Shortest Non-Crossing Walks in the Plane

Jeff Erickson[*] Amir Nayyeri[*]

Submitted to SOCG 2010 — December 3, 2009

Abstract

Let G be a 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 Nikizeki [*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.

[*]Department of Computer Science, University of Illinois, Urbana-Champaign; {jeffe,nayyeri2}@illinois.edu. Research partially supported by NSF grant CCF 09-15519.

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 more a more formal statement of the problem in Section 2.)

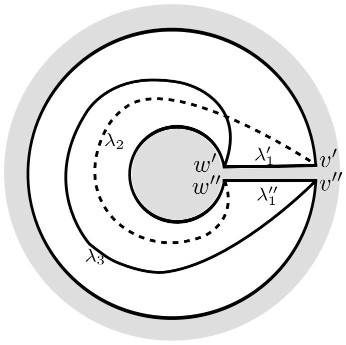


Figure 1. Reducing two obstacles to one.

Motivated by problems in VLSI design, Takahashi *et al.* [21] 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; thus, by cutting along each of these 3 paths in turn, they reduce the problem to the single-obstacle case. The output walks could have complexity $\Omega(kn)$ in the worst case, however, their algorithm actually computes a implicit representation with complexity $O(n)$. The geometric formulation of the shortest non-crossing walks problem was proposed by Papadopoulou [17], 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.* [22] 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.

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 [18] 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 and Mitchell [19, 18] 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 4we describe an algorithm for the graph formulation that runs in time $2^{O(h^2)} n \log k$, and in Section 5, 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 3, that in the shortest set of 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, 14]. Like earlier algorithms for the case $h \leq 2$ [21, 17], our algorithms can be modified to find non-crossing walks that minimize any increasing function of their lengths.

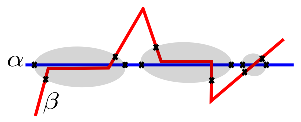
Due to space limitations, some auxiliary results are deferred to the appendix.

[1]Takahashi *et al.*report a running time of $O(n \log n)$, but the time bound can be improved using the linear-time shortest-path algorithm of Henzinger *et al.* [11].

2 Preliminaries

2.1 Background

Curves in the plane. Let $\mathcal{S}$ be a compact subset of the plane. A ***curve*** in a surface $\mathcal{S}$ 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 $t \le 1/2$ and $(\alpha \cdot \beta)(t) = \beta(2t-1)$ if $t \ge 1/2$. 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 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. 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.

A ***homotopy*** between two walks α and β is a continuous map $h\colon [0,1] \times [0,1] \to \mathcal{S}$ 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 $\mathcal{S}$ maps the vertices of G to distinct points in $\mathcal{S}$ and 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 is a record of 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 [16]. 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 where consecutive elements are incident; the ***endpoints*** of a walk are its initial and final vertices; a walk is 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.* [21], and a *geometric* formulation previously considered by Takahashi *et al.* [22] and Papadopoulou [17].

In the geometric formulation, the input consists of k 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***. 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. Our goal is to compute a set of *non-crossing* ST-walks of minimum total length. To make the definition of crossing precise, we implicitly extend each walk ω_i infinitesimally into the obstacles at their endpoints.

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 $H = \{f_1, f_2, \dots, f_h\}$ of faces of G, called ***obstacles***; and two disjoint sets of vertices $S = \{s_1, \dots, s_k\}$ and $T = \{t_1, \dots, t_k\}$, called ***terminals***, where each terminal is incident to a single obstacle face. A ***set of ST-walks*** is a set of walks $\Omega = \{\omega_1, \dots, \omega_k\}$ in G, where each walk ω_i connects s_i and t_i. Again, our goal is to compute a set of *non-crossing* ST-walks in G of minimum total length. 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 in our algorithms that each terminal has degree 1, and each walk ω_i is forbidden to visit terminals s_j or t_j with $j \neq i$. It is convenient to think of the obstacles as holes in the plane.

When $h = 1$, the shortest non-crossing ST-walks are actually *shortest paths* joining corresponding terminals; however, for any $h \geq 2$, there are inputs where 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.

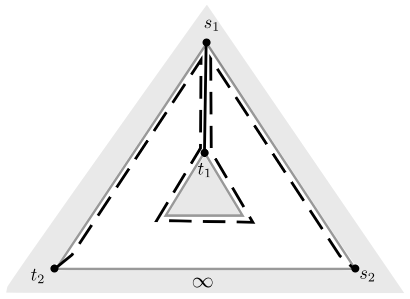


Figure 2. Shortest non-crossing walks may not be simple.

Even when $h = 1$, the total complexity of the shortest non-crossing ST-walks is $\Omega(nk)$ in the worst case. For the special case $h = 1$, to avoid worst-case quadratic running time, the algorithms of Takahashi *et al.* [21] and Papadopoulou [17] compute a forest that contains the shortest ST-paths. Our algorithms compute a similar implicit representation for all h. A ***non-crossing forest*** is an acyclic graph F, together with a continuous map from F to either $\mathbb{S}$ or G, such that the image of any two paths in F is a pair of non-crossing walks. A non-crossing forest represents a set of non-crossing ST-walks 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. For any constant h, our algorithms compute a shortest non-crossing forest of complexity $O(n)$ that represent the shortest non-crossing ST-walks. If an explicit representation of the walks is required, we can extract each walk in time proportional to its complexity, using fast least-common-ancestor queries [2]; our reported running times suppress this output term.

2.3 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 $\mathbb{S}$ be a polygon with holes in the plane; a ***properly embedded arc*** in $\mathbb{S}$ is a simple path whose endpoints lie on the boundary of $\mathbb{S}$, and that is otherwise disjoint from the boundary. For any properly embedded arc α, let $\boldsymbol{\mathbb{S} ✂ \alpha}$ denote the surface obtained by ***cutting*** $\mathbb{S}$ along α; each point of σ becomes a pair of boundary points in the new surface. topologically, $\mathbb{S} ✂ \alpha$ is the closure of $\mathbb{S} \setminus \alpha$; geometrically, $\mathbb{S} ✂ \alpha$ is a degenerate simple polygon with holes.

Now let ω be a non-self-crossing walk in $\mathbb{S}$ whose endpoints lie on $\partial\mathbb{S}$. We intuitively define $\mathbb{S} ✂ \omega$ 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 $\mathbb{S} ✂ \omega$ to be the topological space $\mathbb{S} ✂ \tilde{\omega}$ together with a continuous function $\phi\colon \mathbb{S} ✂ \tilde{\omega} \to \mathbb{S}$ 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

$\mathbb{S} ⫽ \tilde{\omega}$ is now defined to be the length of the projected walk $\phi(\omega')$ in the original space $\mathbb{S}$. 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, 9, 10, 14]. 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.

3 Crossing Bounds

In this section, we derive an upper bound on the maximum number of times a minimum-length set of non-crossing walks can cross an arbitrary shortest path. Our proof uses an exchange argument, similar to arguments previously used to characterize shortest noncontractible cycles [3], shortest splitting cycles [4], and minimum cuts in surface-embedded graphs [5], as well as weak realizations of string graphs [20]. 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 Appendix B we derive a lower bound for the same value, using an explicit inductive construction. Our crossing bounds immediately imply bounds on the worst-case complexity of a minimum-length set of non-crossing walks.

Throughout this section, 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 the results of the previous section, we assume that the terminal pairs can be connected by non-crossing walks in G.

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 of minimum total length, which has the minimum number of crossings between walks ω_i and shortest paths σ_j among all such sets of walks. (In the geometric setting, minimizing length automatically minimizes the number of crossings, but the combinatorial setting is more subtle.) In this section, we prove that each walk ω_i crosses each shortest path σ_j at most $2^{O(h)}$ times; this bound does not depend on k or n at all.

For each walk ω_i and each shortest path σ_j, we define a plane graph $H_{ij} = (V_{ij}, E_{ij})$, whose vertices V_{ij} are the crossing points of ω_i and σ_j, and whose edges E_{ij} 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} *red* if it is a subwalk of ω_i and *blue* if it is a subwalk of σ_j. The graph H_{ij} has a natural planar embedding, in which red and blue edges alternate around every vertex of degree 4. Thus, H_{ij} has 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; clearly there are at most h non-empty faces. A face of H_{ij} is a ***bigon*** if it has exactly two boundary edges.

Lemma 3.1. *No bigon in H_{ij} is empty.*

Proof: Suppose H_{ij} has an empty bigon B, whose boundary is composed of a red edge $r \subset \omega_i$ and a blue edge $b \subset \sigma_j$. Every other walk in Ω that intersects B must cross b an even number of times, but cannot cross r. For each 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 b. In particular, ω_i' is defined by replacing r with b in ω_i. See Figure 3.

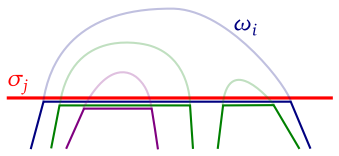


Figure 3. Removing an empty bigon.

Let $\Omega' = \{\omega_1', \dots, \omega_k'\}$. 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 b. 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 path b. 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 in H_{ij}, and let C_{ij} denote the subgraph of red edges. The subgraph T_{ij} is a spanning tree of H_{ij}; thus, the dual subgraph C_{ij}^* is a spanning tree of the dual graph H_{ij}^*. Call a vertex of C_{ij}^* ***good*** if the corresponding face of H_{ij} is an empty rectangle, and ***bad*** otherwise.

Lemma 3.2. *The total degree of all bad vertices of C_{ij}^* is less than $5h$.*

Proof: We separately consider bad vertices with degree at most 2 and vertices with degree at least 3 (which are all bad). At most h vertices of C_{ij}^* are dual to non-empty faces of H_{ij}. Thus, the bad vertices with degree at most 2 have total degree at most $2h$.

Each leaf in C_{ij}^* is dual to a bigon, and by Lemma 3.1, each bigon is non-empty; thus, C_{ij}^* has at most h leaves. Let r be the number of nodes in C_{ij}^* with degree greater than 2, and let d be their total degree. We clearly have $d \geq 3r$, and because the average degree of all vertices in any tree is less than 2, we also have $d + \ell < 2(r + \ell)$, where ℓ is the number of leaves. It follows that $d + \ell < 2(d/3 + \ell)$, which implies that $d < 3\ell < 3h$. $\square$

The good vertices in C_{ij}^* induce a collection of paths. We call the sequence of faces dual to each path a ***street***. 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. Moreover, the subwalks of Ω that traverse any street do it in parallel; their endpoints have the same order at both ends of the street. This parallelism is crucial for our surgery argument.

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} crosses or ends a street, it 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 streets. 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}.

A string is *even* if every character appears an even number of times. For example, `deed` is an even substring of `abdcdeedba`.

Lemma 3.3. *Any string of length 2^k with at most k distinct characters has a non-empty even substring.*

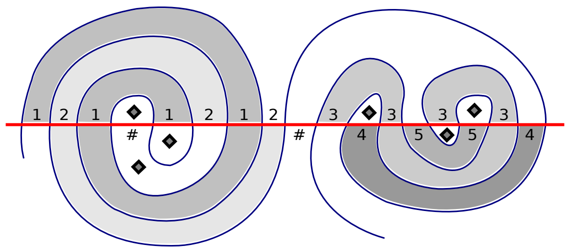


Figure 4. Five streets defining the street sequence 121#1212#34353534. Black diamonds indicate obstacles.

Proof: Let A be a string of length $n = 2^k$ over the alphabet $\{x_1, \dots, x_k\}$. We define a sequence of k-bit strings $B_1, B_2, \dots, B_n$ 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. $\square$

Theorem 3.4. *Each walk ω_i crosses each shortest path σ_j at most $5h\,2^{5h-1}$ 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}. Every street in H_{ij} starts and ends at a red edge whose dual in C^*_{ij} is incident to exactly one bad vertex. Each occurrence of the symbol # in S_{ij} corresponds to an edge between two bad vertices in C^*_{ij}. Thus, Lemma 3.2 implies that $s + 2x < 5h$. In particular, we have both $s < 5h$ and $x < 5h/2$.

Suppose some walk ω_i crosses some shortest path σ_j more than $5h\,2^{5h-1}$ times. Then the length of the street sequence S_{ij} is at least $5h\,2^{5h-1}$. Thus, S_{ij} has a substring S' of length at least 2^{5h} in which the symbol # does not occur. Because s has less than $5h$ distinct characters, Lemma 3.3 implies that S' contains a nonempty even substring S''. Let σ'' be the subpath of σ_j that corresponds to the substring S'' of S_{ij}. We claim that we can modify all the walks in Ω that cross σ'' to reduce the total number of crossings, without increasing their length, thereby contradicting our original choice of walks Ω.

We define an ***active substreet*** to be a portion of a street that starts and ends at edges in σ''. Each street that intersects σ'' has an odd number of active substreets (possibly with subsets at the ends that are not active substreets); a street that does not intersect σ'' has no active substreets. We index the active substreets of each street in order from one end of the street to the other; an ***odd substreet*** is an active substreet whose index is odd. Every edge in σ'' is the end of exactly one odd substreet, and if a street has $2r$ intersections with σ'', it has exactly r odd substreets.

For each index x and each odd substreet T, we call each component of $\omega_x \cap T$ an ***odd subwalk*** of ω_x. We modify Ω by replacing each odd subwalk α with the subpath $\alpha' \subset \sigma''$ between the endpoints of α. See Figure 5. Subwalks of ω_i (the blue edges of H_{ij}) lie on the boundary between streets; to make the surgery for these subwalks ω_i consistent and unambiguous, we arbitrarily orient ω_i and then pretend each subwalk of ω_i lies inside the substreet on its left (if any). Let $\Omega' = \{\omega'_1, \dots, \omega'_k\}$ denote the resulting set of modified walks.

Each modified walk ω'_x is no longer than the corresponding original walk ω_x, because we replace odd subwalks with shortest paths. Moreover, each walk ω'_x crosses each shortest path $\sigma_y \in \Sigma$ fewer times than ω_x.

Let α and β be two odd subwalks of walks in Ω, and let α' and β' be their replacement paths. To make our crossing argument exact, we infinitesimally extend α, β, α', and β' at both ends, to obtain walks α_e, β_e, α'_e, and β'_e. We claim that α'_e and β'_e do not cross. This claim implies that Ω' is a set of non-(self-)crossing walks, which completes the proof of the theorem.

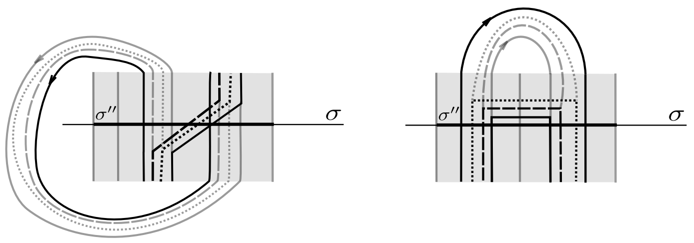


Figure 5. Surgery on σ.

Suppose to the contrary that α'_e and β'_e do cross. Because α' and β' are subpaths of a common path, paths α'_e and β'_e cross exactly once; the possible crossing patterns are shown in Figure 6. In each case, α_e crosses β'_e exactly once, and α'_e crosses β_e exactly once. The walks $\alpha'_e \cdot \alpha_e$ and $\beta'_e \cdot \beta_e$ are closed and therefore must cross an even number of times. It follows that α_e and β_e must cross an odd number of times, and therefore at least once. But this is impossible, because α_e and β_e are subwalks of walks in Ω. □

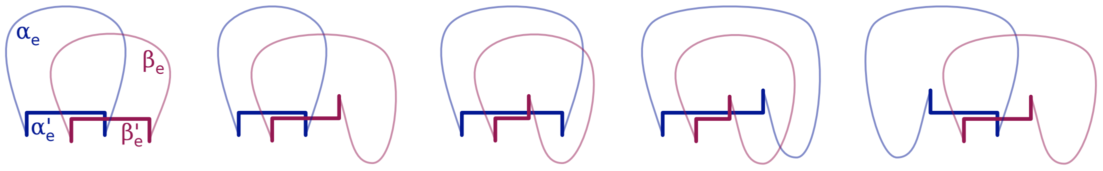


Figure 6. If α'_e and β'_e cross, then α_e and β_e also cross.

4 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.* [21], 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 n, compute the shortest path σ_i in G connecting s_i and t_i, and then replace G with $G \not\!\!\!\text{✂} \sigma_i$. Takahashi *et al.* describe a divide-and-conquer algorithm (called 'PATH2') to compute a forest containing all the shortest paths between terminals on a *single* face in $O(n \log k)$ time.

Lemma 4.1. *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 all 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.* [21] for the special case $h = 2$; similar strategies have been used to find various optimal topologically interesting cycles in combinatorial surfaces [3, 4, 5, 14].

4.1 Spanning Walks

The obstacles and terminal pairs naturally define an undirected (multi-)graph C, called the *connection graph*, which has a vertex for each obstacle face and an arc joining the obstacles incident to each terminal pair s_j and 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 $(s_1, t_1), \dots, (s_m, t_m)$; let $S' = \{s_1, \dots, s_m\}$ and $T' = \{t_1, \dots, t_m\}$. Note that $m \le h - 1$.

We consider two walks in G to be homotopic if they are homotopic in the plane minus the h obstacle faces. A walk is ***tight*** if it is a minimum-length walk in its homotopy class. Results of Colin de Verdière [8, 9, 10] 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 $\tilde{\Omega}$ is 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 3.1.

Thus, given a set Ω' of tight non-crossing $S'T'$-walks *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 $G ✂ \Omega'$, which has exactly one obstacle for each connected component of the connection graph C.

4.2 Enumerating Homotopy Classes

We enumerate the homotopy classes of non-crossing $S'T'$-walks that satisfy the crossing bound in Theorem 3.4 as follows. We first compute a set $\Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_{h-1}\}$ of non-crossing shortest paths that connect the obstacle faces $f_0, f_1, \dots, f_{h-1}$. Specifically, we compute a shortest-path tree rooted at an arbitrary vertex v_0 on obstacle f_0 in $O(n)$ time [11], 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 may require subdividing some edges in G.)

Let $G ✂ \Sigma$ denote the planar graph obtained by cutting G along every shortest path $\sigma_i \in \Sigma$. Following Chambers *et al.* [4], we represent $G ✂ \Sigma$ 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 paths σ_i in $G ✂ \Sigma$, 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 3.1. 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 3.4, then the each diagonal has weight at most $2^{O(h)}$.

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.

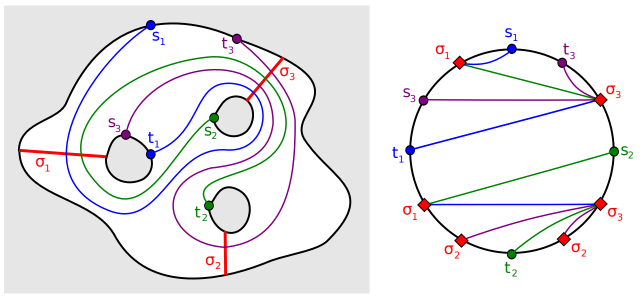


Figure 7. Representing non-crossing walks using an abstract polygonal schema.

4.3 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 [14]. 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 walks 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 i, let $x_i \leq 2^{O(h)}$ denote the length of crossing pattern X_i.

We can compute a shortest walk with a given crossing sequence X_i as follows. First, glue together x_i copies of $G \,\text{✂}\, \Sigma$ 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 $G \,\text{✂}\, \Sigma$ and t_i in the final copy of $G \,\text{✂}\, \Sigma$, using the linear-time shortest path algorithm of Henzinger *et al.* [11]. 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 $H = G \,\text{✂}\, \Sigma$. For each 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 $G \,\text{✂}\, \omega_i$, and replace H with $H \,\text{✂}\, \omega_i$. 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 $G \,\text{✂}\, \Sigma$ 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.

4.4 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 $G \,\text{✂}\, \Omega'$ has complexity at most $n + 2^{O(h)}$; thus, we can extend Ω' to a set of tight non-crossing ST-walks in time $O((n + 2^{O(h)}) \log k)$ using Lemma 4.1. We conclude:

Theorem 4.2. *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.*

5 Geometric Algorithm

Now we describe the geometric version of our shortest non-crossing ST-walk algorithm to the geometric setting. The input consists of h simple polygonal obstacles $P_1, P_2, \ldots, P_h$ in the plane with total complexity n, along with two disjoint sets $S = \{s_1, \dots, s_k\}$ and $T = \{t_1, \dots, t_k\}$ 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.

5.1 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 shortest set of non-crossing ST-walks consists of the *unique* globally shortest paths between the terminal pairs, which are unique if 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 [13]. Here we describe an algorithm that runs in $O(n)$ time when h is constant, generalizing an algorithm of Papadopoulou [17] 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 $\Sigma = \{\sigma_1, \dots, \sigma_h\}$, 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 easily compute these segments in $O(hn)$ time by brute force. The space $W' = W \,✂\, \Sigma$ 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 easily 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 easily 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* [12], 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 [17]. 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, 15, 12]. The output of Papadopoulou's algorithm is a forest F_X of complexity $O(h!\,n)$ containing the required shortest paths.

We now have $O(h!)$ non-crossing forests F_X, one for each crossing sequence X. For each index i, we can easily 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-paths that have crossing sequence X. Finally, we return the union of all forests F'_X.

Lemma 5.1. *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 all i, terminals s_i and t_i lie on the same obstacle.*

5.2 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 $\Sigma = \{\sigma_1, \dots, \sigma_h\}$ 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 $S' = \{s_1, \dots, s_m\}$ and $T' = \{t_1, \dots, t_m\}$. 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 [12] in place of the planar graph algorithm of algorithm of Henzinger *et al.* [11]. Finally, we extend Ω' to a set of tight non-crossing ST-walks using the same-obstacle algorithm in the space W ✂ Ω'.

Theorem 5.2. *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.*

References

[1] O. Bastert and S. P. Fekete. Geometric wire routing. Technical Report 98-332, Zentrum für Angewandte Informatik, Univ. Köln, 1998. Cited by [19].

[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 pants decompositions and shortest homotopic cycles on an orientable surface. *Proc. 11th Symp. Graph Drawing,* 478–490, 2003. Lecture Notes Comput. Sci. 2912, Springer-Verlag.

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

[11] 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.

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

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

[14] 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.

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

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

[17] 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.

[18] 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⟩.

[19] 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.

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

[21] 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.

[22] 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.

A 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 $s_1, t_2, \dots, s_k, t_k$, 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 vertex v_i for each obstacle face f_i and k arcs $a_1, a_2, \dots, a_j$, where each a_j join the obstacles incident to 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 A.1. *Let $s_1, t_1, s_2, t_2, \dots, s_k, t_k$ 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 $w_1, w_2, \dots, w_k$ in G, where each walk w_i connects terminals s_i and t_i. As discussed in Section 2.3, we can perturb these walks infinitesimally to obtain a set of disjoint simple paths $\tilde{w}_1, \tilde{w}_2, \dots, \tilde{w}_k$ in the plane, where each path $\tilde{w}_i$ connects terminals s_i and t_i. Place a point v_i in the interior of each face f_i. For each obstacle face f_i, extend all the paths $\tilde{w}_j$ 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 vertex of C to the corresponding sentinel point and maps the arcs of C to disjoint simple paths $a_1, a_2, \dots, a_k$ 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 $\tilde{w}_1, \tilde{w}_2, \dots, \tilde{w}_k$ 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 w_i be the graph-theoretic walk that visits the vertices of G in the same order as $\tilde{w}_i$. The walks $w_1, w_2, \dots, w_k$ are homotopic to the non-crossing topological walks $\tilde{w}_1, \tilde{w}_2, \dots, \tilde{w}_k$ and therefore do not cross. □

This lemma implies the following 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 in $O(n)$ time, 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 [16]. 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 A.2. *Let $s_1, t_1, s_2, t_2, \dots, s_k, t_k$ 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 $P_1, P_2, \dots, P_k$ 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 vertex for each obstacle P_i and an edge 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 A.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 A.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.*

B Lower Bound

Fix a positive integer n. Let G be the plane graph with vertices $\{s_1, t_1, \ldots, s_n, t_n, u, v, w\}$, with an edge between v and every other vertex and loops at each vertex s_i. We embed G in the plane so that the counterclockwise order of neighbors around v is $s_1, u, s_2, t_1, s_3, t_2, \ldots, t_{n-2}, s_n, t_{n-1}, t_n, w$. See Figure 8. The loops at the terminal vertices s_i are the obstacles. For each i, let ℓ_i denote the loop at vertex s_i. 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.

We inductively construct a set of ***canonical*** non-crossing walks $\Omega^* = \{\omega_1^*, \omega_2^*, \ldots, \omega_n^*\}$ in G, where each walk ω_i connects s_i and t_i, as follows. We first define a sequence $\alpha_1, \alpha_2, \ldots, \alpha_k$ of closed walks starting and ending at v. Specifically, α_1 is the empty walk, and for each $i \geq 2$, we set $\alpha_i = rev(\alpha_{i-1}) \cdot (v, s_{i-1}) \cdot \ell_{i-1} \cdot (s_{i-1}, v) \cdot \alpha_{i-1}$. Finally, $\omega_i^* := (s_i, v) \cdot \alpha_i \cdot (v, t_i)$ for all 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.

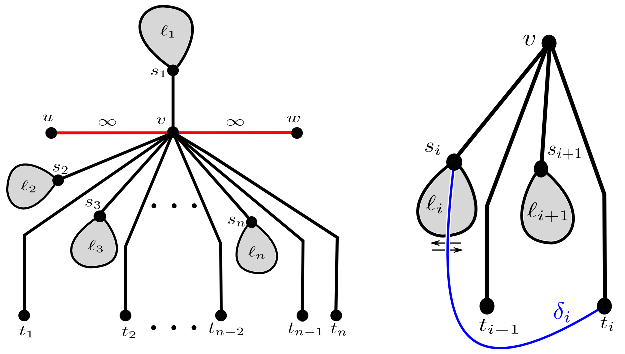


Figure 8. Left: Shortest non-crossing walks can have exponential complexity. Right: Defining the path δ_i.

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

Lemma B.1. *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* $\max\{0, 2^{j-i-1}\}$ *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 $\max\{0, 2^{j-i-2}\}$ 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. (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 $i' > i+1$, the loop $\ell_{i'}$ lies completely outside ρ_i.

Walk ω_{i+1} starts outside ρ_i and ends inside 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 through 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

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

Thus, if any walk ω_j traversed loop ℓ_i more than $\max\{0, 2^{j-i-1}\}$ 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 $\max\{0, 2^{j-i-1}\}$ 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 $n = 3$.

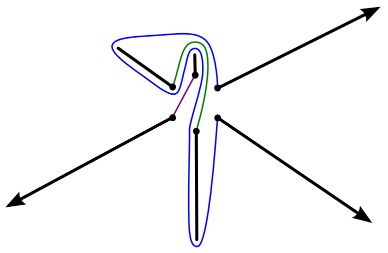


Figure 9. A geometric version of our exponential lower bound.