.1. *Shortest non-crossing ST-walks in the complement of h polygonal obstacles with total complexity n can be computed in $h^{O(h)} n$ time, if for every index i, terminals s_i and t_i lie on the same obstacle.*

6.2 General Case

Our solution strategy for the general case is the same as in the graph setting. As in the same-obstacle case, we construct a set $\Sigma = \{\sigma_1, \ldots, \sigma_h\}$ of vertical line segments that cut W into a topological disk, in $O(hn)$ time. Let F be an arbitrary maximal spanning forest of the connection graph of the terminals; assume that the edges of F join terminals $S' = \{s_1, \ldots, s_m\}$ and $T' = \{t_1, \ldots, t_m\}$. We enumerate all homotopy classes of tight $S'T'$-walks that cross each segment σ_j at most $2^{O(h)}$ times using weighted triangulations. For each of the $2^{O(h^2)}$ valid weighted triangulations Δ, we compute a set Ω' of tight non-crossing walks consistent with Δ in $2^{O(h)} n$ time, using the homotopic shortest path algorithm of Hershberger and Snoeyink [15] in place of the planar graph algorithm of Henzinger *et al.* [14]. Finally, we extend Ω' to a set of tight non-crossing ST-walks using the same-obstacle algorithm in the space $W \mathbin{\text{✂}} \Omega'$.

Theorem 6.2. *Shortest non-crossing ST-walks in the complement of h polygonal obstacles with total complexity n can be computed in $2^{O(h^2)} n$ time and $2^{O(h)} n$ space.*

Acknowledgments. Thanks to the anonymous reviewers of two different versions of this paper for their helpful feedback.

References

[1] O. Bastert and S. P. Fekete. Geometric wire routing. Technical Report 98-332, Zentrum für Angewandte Informatik, Univ. Köln, 1998. Cited by [24].

[2] M. A. Bender and M. Farach-Colton. The LCA problem revisited. *Proc 4th Latin American Symp. Theoret. Informatics (LATIN)*, 88–94, 2000. Lecture Notes in Computer Science 1776, Springer-Verlag.

[3] S. Cabello and B. Mohar. Finding shortest non-separating and non-contractible cycles for topologically embedded graphs. *Discrete Comput. Geom.* 37:213–235, 2007.

[4] E. W. Chambers, É. Colin de Verdière, J. Erickson, F. Lazarus, and K. Whittlesey. Splitting (complicated) surfaces is hard. *Comput. Geom. Theory Appl.* 41(1–2):94–110, 2008.

[5] E. W. Chambers, J. Erickson, and A. Nayyeri. Minimum cuts and shortest homologous cycles. *Proc. 25th Ann. ACM Symp. Comput. Geom.*, 377–385, 2009.

[6] B. Chazelle. A theorem on polygon cutting with applications. *Proc. 23rd Ann. Symp. Foundations Comput. Sci.*, 339–349, 1982.

[7] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete Comput. Geom.* 6(5):485–524, 1991.

[8] É. Colin de Verdière. *Raccourcissement de courbes et décomposition de surfaces [Shortening of Curves and Decomposition of Surfaces]*. Ph.D. thesis, University of Paris 7, Dec. 2003. ⟨http://www.di.ens.fr/~colin/textes/these.html⟩.

[9] É. Colin de Verdière and F. Lazarus. Optimal system of loops on an orientable surface. *Discrete Comput. Geom.* 33(3):507–534, 2005.

[10] É. Colin de Verdière and F. Lazarus. Optimal pants decompositions and shortest homotopic cycles on an orientable surface. *J. ACM* 54(4), 2007.

[11] A. Efrat, S. G. Kobourov, and A. Lubiw. Computing homotopic shortest paths efficiently. *Comput. Geom. Theory Appl.* 35(3):162–172, 2006.

[12] D. Eppstein. Dynamic generators of topologically embedded graphs. *Proc. 14th Ann. ACM-SIAM Symp. Discrete Algorithms*, 599–608, 2003.

[13] J. Hass, J. Snoeyink, and W. P. Thurston. The size of spanning disks for polygonal curves. *Discrete Comput. Geom.* 29(1):1–17, 2003.

[14] M. R. Henzinger, P. Klein, S. Rao, and S. Subramanian. Faster shortest-path algorithms for planar graphs. *J. Comput. Syst. Sci.* 55(1):3–23, 1997.

[15] J. Hershberger and J. Snoeyink. Computing minimum length paths of a given homotopy class. *Comput. Geom. Theory Appl.* 4(2):63–97, 1994.

[16] J. Hershberger and S. Suri. An optimal algorithm for Euclidean shortest paths in the plane. *SIAM J. Comput.* 28(6):2215–2256, 1999.

[17] J. Kratochvíl and J. Matoušek. String graphs requiring exponential representations. *J. Comb. Theory Ser. B* 53(1):1–4, 1991.

[18] M. Kutz. Computing shortest non-trivial cycles on orientable surfaces of bounded genus in almost linear time. *Proc. 22nd Ann. ACM Symp. Comput. Geom.*, 430–438, 2006.

[19] D.-T. Lee and F. P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks* 14:393–410, 1984.

[20] B. Mohar and C. Thomassen. *Graphs on Surfaces*. Johns Hopkins University Press, 2001.

[21] J. Pach and G. Tóth. Recognizing string graphs is decidable. *Discrete Comput. Geom.* 28:593–606, 2002.

[22] E. Papadopoulou. k-pairs non-crossing shortest paths in a simple polygon. *Proc. 7th Int. Symp. Algorithms Comput.*, 305–314, 1996. Lecture Notes Comput. Sci. 1178, Springer-Verlag.

[23] V. Polishchuk. *Thick non-crossing paths and minimum-cost continuous flows in geometric domains*. Ph.D. thesis, Stony Brook Univ., 2007. ⟨http://hdl.handle.net/1951/44607⟩.

[24] V. Polishchuk and J. S. Mitchell. Thick non-crossing paths and minimum-cost flows in polygonal domains. *Proc. 23rd Ann. Symp. Comput. Geom.*, 56–65, 2007.

[25] M. Schaefer, E. Sedgwick, and D. Štefankovič. Spiraling and folding: The word view. *Algorithmica* , in press, 2009. ⟨http://dx.doi.org/10.1007/s00453-009-9362-8⟩.

[26] M. Schaefer and D. Štefankovič. Decidability of string graphs. *J. Comput. Syst. Sci.* 68(2):319–334, 2004.

[27] J. Takahashi, H. Suzuki, and T. Nishizeki. Algorithms for finding non-crossing paths with minimum total length in plane graphs. *Proc. 3rd Int. Symp. Algorithms Comput.*, 400–409, 1992. Lecture Notes Comput. Sci. 650, Springer-Verlag.

[28] J. Takahashi, H. Suzuki, and T. Nishizeki. Finding shortest non-crossing rectilinear paths in plane regions. *Proc. 4th Int. Symp. Algorithms Comput.*, 98–107, 1993. Lecture Notes Comput. Sci. 762, Springer-Verlag.