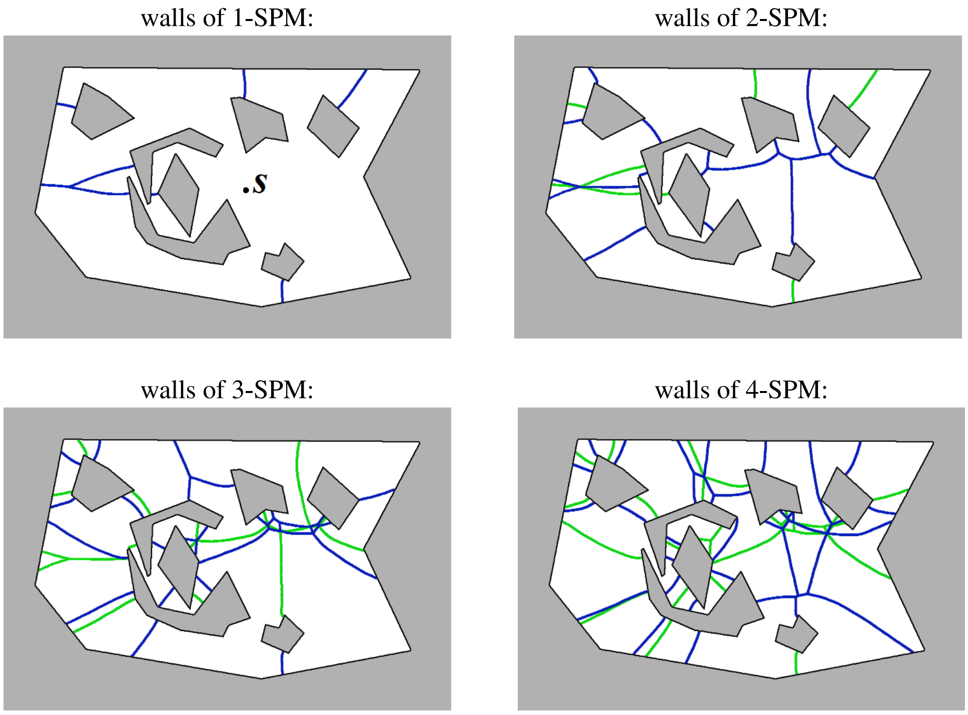
Geometric kth Shortest Paths*

Sylvester Eriksson-Bique† John Hershberger‡ Valentin Polishchuk§
Bettina Speckmann¶ Subhash Suri‖ Topi Talvitie§ Kevin Verbeek‖
Hakan Yıldız‖

Abstract

This paper studies algorithmic and combinatorial properties of shortest paths of different homotopy types in a polygonal domain with holes. We define the "second shortest path" to be the shortest path that is homotopically different from the (first) shortest path; the kth shortest path for an arbitrary integer k is defined analogously. We introduce the "kth shortest path map"—a structure to answer kth shortest path queries. Given a polygonal domain with n vertices and h holes, we show that the complexity of the kth shortest path map is $O(k^2h + kn)$, which is tight. Furthermore, we show how to build the kth shortest path map in $O((k^3h + k^2n) \log (kn))$ time. We also present a simple visibility-based algorithm to compute the kth shortest path between two points in $O(m \log n + k)$ time, where m is the complexity of the visibility graph. This last approach can be extended to compute the kth simple (i.e., without self-intersections) shortest path in $O(k^2m(m + kn) \log kn)$ time.



We invite the reader to play with our applet demonstrating k-SPMs at
`http://www.cs.helsinki.fi/group/compgeom/kpath_slides/visualize/`.

*B. Speckmann and K. Verbeek were supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 639.022.707. Sylvester E-B was supported as a Graduate Student Fellow by the National Science Foundation grant no. DGE-1342536.

†Courant Institute, NYU ebs@cims.nyu.edu

‡Mentor Graphics Corporation john_hershberger@mentor.com

§Helsinki Institute for IT, CS Dept, University of Helsinki firstname.lastname@helsinki.fi

¶Dept. of Mathematics and Computer Science, TU Eindhoven b.speckmann@tue.nl

‖Computer Science, University of California Santa Barbara [suri|kverbeek|hakan]@cs.ucsb.edu

1 Introduction

Computing shortest paths in polygonal domains is one of the oldest and most studied problems in computational geometry. Given a planar domain with polygonal holes and two points in this domain (a source and a target), the problem is to compute a path in the domain that connects the source to the target and has the shortest length possible. Due to its natural formulation and practical applications (such as in robotics) the problem has drawn interest of many computational geometers.

In this paper, we study a variation of the geometric shortest path problem in which the goal is to compute, for a given k, the first k shortest paths between two points, rather than a single shortest path. A similar variation has been studied for shortest paths on graphs. In addition to its theoretical interest, the geometric kth shortest path problem is also motivated by some real-world applications. One particular application is air traffic management (ATM), in which the airspace at a given flight level is modeled by a polygonal domain with holes corresponding to hazardous weather cells, no-fly zones, and other obstacles for traffic. Because it is impossible to capture formally all nuances of ATM route design, it seems natural to present an air traffic controller with a set of options, leaving the final choice of the flight path to human judgment. More generally, various applications of kth shortest paths in graphs are relevant also in geometric domains; one example is multiple object tracking [1].

The reader may have noticed that the concept of kth shortest paths, in its exact meaning, is not formally well-defined in the geometric setting. Unlike paths in graphs, the paths in a polygonal domain do not form a countable set and thus one cannot talk about a kth shortest path without additional restrictions. (In the geometric setting, new paths can be created by infinitesimal deviations.) In order to establish a well-defined problem, we consider *homotopically different* paths only. In other words, we define the second shortest path as the shortest path that is homotopically different from the (first) shortest path. Similarly, the third shortest path is homotopically different from the first two, and so on. This leads to the following problem:

> Given a polygonal domain P, two points $s, t \in P$ and a number k, find k homotopically different shortest s–t paths (Fig. 1).

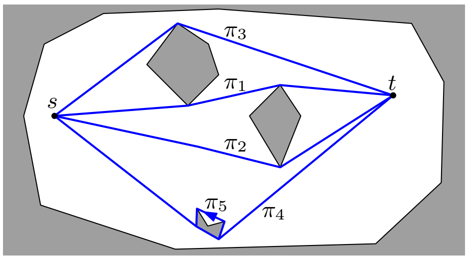


Figure 1: $|\pi_1|<|\pi_2|=|\pi_3|<|\pi_4|<|\pi_5|$. π_1 is the shortest path to t (a 1-path; cf. Def. 2.2), each of π_2 and π_3 is a 2-path, π_4 is a 4-path, π_5 is a 5-path (π_5 is nonsimple—it is equal to π_4 plus the loop around the hole).

Since any homotopy type can be associated with the length of the shortest path of the type, our problem can be viewed as that of *listing* homotopy types in order of increasing length.

Related work Finding shortest paths is also a central problem in the study of graph algorithms. Apart from finding the shortest path itself, considerable attention has been paid to computing its various alternatives including the second, third, and in general kth shortest path between two nodes in a graph; see, e.g., [9, 11] and references therein. On the other hand, *geometric* kth shortest paths have not been explored before. (One problem for which both the graph and the geometric versions were considered is finding the k smallest spanning trees [7, 8].)

In [17] Mitchell surveys many variations of the geometric shortest path problem; for some recent work see [4, 5]. In addition to computing one shortest path to a single target point, a lot of attention in the literature has been devoted to building shortest path *maps*—structures supporting efficient shortest-path queries. A shortest path map can be viewed as the Voronoi diagram of vertices of the domain, where each vertex is (additively) weighted by the shortest-path distance from the source s [12]. Our study of "kth shortest path maps" benefits from notions introduced by Lee [14] for *higher-order* Voronoi diagrams: when bounding the complexity of the maps in Section 4.2, we employ Lee's ideas to define "old" and "new" features of the map and to derive relationships between them. Higher-order Voronoi diagrams have been recently reexamined in [2, 15, 16, 18]; in particular, [15] considered geodesic diagrams in polygonal domains. Perhaps unsurprisingly, the complexity of our kth shortest path map differs from that of an order-k geodesic Voronoi diagram; the major difference is that homotopies are irrelevant for Voronoi diagrams, but are central in our work.

Results In Section 3 we give a simple algorithm for finding the kth shortest path. If n is the number of vertices of P and m is the size of its visibility graph, the algorithm runs in $O(m \log n + k)$ time and $O(m + k)$ space. Note that $m = \Omega(n)$, and in the worst case $m = O(n^2)$. We also study the query version of the problem: report (the lengths of) k shortest paths from a query point to a fixed source s. In Section 4 we present our main contribution—an $O(k^2 h + kn)$-size data structure (for a domain with h holes) that can be built in $O((k^3 h + k^2 n) \log(kn))$ time and answers kth shortest path queries in $O(\log(kn))$ time apiece. If we want to report all k shortest paths from a query point, the preprocessing time remains the same, but the storage and query time both increase by a factor of k. Finally, in Section 5 we present an $O(k^2 m(m + kn) \log kn)$-time algorithm to find the kth simple (i.e., without self-intersections) shortest path. Omitted proofs can be found in Appendix B.

2 Preliminaries

We are given a polygonal domain P with n vertices and h holes; the holes are also called "obstacles" and the domain is called the "free space." We assume that no three vertices of P are collinear and make other general position assumptions below, as needed. We are also given a source point $s \in P$; unless otherwise stated, all paths will have s as an endpoint. For a point $p \in P$, two paths to p are *homotopically equivalent* if one can be continuously deformed to the other while staying within P. Homotopically equivalent paths form an equivalence class (the *homotopy class*) in the set of s–p paths. The unique shortest path in a homotopy class (i.e., a pulled taut path) is called *locally shortest*.

Observation 2.1. *All bends of a locally shortest path π are at vertices of P and turn toward the corresponding obstacles.*

Let $d(p)$ denote the shortest-path (geodesic) distance from s to p. A vertex v of P is a *predecessor* of p if segment $\overline{vp}$ is in free space and $d(p) = d(v) + |vp|$. The *shortest path map* of P (or SPM for short) is the partitioning of P such that all points within the same cell of the SPM have the same unique predecessor. The edges of the partition are called *bisectors*; points on bisectors have more than one predecessor. We distinguish between two types of bisectors: *walls* and *windows*. A bisector is a wall if, for a point p on the bisector, there exist two homotopically different paths to p with length $d(p)$. If there is a unique shortest path to a point p on a bisector, then this bisector is a window; any point p on a window has two predecessors that are collinear with p. We assume that there is a unique shortest path to each vertex of P, and that there are at most three homotopically different shortest paths to each point in P. The former assumption implies that walls are 1-dimensional curves. The endpoints of a wall are either at an obstacle or at a *triple point*, where three walls meet. Windows start at vertices of P and extend until an obstacle or wall is hit. Intuitively, windows can mostly be ignored as far as homotopy types are concerned; walls, by contrast, are central to our study. Fig. "1-SPM" on the title page shows an example of walls in the SPM. By using standard point location structures on the SPM of P, one can query the shortest path length to any point in P in $O(\log n)$ time and, in addition, report the path in linear output sensitive time [12]. Our goal is to compute a similar structure for kth shortest paths.

We now introduce the subject of our study. For a point $p \in P$, let $H(p)$ denote the set of locally shortest paths from s to p of all possible homotopy types.

Definition 2.2. *A path $\pi \in H(p)$ is a k*th shortest path *(or is a k-*path*) to p if there are exactly $k - 1$ shorter paths in $H(p)$ (see Fig. 1).*

We denote the length of the k-path(s) to p by $d_k(p)$. Notice that, under these definitions, the term 1-path is synonymous with "shortest path" and $d(p) = d_1(p)$.

In order to extend the map concept to k-paths, we first generalize the definition of a predecessor. Let v be an obstacle vertex and i be an integer between 1 and k. For a point p on the plane, the pair (v, i) is a *k-predecessor* of p if the segment $\overline{vp}$ is in free space and $d_k(p) = d_i(v) + |\overline{vp}|$. This implies that a k-path to p can be obtained by concatenating the segment $\overline{vp}$ with the i-path to v. As with the SPM, we assume that each obstacle vertex has a unique i-path for any i, and that there are at most three i-paths in $H(p)$ for each point $p \in P$. Interestingly, for $i > 1$, the former assumption does not follow from a general position assumption. We discuss this issue in Appendix A. For the sake of simplicity, we will ignore the issue in the main body of the paper and stick to the assumption above.

Observe that, given the k-predecessors of all points in the plane and the i-predecessors of all obstacle vertices for $1 \leq i \leq k$, one can construct the k-path to any given point p. The *kth shortest path map* (or k-SPM for short) of P is a subdivision of P into cells such that all points within the same cell have the same unique k-predecessor. In order to construct k-paths from the k-SPM, we also assume that it stores the i-predecessors of all vertices, for all $1 \leq i \leq k$. As with the SPM, one can use standard point location structures to report the k-path length of a query point in $O(\log C_k)$ time, where C_k is the complexity of the k-SPM.

To distinguish the different types of bisectors that form the boundaries of the k-SPM, we generalize the definitions of walls and windows as follows:

Definition 2.3. *A point p is on a k-*wall *if $H(p)$ contains at least two k-paths.*

Definition 2.4. *A point p is on a k-window if $H(p)$ contains exactly one k-path and p has two k-predecessors.*

Note that the two predecessors of a point p on a k-window must be collinear with p. Furthermore, by the definition of k-paths, a point cannot be on a k-wall and a $(k+1)$-wall at the same time (if a point has two k-paths, then it has no $(k+1)$-path). Similarly to walls in the SPM, k-walls have their endpoints either on obstacles or at triple points, where three k-walls meet. In Section 4.1, we show that edges of the k-SPM are $(k-1)$-walls, k-walls and k-windows. We also show that our assumption that a k-predecessor is of the form (v, i) with $1 \leq i \leq k$ is indeed correct.

3 A simple visibility-based algorithm

In this section we present a simple visibility-based algorithm to compute the k-path from s to some fixed target $t \in P$. For large k, this algorithm is faster than the k-SPM approach of Section 4. Moreover, this algorithm is relatively easy to implement and may therefore be of more practical interest.

We first compute the visibility graph (VG) of P in $O(n \log n + m)$ time [19], where $m = O(n^2)$ is the size of VG. We also include visibility edges to s and t. The graph contains every locally shortest path from s to t and hence also the k-path to t. However, we cannot simply compute the kth shortest path in VG, since different paths in the graph may be homotopic. We therefore modify VG so that locally shortest paths are in one-to-one correspondence with paths in the modified graph—this ensures that different paths in the graph belong to different homotopy classes. First, we make the graph directed by doubling each edge. Then we expand each vertex v as illustrated in Fig. 2: Draw the two lines supporting the two obstacle edges incident to v; the lines partition the relevant visibility edges at v into two sets A and B (the visibility edges between the lines opposite the obstacle are irrelevant, because they cannot be used by shortest paths). Radially sweep a line through v, initially aligned with one of the obstacle edges, until it is aligned with the other obstacle edge. For each encountered visibility edge e, create a node with an incoming edge if $e \in A$, and an outgoing edge if $e \in B$. Connect all created nodes with a directed path. Also make a copy of this construction with all edges reversed. The expansion of v is connected with other expansions in the obvious way, as dictated by the visibility graph. Finally, remove edges directed toward s and away from t. The constructed graph—which we call the *taut graph* $\vec{G}(P)$—has $O(m)$ vertices and $O(m)$ edges and can be built in $O(m)$ time. Note that, by construction, every path in $\vec{G}(P)$ must be locally shortest and every locally shortest path from s to t exists in $\vec{G}(P)$.

We can now use the algorithm by Eppstein [9] to compute the kth shortest path from s to t in $\vec{G}(P)$, which corresponds to the k-path from s to t in P. This algorithm computes the k-path from s to t in $O(m \log n + k)$ time. It also simultaneously computes all i-paths from s to t for $1 \leq i \leq k$.

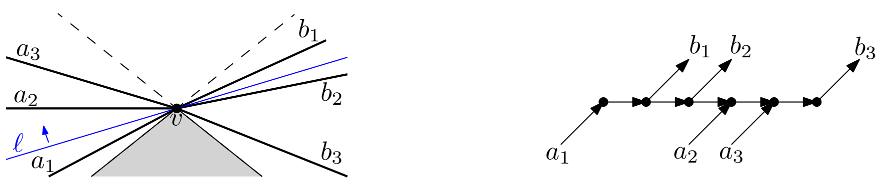


Figure 2: Vertex expansion for the taut graph.

4 The k-SPM

In this section we discuss the main contribution of this paper: the k-SPM. We first study the behavior of k-paths with respect to k-walls to derive the structure of the k-SPM. We then analyze the worst-case complexity of the k-SPM. Finally we show how to compute the k-SPM efficiently.

4.1 Structural results

Consider a path π from s to some target $t \in P$. This path crosses several walls (1-walls, 2-walls, etc.) in P. We define the *crossing sequence* of π as the sequence of positive integers that represents all the k-walls crossed by this path going back from t to s. That is, if π crosses an i-wall, we add i to the sequence. Although it is not strictly necessary, we generally assume an upper bound on the sequence values (the maximum wall class), so that the sequence is finite. We call a sequence a *k-sequence* if it adheres to the following inductive definition:

- A 1-sequence does not contain 1.
- A k-sequence contains $(k-1)$, the first $(k-1)$ occurs before the first k, and the tail of the sequence after the first $(k-1)$ is a $(k-1)$-sequence.

We need the following property of k-sequences.

Lemma 4.1. *A sequence σ cannot be both a k-sequence and an ℓ-sequence if $k \neq \ell$.*

The relation between k-sequences and k-paths is summarized in the following lemma.

Lemma 4.2. *A locally shortest path π is a k-path if and only if its crossing sequence is a k-sequence.*

Proof. We first show that the crossing sequence of a k-path π is a k-sequence. Let us assume that distances have been scaled so that the length of π is 1. Define $p(x)$ for $0 \leq x \leq 1$ as the point on π such that the distance from t to $p(x)$ along π is x. Let $\gamma(x)$ be the subpath of π from $p(x)$ to t. For any $i \geq 1$, let π_i denote the i-path to t ($\pi = \pi_k$). (We assume that t is not on an i-wall, for any $1 \leq i \leq k$.) The concatenation of π_i and $\gamma(x)$ is a path from s to $p(x)$, via t; let $\pi'_i(x)$ denote the shortest path of this homotopy class (Fig. 3, left). All paths $\pi'_i(x)$ must have different homotopy classes for different i.

Let $l_i(x)$ be the length of $\pi'_i(x)$; clearly l_i is continuous. By the definition of k-paths, $l_i(0) \leq l_j(0)$ for $i < j$. On the other hand, $l_k(1) = 0$ and $l_i(1) > 0$ for $i \neq k$. Note that as x grows from 0 to 1, $l_k(x)$ decreases not slower than any other $l_i(x)$, $i \neq k$. Thus, the graph of $l_k(x)$ crosses the graphs of all $l_i(x)$ for $i < k$, but no other graphs (Fig. 3, right).

The proof proceeds by induction. A point $p(x)$ is on a j-wall if two graphs cross at x, and there are exactly $j-1$ graphs that pass below this intersection. Clearly, if $k = 1$, the path π_k cannot cross a 1-wall, since $l_1(x)$ cannot intersect anything. For $k > 1$, the first intersection of $l_k(x)$ must be with a graph $l_i(x)$ with $i < k$, as described above. This means that $p(x)$ must cross a $(k-1)$-wall before crossing a k-wall. After the $(k-1)$-wall at $x = x^*$, the path $\pi'_k(x^*)$ is the $(k-1)$-path to $p(x)$. By induction, the remainder of the crossing sequence must be a $(k-1)$-sequence.

Finally note that a locally shortest path π must be an i-path for some $i \geq 1$. If the crossing sequence of π is a k-sequence, then it cannot be an i-sequence for $i \neq k$ by Lemma 4.1. Thus $i = k$, and π is a k-path. □

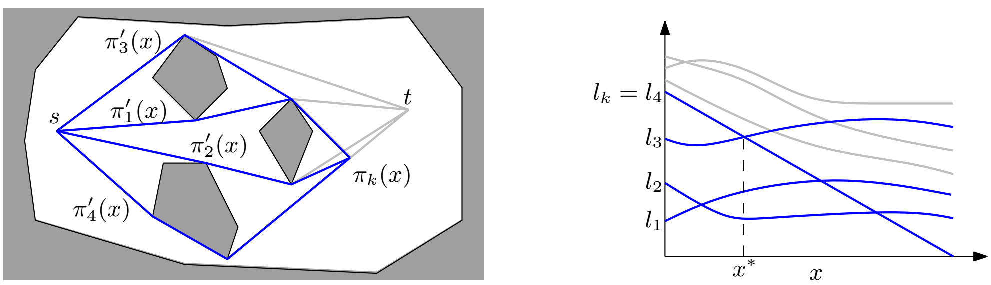


Figure 3: $k = 4$. Left: $\pi'_i(x)$ is the shortest path from $\pi_k(x)$, homotopically equivalent to $\pi_k(x)$–t–π_i–s. Right: l_k is kth smallest at $x = 0$ and decreases faster than any other l_i.

Lemma 4.2 means that a k-path from s to t crosses walls "in order": it crosses a 1-wall, then a 2-wall, etc., until it crosses a $(k-1)$-wall, after which it reaches t. Also, any prefix of the k$-path$ is an i-path if it crosses the $(i-1)$-wall and not the i-wall. This property of k-paths inspires the construction of a "parking garage" obtained by "stacking" k copies (or *floors*) of P on top of each other and gluing them along i-walls, for $1 \leq i \leq k$. To be precise, the *k-garage* is inductively defined as follows:

> The 1-garage is simply P. The $(k+1)$-garage can be obtained by adding a copy of P (the *$(k+1)$-floor*) on top of the k-garage. We cut both the k-floor of the k-garage and the $(k+1)$-floor along k-walls. We then glue one side of a k-wall on the k-floor to the opposite side of the same k-wall on the $(k+1)$-floor, and vice versa, to obtain the $(k+1)$-garage.

The k-garage resembles a covering space of P. However, due to the triple points formed by the i-walls $(i < k)$, the k-garage is technically not a covering space, but something that is known as a ramified cover. Nonetheless, each path π in the garage can be projected down to a unique path $\pi^{\downarrow}$ in P. The next lemma relates the k-SPM of P to the SPM of the k-garage.

Lemma 4.3. *If π is the shortest path in the k-garage from s on the 1-floor to some t on the k-floor, then $\pi^{\downarrow}$ is a k-path to t.*

Lemma 4.3 directly implies that the SPM on the k-floor of the k-garage is exactly the k-SPM of P. Thus, as claimed before, the edges of the k-SPM consist of $(k-1)$-walls, k-walls, and k-windows. Furthermore, the k-predecessor of a point $p \in P$ must be (v, i) for some i between 1 and k.

4.2 The complexity of the k-SPM

Lower Bound. For a lower bound on the complexity of the k-SPM, consider the example shown in Fig. 4. We construct the example in such a way that the shortest paths from the source s to the vertices p_1, p_2, and p_3 have the same length. Let q be the unique point such that $|q-p_1| = |q-p_2| = |q-p_3|$. Furthermore, let π_{ij} ($i \in \{1,2,3\}$ and $1 \leq j \leq k$) be the j-path from s to p_i, and let l_{ij} be the length of π_{ij}. If the obstacle ω_i is small enough, then π_{ij} simply loops around ω_i zero or more times in a clockwise or counterclockwise direction. Hence, for any $\epsilon > 0$, we can ensure that $|l_{ik} - l_{i1}| \leq \epsilon$ for $i \in \{1,2,3\}$ by making the obstacles ω_i small enough. Now define q_{abc} as the unique point such that $|q_{abc} - p_1| + l_{1a} = |q_{abc} - p_2| + l_{2b} = |q_{abc} - p_3| + l_{3c}$. This point must exist, since it is the vertex of an additively weighted Voronoi diagram of p_1, p_2, and p_3.

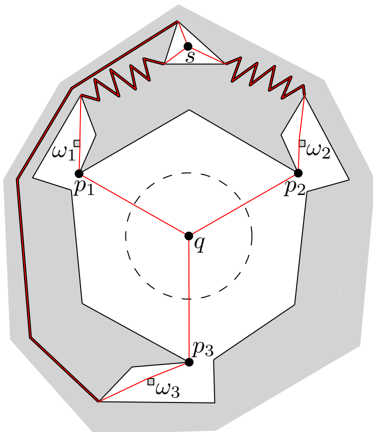


Figure 4: Lower bound construction.

Lemma 4.4. *If $\epsilon < |q - p_i|$ for $i \in \{1,2,3\}$, then $|q_{abc} - q| < \epsilon$.*

By Lemma 4.4, q_{abc} must lie in the free space (in the circle of Fig. 4), if ϵ is small enough. By construction there are three paths with equal length from s to q_{abc}, and there are exactly $a+b+c-3$ shorter paths from s to q_{abc}. This means that q_{abc} is a triple point of the $(a+b+c-2)$-SPM. Thus, the number of triple points of the k-SPM is exactly the number of triples (a,b,c) with $1 \leq a,b,c \leq k$ for which $a+b+c-2 = k$. It is easy to see that there are $\Omega(k^2)$ triples that satisfy these conditions. By connecting several copies of the construction together, we get a domain with h holes. Finally, we can replace p_3 in one copy by a convex chain of n vertices $v_1, \ldots, v_n$, such that the line through v_i and v_{i+1} is very close to q for $1 \leq i < n$. This way each vertex v_i contributes k k-windows to the k-SPM.

Theorem 4.5. *The k-SPM of a polygonal domain with n vertices and h holes can have $\Omega(k^2 h)$ k-walls and $\Omega(kn)$ k-windows.*

Upper Bound. To obtain an upper bound on the complexity of the k-SPM, we consider a sparser partitioning of P. We define the $(\leq k)$-SPM of P as the partitioning induced by only the k-walls of P. Let $H_k(p)$ be the set of the k shortest homotopy classes to $p \in P$. We refer to $H_k(p)$ as the *k-homotopy set* of p. We would like to claim that the set $H_k(p)$ is constant within each cell of the $(\leq k)$-SPM. Unfortunately we cannot claim this, since the homotopy classes of paths with different endpoints cannot be compared. To overcome this technicality, we define $H_k(p) \oplus \pi$ as the set of homotopy classes obtained by concatenating each path in $H_k(p)$ with π. If π is a path between p and p', then we can directly compare $H_k(p) \oplus \pi$ and $H_k(p')$.

Lemma 4.6. *If p and p' lie in the same cell of the $(\leq k)$-SPM, and π is a path between p and p' that does not cross a k-wall, then $H_k(p) \oplus \pi = H_k(p')$.*

To keep the notation simple, we simply compare $H_k(p)$ and $H_k(p')$ directly, in which case we really mean that we compare $H_k(p) \oplus \pi$ and $H_k(p')$, where π is the shortest path in P between p and p'. Note that π can cross a k-wall. We need the following property of the $(\leq k)$-SPM.

Lemma 4.7. *The cells of the $(\leq k)$-SPM are simply connected.*

We now count the number of k-walls, starting with the case $k = 1$. Let F_1, V_1, and B_1 be the number of faces, triple points, and 1-walls of the (≤ 1)-SPM, respectively. It is easy to see that the (≤ 1)-SPM is simply connected, hence $F_1 = 1$. Now consider the graph G in which each node corresponds to either a hole (including the outer polygon) or a triple point, and there is an edge between two nodes if there is a 1-wall between the corresponding holes/triple points. Since the (≤ 1)-SPM is simply connected, G must be a tree. Hence $B_1 = h + V_1$. (The number of polygons bounding P is $h + 1$.) Furthermore note that the degree of a triple point in G is three, and every node in G has degree at least one. So, by double counting, $2B_1 \geq 3V_1 + h + 1$ or $V_1 \leq h - 1$. To summarize, $F_1 = 1$, $V_1 \leq h - 1$, and $B_1 = h + V_1$.

To bound the complexity of the $(\leq k)$-SPM for $k > 1$, we consider the k-homotopy sets $H_k(p)$. We use lower-case letters $a, b, c, \ldots$ to denote the members of $H_k(p)$. Each k-wall of the $(\leq k)$-SPM locally separates regions of P that differ in exactly one of their k shortest path homotopy classes. Note that a k-wall e of the $(\leq k)$-SPM is not present in the $(\leq k+1)$-SPM: if the k-homotopy sets belonging to the two sides of e are $H \cup a$ and $H \cup b$, with $a \neq b$, then the $(k+1)$-homotopy set of points in the neighborhood of e is uniformly $H \cup \{a, b\}$.

The triple points of the $(\leq k)$-SPM fall into two classes, which we call *new* and *old* (borrowing the terms from [14]). If the three k-homotopy sets in the vicinity of a triple point p are $H \cup a$, $H \cup b$, and $H \cup c$, with a, b, and c all distinct, then p is a new triple point. On the other hand, if the three k-homotopy sets are $H \cup \{a, b\}$, $H \cup \{b, c\}$, and $H \cup \{a, c\}$, with a, b, and c all distinct, then p is an old triple point. These names highlight the difference between what happens in the vicinity of p in the $(\leq k+1)$-SPM. If p is a new triple point in the $(\leq k)$-SPM, then it becomes an old triple point in the $(\leq k+1)$-SPM. The three $(k+1)$-walls incident to p in the $(\leq k+1)$-SPM separate points with $(k+1)$-homotopy sets $(H \cup a) \cup b$ from $(H \cup a) \cup c$, $(H \cup b) \cup a$ from $(H \cup b) \cup c$, and $(H \cup c) \cup a$ from $(H \cup c) \cup b$. If p is an old triple point in the $(\leq k)$-SPM, then the $(k+1)$-homotopy set of points in the neighborhood of e is uniformly $H \cup \{a, b, c\}$, and hence p is in the interior of a face of the $(\leq k+1)$-SPM. See Fig. 5.

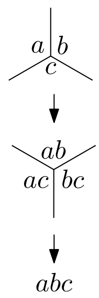


Figure 5: Life cycle of a triple point.

To transform the $(\leq k)$-SPM to the $(\leq k+1)$-SPM, we consider shortest distances to points in each face f of the $(\leq k)$-SPM from its k-walls. The distances from a particular k-wall e are measured according to the homotopy class belonging to the face on the opposite side of e from f. More concretely, let $p \in f$ be a point close to e, and let p' be on the other side of f. Then the shortest paths measured from e use the homotopy class $h_f(e) = H_k(p') \setminus H_k(p)$. For every point $q \in f$, we identify the k-wall e whose homotopy class $h_f(e)$ gives the shortest path to q. Hence $H_{k+1}(q) = H_k(q) \cup h_f(e)$, and this partitions the face f into subfaces, one for each k-wall e, separated by $(k+1)$-walls. To finish the construction of the $(\leq k+1)$-SPM, we erase the k-walls on the boundary of f (recall that their neighborhoods have uniform $(k+1)$-homotopy sets), delete any old triple points whose neighborhoods have uniform $(k+1)$-homotopy sets, and erase any newly added $(k+1)$-walls incident to deleted old triple points on the boundary of f. (These "walls" are actually just windows generated by the triple points; they separate regions with equal $(k+1)$-homotopy sets).

If a face f of the $(\leq k)$-SPM is bounded by B k-walls, it is initially partitioned into B subfaces. Every pair of subfaces incident to a common old triple point will be merged, so the final number of subfaces is $F' = B - W$, where W is the number of old triple points of the $(\leq k)$-SPM on the boundary of f. Since f is simply connected by Lemma 4.7, and every subface corresponding to a k-wall e must be adjacent to e, the dual graph of the subfaces inside f must be an outerplanar graph. The number of triple points V' added inside f (all of them new) corresponds to the number of (triangular) faces of this outerplanar graph, and hence $0 \le V' \le \max(F'-2, 0)$. By Euler's formula, the number of $(k+1)$-walls created inside f (duals to the edges of the outerplanar graph) is $B' = F' - 1 + V'$.

During the iterative construction of the $(\le k)$-SPM, we track the number of features of the $(\le k)$-SPM at each step. Let F_i and B_i be the number of faces and i walls in the $(\le i)$-SPM. To distinguish between new and old triple points, let V_i and W_i be the number of new and old triple points of $(\le i)$-SPM, respectively. Note that $W_1 = 0$.

The description above considers what happens within a single face of the $(\le k)$-SPM during the transformation to the $(\le k+1)$-SPM. To account for what happens in all the faces simultaneously, we note that each i-wall is shared between two faces, and each triple point is shared between three faces. Thus, if we count just the features added inside faces of $(\le i)$-SPM, using primed notation, we have

$$\begin{aligned} F'_{i+1} &= 2B_i - 3W_i \\ B'_{i+1} &= 2B_i - 3W_i - F_i + V'_{i+1} \\ V'_{i+1} &\le 2B_i - 3W_i - 2F_i \\ W'_{i+1} &= 0 \end{aligned}$$

Now let us take into account the deletion of previous i-walls and triple points. All the i-walls and old triple points are deleted between one phase and the next. All new triple points turn into old ones. All subfaces incident to an old triple point merge into one. Thus we obtain the following recurrence relations.

$$\begin{aligned} F_{i+1} &= F'_{i+1} - B_i + W_i \;=\; B_i - 2W_i & F_1 &= 1 \\ B_{i+1} &= B'_{i+1} \;=\; 2B_i - 3W_i - F_i + V_{i+1} & V_1 &\le h-1 \\ V_{i+1} &= V'_{i+1} \;\le\; 2B_i - 3W_i - 2F_i & B_1 &= h + V_1 \\ W_{i+1} &= V_i & W_1 &= 0 \end{aligned}$$

Lemma 4.8. *The number of faces, walls, and triple points of the $(\le k)$-SPM is $O(k^2 h)$.*

We now return to the complexity of the k-SPM. The number of k-walls and $(k-1)$-walls can be bounded by Lemma 4.8. Each k-wall consists of one or more hyperbolic arcs. Note that the number of hyperbolic arcs for a single k-wall is exactly one more than the number of k-windows that end on the k-wall (and a k-window can end on only one k-wall). Hence it is sufficient to count the number of k-windows. Each k-window is an extension of the edge between a vertex v of P and its i-predecessor for $i \le k$. Thus there can be at most $O(kn)$ k-windows.

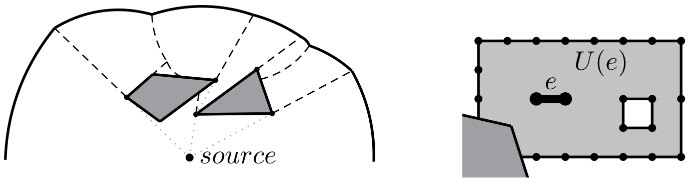
Theorem 4.9. *The k-SPM of a polygonal domain with n vertices and h holes has complexity $O(k^2 h + kn)$.*

4.3 Computing the k-SPM

We now describe how to compute the k-SPM in $O((k^3 h + k^2 n) \log(kn))$ time. Inspired by the structure of the k-garage and Lemma 4.3, our algorithm iteratively computes the k-SPM for increasing values of k, starting from $k = 1$. Essentially we compute the SPM on the k-garage, one floor at a time. To compute the k-SPM at each iteration, we apply the "continuous Dijkstra" method, which Hershberger and Suri [12] used to compute the shortest path map among polygonal obstacles. We adopt most of the details of the Hershberger–Suri algorithm unchanged; however, we also introduce several modifications to the algorithm to support k-SPM computation.

We begin our description with a brief overview of the continuous Dijkstra method. The main idea is to simulate the progress of a wavefront that emerges from the source and expands through the free space with unit speed. If the wavefront reaches a point p at time t, then the shortest path distance between p and the source is t. At any time, the wavefront consists of circular arc *wavelets*, each of which emanates from an obstacle vertex called a *generator*, which serves as an intermediate source with a delay (see Fig. 6a). In particular, a generator γ is represented as a pair (v, w), where v is an obstacle vertex and w is a positive real weight, equal to the shortest path distance from the source to v. For a generator $\gamma = (v, w)$ and a point p such that the segment $\overline{vp}$ is contained in free space, the (weighted) distance between γ and p, denoted $d(p, \gamma)$, is defined as $w + |\overline{vp}|$; it represents the length of the shortest path from the source to p that passes through v.

Points in the wavelet corresponding to a generator γ at time t satisfy the equation $d(p, \gamma) = t$. We



(a) (b)

Figure 6: (a) An expanding wavefront. (b) The well-covering region $\mathcal{U}(e)$ (light gray) for an edge e in the conforming subdivision.

say that a point p is *claimed* by γ if γ is the generator whose wavelet first reaches p; this implies that the shortest path to p passes through v and has length $d(p, \gamma)$. The points where adjacent wavelets on the wavefront meet trace out the bisectors that form the walls and the windows of the shortest path map. Each bisector separates two cells of the shortest path map, each of which consists of points claimed by a particular generator. The bisector curve separating the regions claimed by two generators γ and γ' satisfies the equation $d(p, \gamma) = d(p, \gamma')$. Because $|vp| - |v'p| = w' - w$, the curve is a hyperbolic arc.

The Hershberger–Suri algorithm simulates the wavefront expansion on a "conforming subdivision" of the free space. Each internal (free-space) edge e of this subdivision is contained in a set of cells whose union is called the "well-covering region" of e and denoted by $\mathcal{U}(e)$. (See Fig. 6b.) Briefly, the wavefront simulation computes the wavefront passing through each internal subdivision edge. The wavefront for a subdivision edge e is computed by propagating and combining the already computed wavefronts on the edges bounding $\mathcal{U}(e)$.[1] Once the wavefronts for all edges have been computed, the shortest path map in each subdivision cell is constructed locally by computing a weighted Voronoi diagram for the generators that claim the boundaries of the cell or are inside the cell. These cell-wide maps are then easily combined into a global shortest path map.

The Hershberger–Suri algorithm also works for shortest paths from multiple sources with delays. This is summarized in the following lemma, which was proved in [12].

Lemma 4.10 ([12]). *Given a set of polygonal obstacles with n vertices and a set of $O(n)$ sources with delays, one can compute the corresponding shortest path map in $O(n \log n)$ time.*

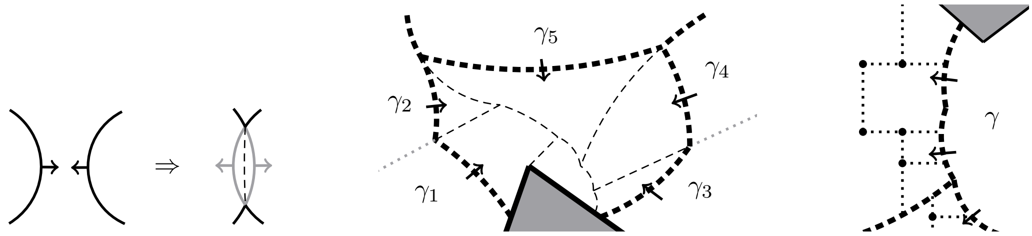
Within the framework of the Hershberger–Suri method, we can now explain our algorithm for computing the k-SPM. Conceptually, we apply the continuous Dijkstra framework on multiple floors of the k-garage. Imagine that we start a wavefront expansion from the source. When a wavelet collides with another wavelet during propagation (and thus forms a 1-wall), the portion of the wavelet that is claimed by the other wavelet continues to expand on the 2-floor (see Fig. 7a). Since this portion of the wavelet has passed through a 1-wall, it represents a set of 2-paths by Lemma 4.3. Any bisectors formed by adjacent wavelets on the 2-floor belong to the 2-SPM. Similarly to the 1-floor, when two wavelets collide on the 2-floor, they form a 2-wall and continue to expand on the 3-floor. We continue to push the colliding wavelets up to higher floors until they reach the k-floor, which will correspond to the k-SPM.

Notice that the wavefront expansion on a single floor is not affected by the expansion on another floor, with the exception of wavelet collisions on the previous floor. As the key step of our algorithm, we now describe a method that exploits this fact to compute the k-SPM once the $(k-1)$-SPM has been computed. This implies that we can construct the k-SPM by first running the Hershberger–Suri algorithm to compute the 1-SPM and then iteratively applying this step to compute higher floor SPMs.

We compute the k-SPM from $(k-1)$-SPM as follows. The boundaries of the $(k-1)$-SPM are formed by $(k-1)$-windows, $(k-1)$-walls and $(k-2)$-walls. The $(k-1)$-windows and $(k-2)$-walls do not appear in the k-SPM, so we simply remove them from the map. The $(k-1)$-walls remain in the map and they subdivide the free space into simply connected regions (by Lemma 4.7). To complete the k-SPM, in each such region we compute a special shortest path map whose walls and windows form the k-windows and k-walls of the k-SPM.

The shortest path map computed in each region R is drawn with respect to multiple "restricted"

[1]Well covering regions have special properties ensuring an acyclic propagation order between the edges of the subdivision.



(a) (b) (c)

Figure 7: (a) Two colliding wavelets. After the collision, a wall is formed and both wavelets continue to grow on the next floor. (b) A shortest path map is computed by propagating outside generators into the region R. (c) The set of subdivision edges in the vicinity of the $(k-1)$-walls through which a generator γ is propagated.

sources with delays, which are determined as follows. Consider a $(k-1)$-wall W bounding R in the $(k-1)$-SPM and let $\gamma = (v, w)$ be the generator that claims the region outside R in the vicinity of W. (It is possible that both sides of W are contained in R. In this case, our description applies to the generators claiming both sides.) Note that W is formed by the collision of the wavelet of γ with another wavelet, and the wavelet of γ is pushed up to the k-floor inside R. Conceptually, we want to continue expanding the wavelet of γ inside R. To do this, we introduce γ as a source at v with delay w and impose the additional restriction that all paths from γ to the interior of R pass through W.[2] In other words, we do not allow any paths from v that do not pass through W. We create sources in this manner for each $(k-1)$-wall bounding R and draw the shortest path map with respect to these sources (see Fig. 7b).

We can compute the shortest path map inside each region by running a single instance of the Hershberger–Suri algorithm for delayed sources; however, our restrictions necessitate some modifications. First, in order to divide the free space into the separate regions of interest, we treat the $(k-1)$-walls as obstacles. The original subdivision construction algorithm given in [12] assumes that the obstacles have straight boundaries, which may not hold for the $(k-1)$-walls. (Each $(k-1)$-wall consists of hyperbolic arcs.) We can easily overcome this issue by using a slightly modified algorithm that creates conforming subdivisions for "curved" obstacles (within the same complexity bounds). This modified algorithm was described in [13], where it was used to compute shortest paths among curved obstacles; we omit its details. Note that even though we are using a subdivision that may have curved edges, we still apply the wavefront propagation algorithm for polygons on this subdivision, because each curved edge resides on a $(k-1)$-wall whose claiming generator is already known. Thus, the curved edges do not take part in the wavefront propagation or yield additional generators, as they do in [13].

Our second modification to the shortest path algorithm is the initialization of wavefront propagation in the subdivision. The original algorithm of Hershberger and Suri starts the propagation by passing the wavefront directly from each source point s to all edges e whose well covering region $\mathcal{U}(e)$ contains s. The sources that we use are generators to be propagated into certain regions through certain $(k-1)$-walls, and thus we need a different way to initialize the wavefront. To meet our requirements, we initiate the wavefront propagation in the vicinity of the $(k-1)$-walls rather than the generators. In particular, the wavefront for a single generator γ is directly propagated to

(1) All edges e that bound a cell into which γ is to be propagated through a $(k-1)$-wall (see Fig. 7c).
(2) All edges e such that e contains an edge from (1) in its well-covering region $\mathcal{U}(e)$.

Note that propagating a generator's wavefront to an edge does not mean that the wavefront claims the edge, because some or all of the wavefront may be eliminated by other propagated wavefronts when they are merged to compute the final wavefront.

These modifications suffice to enable the Hershberger–Suri algorithm to compute the wavefronts passing through every edge in the conforming subdivision and the shortest path map in each region bounded by $(k-1)$-walls. Since the paths used to compute the map in each region are k-paths by Lemma 4.3, the walls and windows of the map form the k-walls and k-windows of the k-SPM. This completes the construction of the k-SPM.

[2] We also require that the the subpath between v and W is a straight line.

Theorem 4.11. *Given a source point in a polygonal domain with n vertices and h holes, the corresponding k-SPM can be computed in $O((k^3h + k^2n)\log(kn))$ time.*

5 Simple paths

Our definition of k-path allows the path to be self-crossing. This may be undesirable for many applications. In this section we show how to compute the kth shortest *simple* path (*simple k-path*) in polynomial time, albeit slower than when we allow self-crossing paths. Here we define a *simple path* as a path that does not cross itself, although repeated vertices and segments are allowed. Note that we cannot use one of our previous methods to solve this problem: the simple 3-path may be a k-path for arbitrarily high k.

As in Section 3, we consider only the most basic form of the problem, in which we are given a fixed target $t \in P$. For simple paths we need to treat s and t as point obstacles (otherwise pulling a path taut may introduce self-crossings), but this either trivializes the problem (the path may wind around s or t for free) or makes the algorithm more complex; therefore, for ease of presentation, we limit our attention to the case in which s and t are located on the boundaries of obstacles.

We again use the taut graph $\vec{G}(P)$ to reduce the problem to a graph problem. The taut graph ensures that every path between s and t is locally shortest, but it still allows crossings. To avoid crossings, we adapt Yen's algorithm [20] for simple k-paths in directed graphs (here "simple" means free of repeated nodes). Yen's algorithm first computes the shortest path, which must be simple; the same is true in our geometric setting. Next, the algorithm "expands" the shortest path π in the following way: It considers every possible prefix of π and chooses a next edge e that is different from the next edge in π. It then finds the shortest path starting from the endpoint of e that avoids the prefix including e; this ensures that the resulting path is simple and different from π. Such paths are computed for every possible prefix and edge e; the shortest such path is the simple 2-path. The algorithm continues by expanding the simple 2-path and repeats this process until the simple k-path is found.

Note that we cannot use Yen's algorithm directly on $\vec{G}(P)$, since a simple path in $\vec{G}(P)$ is not necessarily simple in the geometric sense. To make this algorithm work in our setting, we need to make one small modification. Before we compute the shortest path with a given prefix π_p (including e), we add π_p as an obstacle to P, obtaining a new polygon P'. We then work with the taut graph $\vec{G}(P')$ of the new polygon (we separate each vertex of π_p and the corresponding obstacle vertex by an infinitesimal amount to allow paths that abut π_p but do not cross it). We need to show that the locally shortest path with a given prefix, i.e., the shortest path in $\vec{G}(P')$ starting after e, is simple. Clearly π_p is simple, and the suffix cannot cross π_p, but it is not clear that the suffix itself is simple. Although it is not obvious due to the geometric nature of our paths, we can prove the following.

Lemma 5.1. *The shortest path in $\vec{G}(P')$ that starts with a fixed (simple) prefix π_p must be simple in P.*

Thus, if we compute $\vec{G}(P')$ before every shortest path computation, every path obtained by our adaptation of Yen's algorithm must be simple. We now obtain the following result.

Theorem 5.2. *The simple k-path between s and t can be computed in $O(k^2m(m + kn)\log kn)$ time, where m is the number of edges of the visibility graph of P.*

6 Concluding remarks

We have introduced the k-SPM, a data structure that can efficiently answer k-path queries. We provided a tight bound for the complexity of the k-SPM, and presented an algorithm to compute the k-SPM efficiently. Our algorithm simultaneously computes all the i-SPMs for $i \leq k$. Whether there is a more direct algorithm to compute the k-SPM is an interesting open problem. We also provided a simple visibility-based algorithm to compute k-paths, which may be of practical interest, and is more efficient for large values of k. This latter approach can be extended to compute simple k-paths. Unfortunately, we do not know how to extend the k-SPM to simple k-paths. It seems that simple k-paths lack the useful property that a subpath of a simple k-path is a simple i-path for $i \leq k$. This makes finding a more efficient algorithm to compute simple k-paths a challenging open problem.

References

[1] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua. Multiple object tracking using k-shortest paths optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(9):1806–1819, 2011.

[2] C. Bohler, P. Cheilaris, R. Klein, C.-H. Liu, E. Papadopoulou, and M. Zavershynskyi. On the complexity of higher order abstract Voronoi diagrams. In *ICALP (1)*, volume 7965 of *Lecture Notes in Computer Science*, pages 208–219. Springer, 2013.

[3] S. Cabello, Y. Liu, A. Mantler, and J. Snoeyink. Testing homotopy for paths in the plane. *Discrete & Computational Geometry*, 31:61–81, 2004.

[4] D. Z. Chen, J. Hershberger, and H. Wang. Computing shortest paths amid convex pseudodisks. *SIAM J. Comput.*, 42(3):1158–1184, 2013.

[5] D. Z. Chen and H. Wang. L_1 shortest path queries among polygonal obstacles in the plane. In *STACS*, volume 20 of *LIPIcs*, pages 293–304. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2nd edition, 2001.

[7] D. Eppstein. Finding the *k* smallest spanning trees. *BIT*, 32(2):237–248, 1992.

[8] D. Eppstein. Tree-weighted neighbors and geometric k smallest spanning trees. *Int. J. Comput. Geometry Appl.*, 4(2):229–238, 1994.

[9] D. Eppstein. Finding the k shortest paths. *SIAM J. Comput.*, 28(2):652–673, 1999.

[10] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics*. Addison-Wesley, Reading, Massachusetts, second edition, 1994.

[11] J. Hershberger, M. Maxel, and S. Suri. Finding the k shortest simple paths: A new algorithm and its implementation. *ACM Trans. Algorithms*, 3(4):45, 2007.

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

[13] J. Hershberger, S. Suri, and H. Yıldız. A near-optimal algorithm for shortest paths among curved obstacles in the plane. In *Proceedings of the Twenty-Ninth Annual Symposium on Computational Geometry*, SoCG '13, pages 359–368. ACM, 2013.

[14] D.-T. Lee. On k-nearest neighbor Voronoi diagrams in the plane. *IEEE Trans. Computers*, 31(6):478–487, 1982.

[15] C.-H. Liu and D. T. Lee. Higher-order geodesic Voronoi diagrams in a polygonal domain with holes. In *SODA*, pages 1633–1645. SIAM, 2013.

[16] C.-H. Liu, E. Papadopoulou, and D. T. Lee. The k-nearest-neighbor Voronoi diagram revisited. *Algorithmica*, 2014. To appear.

[17] J. S. B. Mitchell. Geometric shortest paths and network optimization. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 633–701. Elsevier Science B.V. North-Holland, Amsterdam, 2000.

[18] E. Papadopoulou and M. Zavershynskyi. On higher order Voronoi diagrams of line segments. In *ISAAC*, volume 7676 of *Lecture Notes in Computer Science*, pages 177–186. Springer, 2012.

[19] M. Pocchiola and G. Vegter. Topologically sweeping visibility complexes via pseudotriangulations. *Discrete & Computational Geometry*, 16(4):419–453, 1996.

[20] J. Y. Yen. Finding the K shortest loopless paths in a network. *Management Science*, 17:712–716, 1971.

A Handling Degeneracies and Tie-Breaking

For simplicity of analysis we assumed that P satisfies the following conditions:

1. No three of the vertices of P, including the source s, are collinear.

2. There are at most three homotopically different i-paths to a single point in P, for $1 \leq i \leq k$. Equivalently, no four i-walls meet at a single point.

3. There is a unique i-path to each vertex of P, for $1 \leq i \leq k$. Equivalently, no i-wall goes through a vertex of P.

With these assumptions all walls are one-dimensional curves that meet only at triple points.

We now describe briefly how to adapt our analysis if these assumptions are false. If we are dealing with first shortest paths only, then we can simply apply the standard technique of (symbolic) perturbation to the input (i.e., perturb the positions of the vertices) so that the input is in general position and satisfies all of the assumptions. However, for k-paths with $k \geq 2$, we need more than perturbation to enforce all assumptions. In particular, Assumption 3 cannot be enforced by perturbation because it can be violated even when the input is non-degenerate. For an example see Fig. 8: The 1-path from s to v is a straight line. There are two 2-paths from s to v, labeled π_1 and π_2. The paths π_1 and π_2 are homotopically different; they pass through v first and then loop around the same obstacle in different directions to return to v. Both π_1 and π_2 have the same length, and thus v is on the 2-wall. This implies that v and all of the points to its left below ray r have two distinct 2-paths and thus belong to a 2-wall; the 2-wall is thus a region, not a curve.

In order to avoid this issue, we introduce a tie-breaking mechanism between the paths so that all paths to an obstacle vertex are strictly ordered by length and thus each obstacle vertex has a unique i-path. In particular, suppose that π_1 and π_2 are two i-paths from s to a vertex v with the same length. We break the tie between π_1 and π_2 by arbitrarily assuming that one of the two paths is infinitesimally shorter than the other. Conceptually, this mechanism perturbs the i-wall by moving it slightly to one side. As a result, the i-wall does not go through v and Assumption 3 is satisfied. Once the tie is broken, we assume that all paths that are obtained by extending π_1 and π_2 with the same subpath preserve this order, maintaining consistency.[3]

By applying (symbolic) perturbation and enforcing a strict virtual order between the paths via tie-breaking, we guarantee all our assumptions.

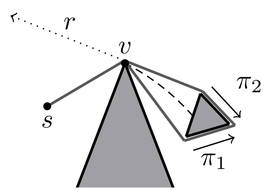


Figure 8: The equal-length paths π_1 and π_2 are both 2-paths to v. The 2-wall is shown with a dashed line.

[3]This still applies even if there are other tie-breakings in the extending subpath.

B Omitted Proofs

Lemma 4.1. *A sequence σ cannot be both a k-sequence and an ℓ-sequence if $k \neq \ell$.*

Proof. Assume without loss of generality that $\ell < k$. The definition of a k-sequence directly implies the following properties: (i) A k-sequence contains all integers in $\{1, \ldots, k-1\}$, and (ii) every tail of a k-sequence is an i-sequence for some $i \leq k$.

Let k be the smallest number for which the lemma does not hold; clearly $k > 1$. If $\ell = 1$, then σ does not contain 1 while a k-sequence must contain 1 (property (i)); so assume $\ell > 1$. Since $k > \ell$, σ must contain ℓ (property (i) again). By definition, the tail of σ after one of the occurrences of ℓ is an ℓ-sequence. Since σ is also an ℓ-sequence, it must contain $(\ell - 1)$ before ℓ, and the tail of σ after $(\ell - 1)$ is an $(\ell - 1)$-sequence. In particular, the tail of σ after the occurrence of ℓ mentioned above must also be an i-sequence for some $i \leq \ell - 1$ (property (ii)). But then the lemma does not hold for $k = \ell, \ell = i$, contradicting our choice of k. □

Lemma 4.3. *If π is the shortest path in the k-garage from s on the 1-floor to some t on the k-floor, then $\pi^{\downarrow}$ is a k-path to t.*

Proof. We show that the crossing sequence of $\pi^{\downarrow}$ is a k-sequence. Then, by Lemma 4.2, $\pi^{\downarrow}$ is a k-path. We again use the property that every tail of a k-sequence is an i-sequence for some $i \leq k$. If, going back from t to s, π only goes "down" in the k-garage, then it is easy to see that the crossing sequence of $\pi^{\downarrow}$ is a k-sequence. (Because regions on the i-floor are bounded by $(i-1)$- and i-walls, π enters the i-floor by crossing an i-wall and does not cross any i-wall before it exits the i-floor by crossing an $(i-1)$-wall. Thus the tail of π's crossing sequence that starts from any point on the i-floor is an i-sequence.) For the sake of contradiction, assume that π also goes up in the k-garage. Then there must be a point where π goes up to some i-floor, and then goes monotonically down to the 1-floor. The crossing sequence of the corresponding subpath of $\pi^{\downarrow}$ must be of the form $\sigma = (i-1, \sigma_i)$, where σ_i is an i-sequence. If σ is a j-sequence for $j \neq i$, then σ_i must be a j-sequence, which is not possible by Lemma 4.1. If σ is an i-sequence, then σ_i must be an $(i-1)$-sequence, which again is not possible by Lemma 4.1. Finally note that σ must be a j-sequence for some j, since $\pi^{\downarrow}$ is locally shortest. Thus, π only goes down in the k-garage, and the crossing sequence of $\pi^{\downarrow}$ must be a k-sequence. □

Lemma 4.4. *If $\epsilon < |q - p_i|$ for $i \in \{1, 2, 3\}$, then $|q_{abc} - q| < \epsilon$.*

Proof. Points p_1, p_2, and p_3 are the vertices of an equilateral triangle, with q at its center. Define $L = |q - p_1|$. By assumption, $L > \epsilon$. Since $0 \leq l_{ij} - l_{i1} \leq \epsilon$ for $i \in \{1, 2, 3\}$ and any $1 \leq j \leq k$, and

$$|q_{abc} - p_1| + l_{1a} = |q_{abc} - p_2| + l_{2b} = |q_{abc} - p_3| + l_{3c},$$

we have $|q_{abc} - p_i| \leq |q_{abc} - p_j| + \epsilon$ for any i and j. The locus of points satisfying these inequalities is bounded by six hyperbolic arcs, as shown in Fig. 9. Each arc bulges toward the center, so putting q_{abc} at a vertex of the region maximizes $|q_{abc} - q|$. There are two classes of vertices of the region. They are defined by intersections of hyperbolae arranged in three pairs along the three angle bisectors at p_1,

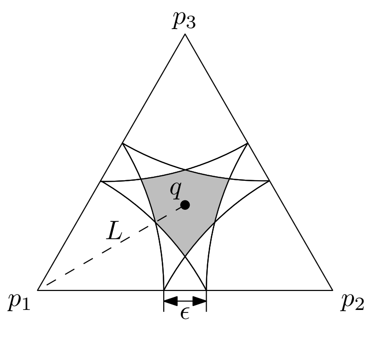


Figure 9: q_{abc} lies in the region around q.

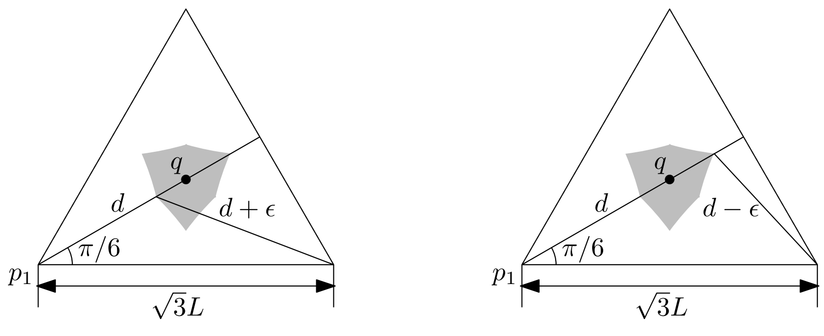


Figure 10: Extreme locations of q_{abc}.

p_2, and p_3. By symmetry we can solve for points lying on an angle bisector satisfying the difference relations shown in Fig. 10. We apply the law of cosines to find minimum and maximum values of d, the distance from any of the p_i to the intersections of hyperbolae on the angle bisector at p_i. Solving for the lower bound on d (Fig. 10(left)), we have

$$\begin{aligned} d^2 + 3L^2 - 2d\sqrt{3}L\cos\frac{\pi}{6} &= (d+\epsilon)^2 \\ 3L^2 - 3dL &= 2d\epsilon + \epsilon^2 \\ d = \frac{3L^2 - \epsilon^2}{3L + 2\epsilon} &= L - \frac{2}{3}\epsilon + \frac{\epsilon^2}{3(3L+2\epsilon)} \\ &> L - \frac{2}{3}\epsilon. \end{aligned}$$

Solving for the upper bound (Fig. 10(right)), we have

$$\begin{aligned} d^2 + 3L^2 - 2d\sqrt{3}L\cos\frac{\pi}{6} &= (d-\epsilon)^2 \\ 3L^2 - 3dL &= -2d\epsilon + \epsilon^2 \\ d = \frac{3L^2 - \epsilon^2}{3L - 2\epsilon} &= L + \frac{2}{3}\epsilon + \frac{\epsilon^2}{3(3L-2\epsilon)} \\ &< L + \epsilon \end{aligned}$$

since $L > \epsilon$. Because q_{abc} is constrained to lie in this hyperbolically bounded region, and the maximum distance from q to the boundary of the region is less than ϵ, we have $|q_{abc} - q| < \epsilon$. □

Theorem 4.5. *The k-SPM of a polygonal domain with n vertices and h holes can have $\Omega(k^2h)$ k-walls and $\Omega(kn)$ k-windows.*

Proof. From the discussion in Section 4.2 it directly follows that the k-SPM of the example has $\Omega(k^2h)$ k-walls. Hence we only need to show that the k-SPM can have $\Omega(kn)$ k-windows. Since we can make the number of vertices in the convex chain at p_3 arbitrarily large, it is sufficient to show that each vertex in the chain (except the first) contributes k k-windows to the k-SPM. Let e_j be the edge formed by extending the edge between v_j and v_{j+1} toward q until it hits the boundary of P. We claim that, for every $i \leq k$, there must be a point $t \in e_j$ such that the path π consisting of the i-path to v_j followed by the segment $\overline{v_jt}$ is the k-path from s to t. If t is at v_j, then π is an i-path by definition. If t is the other endpoint of e_j and e_j is sufficiently close to q, then π must be an ℓ-path for $\ell > k$. Lemma 4.2 now implies that there must be a $t \in e_j$ such that π is the k-path from s to t. Thus, each vertex in the convex chain (except the first) contributes k k-windows, and the k-SPM has $\Omega(kn)$ k-windows. □

Lemma 4.6. *If p and p' lie in the same cell of the $(\leq k)$-SPM, and π is a path between p and p' that does not cross a k-wall, then $H_k(p) \oplus \pi = H_k(p')$.*

Proof. We reuse ideas from the proof of Lemma 4.2. Let us assume that distances have been scaled so that the length of π is 1. Define $p(x)$ ($0 \leq x \leq 1$) as the point on π such that the distance from p to $p(x)$ along π is x. Let $\gamma(x)$ be the subpath of π from p to $p(x)$. Furthermore, let π_i be the i-path to p, and let $\pi'_i(x)$ be the locally shortest path homotopic to the concatenation of π_i and $\gamma(x)$. The length of $\pi'_i(x)$ is denoted by $l_i(x)$ for $0 \leq x \leq 1$. Note that $l_i(0) < l_j(0)$ for $i < j$. If $l_i(x) \neq l_j(x)$ for all $0 \leq x \leq 1$ and $i \leq k < j$, then it is clear that $H_k(p) \oplus \pi = H_k(p')$. For the sake of contradiction, let x^* be the smallest x such that $l_i(x^*) = l_j(x^*)$ for some $i \leq k < j$. Let r be the number of graphs that pass below this intersection. If $r = k - 1$, then $p(x^*)$ is on a k-wall, which is a contradiction. If $r < k - 1$, then there must be an $m \leq k$ such that $l_m(x^*) > l_j(x^*)$. But that means that $l_m(x) = l_j(x)$ for some $x < x^*$, contradicting the choice of x^*. Similarly, if $r > k - 1$, then there must be an $m > k$ such that $l_m(x^*) < l_i(x^*)$. But that means that $l_m(x) = l_i(x)$ for some $x < x^*$, again contradicting the choice of x^*. □

Lemma 4.7. *The cells of the* $(\leq k)$*-SPM are simply connected.*

Proof. For the sake of contradiction, assume there is a cell of the $(\leq k)$-SPM that is not simply connected. Let C be a cycle in this cell that is not contractible. If C contains only k-walls, then there must be a triple point with an angle larger than 180 degrees, which is not possible (a triple point is a Voronoi vertex of an additively weighted Voronoi diagram). Hence there must be an obstacle ω in C. Let $p \in C$ and let the largest winding number of any path in $H_k(p)$ with respect to ω be r. By Lemma 4.6 we have $H_k(p) \oplus C = H_k(p)$, where C is followed in counterclockwise direction. However, $H_k(p) \oplus C$ must contain a path with winding number $r + 1$. This is a contradiction. □

Lemma 4.8. *The number of faces, walls, and triple points of the* $(\leq k)$*-SPM is* $O(k^2 h)$*.*

Proof. We express the recurrence relations and the initial values using generating functions, which are formal power series with the sequence values as coefficients [10]. In general, for a sequence of values g_i, the generating function $g(z)$ is

$$g(z) = \sum_{i \geq 0} g_i z^i.$$

For our sequences, we have

$$\begin{aligned} F(z) &= zB(z) - 2zW(z) + z \\ B(z) &= z\,(2B(z) - 3W(z) - F(z)) + V(z) + zh \\ V(z) &\leq z\,(2B(z) - 3W(z) - 2F(z)) + z(h-1) \\ W(z) &= zV(z) \end{aligned}$$

Note that the constant term is zero, because we assume $F_0 = V_0 = B_0 = W_0 = 0$.

For convenience we will leave the "z" argument of the functions implicit during our manipulations. We can immediately eliminate the function $W = zV$:

$$\begin{aligned} F &= zB - 2z^2V + z \\ B &= z(2B - 3zV - F) + V + zh \\ V &\leq z(2B - 3zV - 2F) + z(h-1) \end{aligned}$$

Next we substitute $F = zB - 2z^2V + z$ into the last two relations to obtain

$$\begin{aligned} B &= z(2B - 3zV - (zB - 2z^2V + z)) + V + zh \\ V &\leq z(2B - 3zV - 2(zB - 2z^2V + z)) + z(h-1) \end{aligned}$$

or, combining terms,

$$\begin{aligned} (1 - 2z + z^2)B &= (1 - 3z^2 + 2z^3)V + z(h - z) \\ (1 + 3z^2 - 4z^3)V &\leq (2z - 2z^2)B - 2z^2 + z(h-1) \end{aligned}$$

Substituting

$$B = V\frac{(1-3z^2+2z^3)}{(1-z)^2} + \frac{z(h-z)}{(1-z)^2}$$

into the inequality for V, we obtain

$$\begin{aligned}(1+3z^2-4z^3)V &\leq V\,\frac{2z(1-z)(1-3z^2+2z^3)}{(1-z)^2} \\ &\qquad +\frac{2z^2(1-z)(h-z)}{(1-z)^2} - 2z^2 + z(h-1) \\ &= 2z(1+z-2z^2)V + \frac{2z^2(h-z)}{1-z} - 2z^2 + z(h-1)\end{aligned}$$

Rearranging terms and simplifying, we obtain

$$V \leq \frac{z(1+z)(h-1)}{(1-z)^3}.$$

Recall that $(1-z)^{-3} = \sum_{i\geq 0}\binom{i+2}{2}z^i$, and hence

$$\begin{aligned}V &\leq \frac{z(1+z)(h-1)}{(1-z)^3} \\ &= \sum_{i\geq 1} z^i(h-1)\left[\binom{i+1}{2}+\binom{i}{2}\right] \\ &= \sum_{i\geq 0} z^i(h-1)i^2.\end{aligned}$$

Returning from the domain of generating functions to our original recurrence relations, we have

$$V_i \leq (h-1)i^2,$$

which immediately implies

$$W_i = V_{i-1} \leq (h-1)(i-1)^2.$$

Solving for $B(z)$ instead of $V(z)$ gives

$$B_i \leq (h-1)(3i^2-3i+2)+1.$$

Finally, using $F_i = B_{i-1} - 2W_{i-1} \leq B_{i-1}$, we get

$$F_i \leq (h-1)(3i^2-9i+8)+1.$$

□

Theorem 4.11. *Given a source point in a polygonal domain with n vertices and h holes, the corresponding k-SPM can be computed in $O((k^3h+k^2n)\log(kn))$ time.*

Proof. We construct the k-SPM iteratively for increasing values of k as described. We argue that at each iteration, the time spent to construct the k-SPM from a given $(k-1)$-SPM is $O((k^2h+kn)\log(kn))$. This implies the total time spent is $O((k^3h+k^2n)\log(kn))$.

By Theorem 4.9, the complexity of the $(k-1)$-SPM is $O(k^2h+kn)$. We construct the k-SPM by running the modified Hershberger–Suri algorithm as described above. The algorithm is run on a set of obstacles with $O(k^2h+kn)$ vertices (including the original obstacle vertices and the endpoints of the hyperbolic arcs forming the $(k-1)$-walls) with $O(k^2h+kn)$ delayed sources (at most two sources per hyperbolic arc). By Lemma 4.10 (which applies also to our modified algorithm), the algorithm completes in $O((k^2h+kn)\log(k^2h+kn)) = O((k^2h+kn)\log(kn))$. This completes the proof. □

Before we can prove Lemma 5.1, we need some additional results.

Let π_{pq} denote the subpath of a path π between two points $p, q \in \pi$. We can apply a *shortcut* to a path π by replacing π_{pq} by the straight segment $\overline{pq}$, so long as $\overline{pq}$ lies in free space. A shortcut is *valid* if it does not change the homotopy class of the path. We assume that a valid shortcut $\overline{pq}$ does not cross π_{pq}, for otherwise we can cut up the shortcut into multiple smaller shortcuts. A shortcut is valid if and only if the cycle formed by π_{pq} and $\overline{pq}$ does not contain an obstacle. Note that a locally shortest path has no valid shortcuts. Furthermore, we can make a path locally shortest by repeatedly applying valid shortcuts until no more valid shortcuts exist.

A path π is *x-monotone* if every vertical line crosses π only once. Given a path π in P, we can obtain π' by repeatedly applying valid vertical shortcuts to π until no more valid vertical shortcuts exist. We call π' the *vertical reduction* of π. We can then find the smallest set S of vertices of P along π' such that the subpath of π' between two adjacent (along π') vertices in S is x-monotone. We call the vertices in S the *extremal vertices* of π'.

Now consider two homotopic paths π_1 and π_2 and their vertical reductions π'_1 and π'_2. As was shown in [3, Lemmas 1 and 7], the set of extremal vertices of π'_1 and π'_2 must be the same. Hence the set of extremal vertices depends only on the homotopy class of π_1, and we can also speak of the extremal vertices of π_1. Finally note that a locally shortest path is its own vertical reduction. Thus the locally shortest path homotopic to a path π must pass through the extremal vertices of π.

Lemma 5.1. *The shortest path in $\vec{G}(P')$ that starts with a fixed (simple) prefix π_p must be simple in P.*

Proof. For the sake of contradiction, assume that the shortest path π with fixed prefix π_p crosses itself at the point $x \in \pi$ on edge e^*, where e^* is the first crossing edge after π_p. (See Fig. 11a.) Assume w.l.o.g. that the bend at the vertex v before e^* makes a right turn. We can rotate the polygonal domain so that the direction of e^* is infinitesimally clockwise from vertically up. As a result, v is an extremal vertex of π.

We will show that there is a locally shortest path π' that is shorter than π and also makes a right turn at v. Since a locally shortest path must turn toward obstacles, it is sufficient to show that π' is shorter and passes through v. We first construct a path π'' that is not longer than π, and then let π' be the locally shortest path homotopic to π'', which is shorter than π.

The path π (from s to t) crosses e^* either (i) from left to right (as in Fig. 11a) or (ii) from right to left (as in Fig. 11c). Let π^* be the subpath of π between the two occurrences of the crossing. In case (i) π'' is obtained by eliminating π^*. (See Fig. 11b.) In case (ii) π'' is obtained by reversing π^*. (See Fig. 11d.) In case (i) π'' is clearly shorter than π. In case (ii) π'' has the same length as π, but note that π' must then be shorter.

In both cases π'' makes a right turn at x. Now note that every vertical shortcut of π'' must also exist in π. To see that, notice that the only shortcuts of π' we need to consider are those that span π^* in case (i) or span or touch π^* in case (ii); any other shortcut also exists in π. A vertical shortcut that connects any point before π^* to a point on or after π^* is blocked by v (i.e., the shortcut is not valid). A shortcut of π' within π^* must also exist in π. A shortcut from a point on π^* to point after π^* (in case (ii)) is blocked by the first extremal vertex after π^*. Since every vertical shortcut of π'' exists in π and π is locally shortest

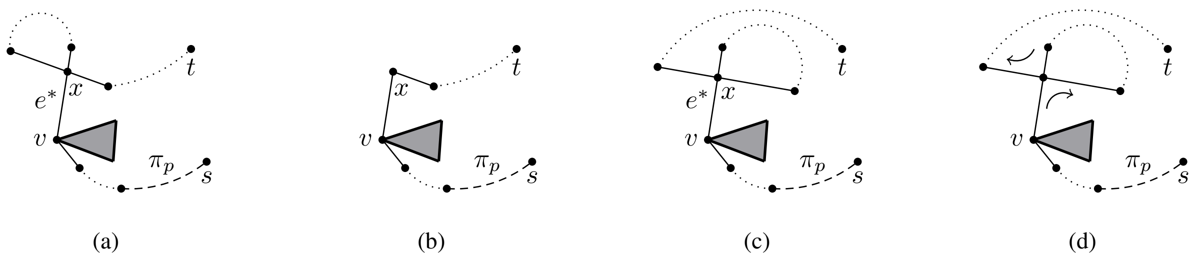


Figure 11: (a) π crosses e^* from left to right. (b) π'' is obtained by eliminating π^*. (c) π crosses e^* from right to left. (d) π'' is obtained by reversing π^*.

(i.e. has no valid shortcuts), π'' must be its own vertical reduction. Thus, v is an extremal vertex of π'', and π' must pass through v.

Finally we need to show that π' is actually a path in $\vec{G}(P')$. Note that $\vec{G}(P')$ contains all locally shortest paths in P that do not cross the fixed prefix π_p. So it is sufficient to show that π' does not cross π_p. Since π did not cross π_p, the same is true for π''. We can obtain π' from π'' by repeatedly applying valid shortcuts. It is now sufficient to show that any valid shortcut $\overline{pq}$ between $p, q \in \pi''$ cannot cross π_p. For the sake of contradiction, assume that $\overline{pq}$ crosses π_p. That means that some part of π_p must go inside the cycle C formed by $\overline{pq}$ and π''_{pq}. Note that s is outside C since we assumed that s belongs to an obstacle. If π_p ends inside C, then there must be an obstacle inside C, which means that the shortcut was not valid. Otherwise, π_p must also leave C. It cannot leave through π''_{pq}, since π'' did not cross π_p. If it leaves C through $\overline{pq}$, then there must be a bend inside C. But this again means that there is an obstacle inside C, which contradicts the validity of the shortcut.

Thus, the path π' contains π_p, it exists in $\vec{G}(P')$, and it is shorter than π. This contradicts the choice of π. □

Theorem 5.2. *The simple k-path between s and t can be computed in $O(k^2 m(m + kn) \log kn)$ time, where m is the number of edges of the visibility graph of P.*

Proof. The simple k-path has at most kn edges since each vertex of P can be visited at most k times. This means that a simple k-path can have at most $O(km)$ prefixes (including e). To compute $\vec{G}(P')$, note that every visibility edge of P' is also a visibility edge of P, although some edges may occur multiple times in P' (edges of P in the prefix are duplicated). Hence, to compute P', we need to understand which visibility edges of P still exist in P'. By considering the prefixes in order of increasing length (one edge at a time), we only need to check which visibility edges of P cross the last edge of the prefix, which can be computed in $O(m)$ time per prefix. Since the prefix can have at most kn edges, the visibility graph of P' can have at most $O(m + kn)$ edges. We can then compute $\vec{G}(P')$ in $O(m + kn)$ time. Finally, we can use Dijkstra's algorithm [6] to compute the shortest path in $\vec{G}(P')$ after the prefix in $O((m + kn) \log kn)$ time. To obtain the simple k-path, we need to expand $k - 1$ paths. Each path may have $O(km)$ prefixes, and the shortest path for each prefix can be computed in $O((m+kn) \log kn)$ time. Thus, we can compute the simple k-path in $O(k^2 m(m + kn) \log kn)$ time. □