Shortest Paths in the Plane with Obstacle Violations

John Hershberger[1], Neeraj Kumar[2], and Subhash Suri[3]

1   **Mentor Graphics Corp., Wilsonville, OR, USA**
     `john_hershberger@mentor.com`
2   **University of California, Santa Barbara, CA, USA**
     `neeraj@cs.ucsb.edu`
3   **University of California, Santa Barbara, CA, USA**
     `suri@cs.ucsb.edu`

─── **Abstract** ───────────────────────────────────────

We study the problem of finding shortest paths in the plane among h convex obstacles, where the path is allowed to pass through (violate) up to k obstacles, for $k \leq h$. Equivalently, the problem is to find shortest paths that become obstacle-free if k obstacles are removed from the input. Given a fixed source point s, we show how to construct a map, called a *shortest k-path map*, so that all destinations in the same region of the map have the same combinatorial shortest path passing through at most k obstacles. We prove a tight bound of $\Theta(kn)$ on the size of this map, and show that it can be computed in $O(k^2 n \log n)$ time, where n is the total number of obstacle vertices.

1   Introduction

Given a set of polygonal obstacles in the plane and an integer parameter k, which k obstacles should we remove to obtain the shortest obstacle-free path between two points s and t? Equivalently, what is the shortest path that is allowed to violate (pass through) up to k obstacles? We call a path violating at most k obstacles a *k-path*, generalizing a traditional obstacle-free path, which is a 0-path. More precisely, we assume a polygonal environment P containing h disjoint convex obstacles in the plane, with a total of n vertices, all lying inside a rectangle R (the outer boundary). The complement of the obstacles within R is called *free space*. Given a fixed source point s in free space, we want to compute shortest k-paths, for $k \leq h$, to all other points of free space. The description of these shortest paths can be compactly encoded as a finite partition of the plane, called the *shortest k-path map*. We use the notation $\pi_k(t)$ to denote the shortest k-path from s to t, with the fixed source s being implicit, and denote the length of this path by $d_k(t)$.

In this paper, we investigate structural and computational aspects of shortest k-paths. The problem differs from the 0-path problem in nontrivial ways even in the plane. In particular, two shortest 0-paths originating at a common source cannot intersect, by the triangle inequality, and this non-crossing property of 0-paths is an essential ingredient for computing them in optimal time [15]. In contrast, two shortest k-paths can cross each other, for any $k > 0$. The geometric k-path problem is interesting both theoretically, as

part of the broad category of optimization with violations [6, 21] or network augmentation problems [2, 10], and practically, for applications such as robot motion planning, where it may be beneficial to modify a robot's environment to shorten frequently used paths. (The geometric k-path problem can be seen as a more complex form of network augmentation, since removal of a single obstacle can create many additional "edges" in the path space.) Besides robot motion planning, the problem can also model situations in which the obstacles are "avoidable" at additional cost, for instance by paying a bridge or tunnel toll in a road network.

Our approach to solving the k-path problem is to compute a *shortest k-path map SPM_k*, which is a partition of the plane into equivalence classes of cells (regions), where all destination points inside a cell have the same combinatorial structure of shortest k-paths to s. Once the map is known, the shortest k-path to any destination can be computed by performing a point location query on the map [8, 18].

Our Results.   We show that SPM_k has $O(kn)$ regions and $O(kn)$ edges and that this bound is tight (Section 3). We present an $O(k^2 n \log n)$ time algorithm for computing SPM_k (Section 4), using the continuous Dijkstra framework, which constructs each SPM_j for $0 \leq j \leq k$ sequentially. The running time of the algorithm is optimal for $k = O(1)$. Due to space limitations, some of the proofs are omitted from this version of the paper.
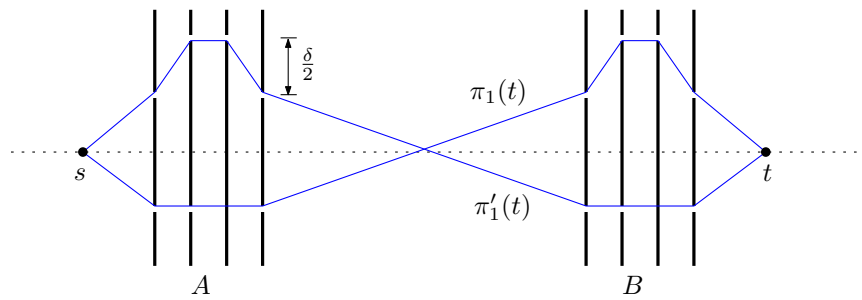
Related Work.   The problem of computing shortest paths in the presence of obstacles has a long history in computational geometry, dating back to the 1970s. The case of polygonal obstacles in the plane, in particular, has been a subject of intense research [3, 4, 11, 17, 22, 23, 25, 26, 28], culminating in an optimal $O(n \log n)$ time algorithm using the continuous Dijkstra framework [15]. Many other variations of the problem, including shortest paths inside a simple polygon [12, 14, 19], among weighted regions [24], and among curved obstacles [7, 16], have also been studied. The general flavor of our problem is related to geometric optimization where a small number of constraints can be violated. This line of work has been pursued in [6, 13, 21, 27], in the context of low-dimensional linear programming, separability with outliers, and geometric optimization. Our problem can also be viewed as a form of network augmentation, where the goal is to add edges to the network to improve connectivity, diameter, or spanning ratio etc. [1, 2, 5, 10].

The prior work most closely related to our research is a recent result by Maheshwari et al. [20], which presents an $O(n^3)$ time algorithm for computing the 1-violation path inside a simple polygon: that is, a shortest path inside a simple n-gon that is allowed to leave the polygon once. Our work deals with finding k-violation paths, for arbitrary k, in an environment containing possibly $O(n)$ convex obstacles.

2   Properties of k-paths

Given a point p in free space, a shortest k-path $\pi_k(p)$ connects s to p, crosses the interiors of at most k obstacles, and has minimum length among all such paths. On occasion, we also need to reason about paths crossing *exactly k* obstacles, and we refer to such a path as an $(= k)$-path. We begin with the easy observation that the problem can be solved in polynomial (quadratic) time, using a Dijkstra-like search on a "visibility graph."

▶ **Theorem 1.** *Given a polygonal domain P with h convex obstacles and n vertices, a source point s and a destination t, we can compute a shortest k-path from s to t in worst-case time $O((kn + h^2) \log n + kh^2)$.*

Figure 1 Two intersecting 1-paths.

The visibility graph-based approach is inherently quadratic in the worst case, because the number of obstacles can be $h = \Omega(n)$. It also is limited to computing the shortest k-path to only one point (or a fixed set of points) at a time, although it can be extended to support queries in $O(h(k + \log n))$ time apiece after quadratic preprocessing.

The main result of our paper is an algorithm to compute shortest k-paths from s to all points of free space in *sub-quadratic* time $O(k^2 n \log n)$. We do this by computing a *shortest k-path map* of free space; we also prove a tight bound of $\Theta(kn)$ on the combinatorial complexity of SPM_k. Note that the length of a shortest k-path to a point is unique, although some points (along bisectors forming the boundaries of regions in the shortest path map) can be reached by multiple shortest k-paths. For simplicity, however, we assume that the obstacles are in general position, so that the shortest k-path to each obstacle vertex is unique. (Otherwise, if a vertex is reached from s by multiple shortest k-paths, we pick one of them arbitrarily.)

We begin by highlighting a conceptual difficulty with shortest k-paths. The shortest paths to two different destinations can cross each other, which poses an inherent difficulty for the continuous Dijkstra framework of geometric shortest paths [15], since that method depends on the fact that two Euclidean shortest paths from a common source cannot intersect.

▶ **Lemma 2.** *There exist obstacle configurations such that for two destinations t_1, t_2 in free space, the shortest k-paths $\pi_k(t_1)$ and $\pi_k(t_2)$ cross each other, for $k > 0$.*

Proof. The construction, shown in Figure 1, has two identical obstacle bundles A and B placed parallel to the y-axis. Each bundle contains four vertical strips with perforations (single-point openings that split the original strip into disjoint sub-strips). The horizontal spacing between the strips in a bundle is infinitesimal, but for clarity the strips are shown separated in the figure. The points s and t both lie on the x-axis at distance 1 to the left and right of bundles A and B, respectively. We show that there are two shortest 1-paths from s to t, which cross each other, as shown in the figure. We then conclude that by perturbing t up and down slightly we obtain two destination points t_1 and t_2 with their shortest 1-paths crossing, as claimed.

Within each bundle, the openings form an *upper* and a *lower* group. In the upper group, strips 2 and 3 have an opening at $y = (1 + \delta/2)$, and strips 1 and 4 have openings at $y = 1$. In the lower group, all except strip 3 have an opening at $y = -1$. If the distance between the bundles is D, then a shortest 0-path has length $2\sqrt{2} + D + 2\delta$, and a shortest 2-path has length $2\sqrt{2} + D$. A path with exactly one crossing in an upper group has length at least $2\sqrt{2} + D + 3\delta/2$, and a shortest path with one crossing in a lower group has length $2\sqrt{2} + \sqrt{D^2 + 4} + \delta < 2\sqrt{2} + D + 2/D + \delta$. By choosing $D = 10$, say, and $\delta = 4/D$, we can force a shortest 1-path to go through exactly one group of each type. This gives two

intersecting shortest k-paths, $\pi_1(t)$ and $\pi_1'(t)$. Now, let t_1 (resp. t_2) be a destination point obtained by shifting t vertically up (resp. vertically down) infinitesimally. Then it is easy to see that the shortest 1-paths $\pi_1(t_1)$ and $\pi_1(t_2)$ cross each other. ◀

Fortunately, as we show in this section, shortest k-paths can always be decomposed into appropriate non-crossing subpaths to which the continuous Dijkstra method can be applied, working on multiple copies of free space connected using the metaphor of a *k-level garage*. Toward that goal, we establish a series of lemmas.

▶ **Lemma 3.** *A shortest path with exactly k crossings can be decomposed into a shortest path with exactly $(k-1)$ crossings, a straight line segment inside an obstacle, and a shortest path with zero crossings.*

Proof. Let $\pi = (v_1, v_2, \ldots, v_m)$ be an $(= k)$-path from v_1 to v_m. Going backward from v_m along π, let v_i be the first vertex such that the segment $\overline{v_{i-1}v_i}$ intersects one or more obstacles. Let H be the obstacle that is closest to v_i along the segment $\overline{v_{i-1}v_i}$. By the convexity of H, the segment $\overline{v_{i-1}v_i}$ intersects H at two points, which we call p and q, and the segment $\overline{pq}$ lies entirely within H. By subpath optimality, the path from v_1 to p is a shortest path with exactly $k-1$ crossings; by construction, the segment $\overline{pq}$ lies inside the obstacle; and the subpath from q to v_m crosses no obstacles. ◀

▶ **Corollary 4.** *In a shortest k-path, the path segments preceding and following any obstacle crossing are collinear with the path segment inside the obstacle.*

Lemma 3 allows us to break any $\pi_k(t)$ into a $(k-1)$-path $\pi_{k-1}(p)$, a subpath line segment $\overline{pq}$, and an obstacle-free subpath between q and t. We label the last two subpaths with the number of obstacles crossed by the prefix of the path, and call these labels the *prefix counts*. In particular, the prefix count for the subpath $\overline{pq}$ is $k-1$, and the prefix count for the subpath from q to t is k. By a recursive application of Lemma 3, we can decompose $\pi_k(t)$ into $2k+1$ disjoint subpaths whose labels are in non-decreasing order.

The key consequence of this decomposition is the following lemma, which says that subpaths with the same prefix count cannot cross.

▶ **Lemma 5.** *Let $\pi_k(t)$ and $\pi_k'(t')$ be two subpaths whose prefix counts are the same. Then $\pi_k(t)$ and $\pi_k'(t')$ do not cross each other.*

Proof. The proof follows from a simple application of the triangle inequality: if two subpaths with the same prefix count intersect, then we can reconnect the prefix of each path to the suffix of the other, and possibly perform a local shortcut, either shortening at least one path or leaving them the same length but without a crossing. Since the intersecting subpaths are either both inside some obstacle or in free space, avoiding the intersection does not increase the number of obstacle crossings for either path. For instance, in the example in Figure 1, the intersecting edges of the two crossing shortest k-paths have different prefix counts. ◀

The next two lemmas establish properties of shortest k-paths that will be useful later.

▶ **Definition 6.** A point p is *k-visible* from the source s if the segment $\overline{sp}$ passes through at most k obstacles. A *k-visibility edge* is a shortest k-path with exactly one edge.

▶ **Lemma 7.** *If p is not $(k-1)$-visible from s, then the path $\pi_k(p)$ must be an $(= k)$-path.*

Proof. By contradiction. Suppose $\pi_k(p)$ passes through fewer than k obstacles. Since p is not $(k-1)$-visible from s, $\pi_k(p)$ must have at least one bend. The path can then be shortened by going through the obstacle causing this bend, thereby increasing the number of crossings by 1. The resulting path is shorter than $\pi_k(p)$ and has at most k crossings, contradicting the optimality of $\pi_k(p)$. ◀

Let $d_k(p)$ be the length of a shortest k-path to a point p. Clearly, a path that crosses j obstacles and contains at least two segments can be made even shorter if it is allowed to pass through more obstacles. Thus, it follows that for any point p that is not $(k-1)$-visible from s, we must have $d_j(p) > d_{j+1}(p)$, for $j < k$.

▶ **Lemma 8.** *For any point p that is not $(k-1)$-visible from s, the lengths of the shortest j-paths form a decreasing sequence:*

$$d_0(p) \;>\; d_1(p) \;>\; \ldots \;>\; d_i(p) \;>\; \ldots \;>\; d_k(p)$$

3 Shortest Path Map SPM_k: Properties and Bounds

Having established the basic properties of shortest k-paths, we now begin our discussion of the shortest k-path map SPM_k.
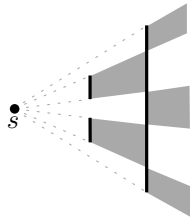
▶ **Definition 9.** Given a shortest k-path $\pi_k(p)$, we define the *k-predecessor* of p to be the vertex of P (including s) that is adjacent to p in $\pi_k(p)$. The partition of free space into connected regions with the same k-predecessor is called the *shortest k-path map*, and denoted SPM_k. The subset of SPM_k for which the shortest path $\pi_k(p)$ to every point p has exactly k crossings is called the shortest *$(=k)$-path map* and denoted by $SPM_{=k}$. See Figure 2 for an example.

Unlike SPM_0, in which the predecessor of a region is always inside or on the boundary of the region, the predecessor of a region in SPM_k may lie outside the region. Moreover, multiple regions in SPM_k may have the same predecessor. (See Figure 2.) Thus, we need to maintain additional information with polygon vertices to disambiguate the predecessor relation. In particular, let v be the *k-predecessor* of p, namely, the vertex adjacent to v in $\pi_k(p)$. Suppose the line segment $\overline{vp}$ crosses $(k-i)$ obstacles, for some $0 \le i \le k$. Then the length $d_k(p)$ of $\pi_k(p)$ is the sum of the length of the i-path to v and the length of segment $\overline{vp}$. We need to maintain the values $d_i(v)$ for all obstacle vertices v and all integers $i = 0, 1, \ldots, k$. In other words,
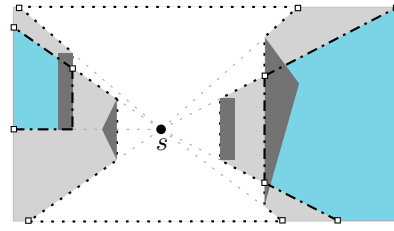
> For a point p in $SPM_{=k}$, we identify the k-predecessor of p by the pair (v, i), where v is a vertex of P and $i \in \{0, 1, \ldots, k\}$, such that $d_k(p) = d_i(v) + |\overline{vp}|$ and the segment $\overline{vp}$ crosses $(k-i)$ obstacles.

Thus, the total number of k-predecessors is $O(kn)$. However, this alone does not bound the number of regions in $SPM_{=k}$ because multiple regions can have the same k-predecessor and the same crossing sequence. Toward our goal of bounding the combinatorial complexity of the map, let us begin with the notion of k-visibility.

We define V_k to be the region consisting of k-visible points, which is star-shaped and therefore simply connected (Figure 3). Now if $\pi_k(p)$ crosses fewer than k obstacles, then by Lemma 7, p must lie in V_{k-1}. The path $\pi_k(p)$ is a straight line segment and the k-predecessor of p is s. Therefore, we have the following.

Figure 2 The 1-predecessor of all points in the shaded region of SPM_1 is $(s, 0)$.

Figure 3 The boundary ∂V_1 of the region V_1 is dash-dotted, and it encloses the boundary ∂V_0, which is shown with dotted segments. The region $V_1 \setminus V_0$ is shown shaded gray.

▶ **Lemma 10.** *All points p such that $\pi_k(p)$ has fewer than k crossings lie in V_{k-1}. Outside of V_{k-1}, SPM_k is the same as $SPM_{=k}$, the shortest path map with exactly k crossings.*

This simplifies our discussion and allows us to decompose SPM_k into two distinct regions, V_{k-1} and $SPM_{=k}$. In the following, we study structural properties of these regions and use them to compute upper bounds on their respective sizes. Later, we combine them to compute an upper bound on the size of the map SPM_k.

3.1 k-Visibility Region

We first bound the complexity of the boundary of V_k, the region visible from s by a segment crossing at most k obstacles.

▶ **Lemma 11.** *The number of edges on the boundary ∂V_k is $O(n + h) = O(n)$.*

Proof. Every vertex of ∂V_k is either a vertex of P or a projection of one of the $2h$ tangents from s to an obstacle of P. The edges on the boundary ∂V_k are therefore sub-segments of the tangents or parts of obstacle boundaries. Each projection vertex belongs to a segment of ∂V_k collinear with s, and the endpoint x farther from s is the end of a maximal segment $\overline{sx}$ that crosses exactly k obstacles. Therefore, each of the $2h$ tangents gives rise to at most one segment of ∂V_k and at most two vertices.   ◀

More interestingly, the bound on the total complexity of these regions is less than the sum of the individual bounds.

▶ **Lemma 12.** *The total number of edges on all ∂V_i, for $0 \le i \le k$, is $O(n + hk)$.*

By connecting s to all vertices on boundary ∂V_{k-1}, we can easily decompose V_{k-1} into constant complexity regions in SPM_k.

3.2   The k-Level Garage and the Structure of $SPM_{=k}$

We now introduce our main idea for computing the shortest k-path map. By Lemma 3, an $(= k)$-path from s to a point p is the concatenation of a $(k - 1)$-path to the boundary of some obstacle H, a shortest path inside H, and a shortest path in free space from the other side of H to p. This suggests an incremental construction of $SPM_{=k}$ from $SPM_{=(k-1)}$. We describe this construction using the metaphor of a k-level *parking garage* with elevators.[1]

---

[1]   The garage metaphor is also used in the context of finding homotopically different paths in [9], but the properties and technical details of our k-garage are quite different.

The idea is to create multiple copies of the input polygonal domain and stack them in levels such that the *shortest paths at each level have the same prefix count and therefore do not intersect.* The planar subdivision of free space at the top level is $SPM_{=k}$.

▶ **Definition 13** (k-garage). We construct the *k-garage structure* by stacking k copies (or floors) of the input polygonal domain P on top of one another, with special connections at the obstacle boundaries. We connect the obstacle H on floor i to its counterpart on floor $i+1$ such that any path that enters H on floor i can exit only on the next higher floor—in a sense, obstacles act as elevators.

Our algorithm to construct $SPM_{=k}$ makes use of the *continuous Dijkstra method*, which simulates the expansion of a unit speed wavefront from the source s in free space. The wavefront at time T contains all points p whose shortest path distance from s is T. The boundary of the wavefront is a set of circular arcs called *wavelets*, each generated by an obstacle vertex (including s) already covered by the wavefront. The generating vertex v is called the *generator* of the wavelet and is identified by the pair (v, w), where w is the time at which v was reached by the wavefront. The generators can be thought of as sources *additively weighted* with delays, since they start emitting wavelets at time w after the start of the simulation. The locus of the meeting points of two adjacent wavelets is a *bisector* curve. Taken together with the obstacle boundaries, bisector curves partition free space into regions of the shortest path map.

We extend the continuous Dijkstra method to our k-garage structure. Each level of the garage is a plane with polygonal obstacles on which wavefronts propagate as usual, but the wavelets can now move to higher floors by entering the obstacles (elevators). More precisely, when the wavefront hits an obstacle H, it is absorbed by the outer boundary of H and is immediately re-emitted into the interior of H. When that wavefront reaches the inner boundary on the other (previously unreached) side of H, it is absorbed and immediately re-emitted on the next higher floor of the garage. This vertical movement therefore adds no delay. In this modified setting, the wavefront at time T contains points on all floors that are at distance T from the source.

The region V_{k-1} is removed from the polygonal domain on floor k of the k-garage because the shortest k-path is known for every point p in V_{k-1}—it is simply the line segment $\overline{sp}$—and leaving these points in the polygonal domain on floor k would create redundant copies of this path. We defer the exact details of our algorithm to Section 4. In the following, we note some properties of the k-garage structure useful to our algorithm.

1. If π is a shortest s–t path from s on floor 0 to t on floor k, then the downward projection $\pi^{\downarrow}$ of π, obtained by projecting π into the planar domain P, is a shortest k-path to t. (To see this, suppose for contradiction we have another k-path π_c from s to t that is shorter. Then by applying Lemma 3 recursively, we can break π_c into $2k+1$ disjoint subpaths ordered by their prefix counts. We now lift the paths into the levels of the garage and concatenate them in order: if the prefix counts of the current and the next subpath are the same, join their common endpoint at the same level as the prefix count; otherwise join their common endpoint at the next level. This transforms the path π_c into a shortest path $\pi_c^{\uparrow}$ from s on floor 0 to t on floor k. Since the vertical movement between the garage floors incurs no delay, the lifted path $\pi_c^{\uparrow}$ is shorter than π, which is a contradiction.)
2. Since wavefront propagation on floor i is affected only by wavelets coming from floors below it, we can think of wavefront propagation on floor i as occurring in a polygonal domain with *multiple* sources. On floor $i > 0$, all sources correspond to generators of wavelets coming from lower floors.

3. To compute the sources at floor $i > 0$, we need to consider only wavelets coming from floor $i - 1$. This follows from Lemma 8, which implies that even if wavelets were allowed to ascend multiple floors in an elevator, a wavelet from floor $i - 1$ would reach floor i no later than the wavelets from other lower floors.

4. The planar subdivision formed by bisectors of colliding wavelets on floor i is the shortest path map for $(= i)$-paths, $SPM_{=i}$. Note that since the obstacles are convex, a shortest path to a point on floor i cannot cross the same obstacle (on any floor) more than once, or else it can be made even shorter.

This suggests a natural way of computing the shortest path map $SPM_{=k}$. We construct maps $SPM_{=i}$ for $i = 0, 1, \ldots, k$ iteratively. Each iteration $i > 0$ is defined by ordinary shortest path propagation with a set of sources that come from the previous iteration. In the following section we use these observations to compute a bound on the size of the shortest k-path map SPM_k.

3.3 Complexity of SPM_k

The shortest k-path map SPM_k on the top floor of the k-garage is precisely $SPM_{=k}$ in the portion of free space that is outside V_{k-1}, as shown in Lemma 7. The boundary of V_{k-1} has linear size, and so we only need to bound the complexity of $SPM_{=k}$. To bound the complexity of $SPM_{=k}$, we consider the embedded planar graph G_k formed by $SPM_{=k}$, V_{k-1}, and the obstacle polygons. We note the following property of planar graphs, which is a direct consequence of Euler's formula.

▶ **Lemma 14.** *Let f be the number of faces in a planar graph $G = (V, E)$. If all the vertices of G have degree three or more, then the size of G is $O(f)$.*

Observe that the "interesting" vertices in G_k are the points where bisectors meet obstacle boundaries or meet each other, and therefore have degree at least three. If f is the number of faces, then by Lemma 14 the complexity of the map due to these vertices is $O(f)$. In addition to this, G_k can also have $O(n)$ vertices of degree two corresponding to the vertices of obstacle polygons, giving a total complexity bound of $O(f + n)$.

Therefore, in order to compute a bound on the complexity of $SPM_{=k}$, it suffices to bound the number of faces f in the graph G_k. We begin with the following well-known result [15].

▶ **Lemma 15.** *The shortest path map of m sources weighted by their delays in a polygonal domain with n vertices and h holes has $f \leq m + n + h \leq m + 2n$ faces. By planarity, the total complexity of the map is $O(f + n)$.*

The key to the proof of the preceding lemma is that each shortest path map region is star-shaped and connected to the predecessor of all points in the region. Since the total number of predecessors is at most $(m + n)$, the number of faces due to these regions is also at most $(m + n)$. Crucially, this lemma does not immediately apply to $SPM_{=k}$, because some predecessors of regions on the kth floor belong to regions *below the kth floor*. That is, some of the m sources are not in the polygonal domain, so the argument that each region is connected to its predecessor does not hold. Fortunately, the argument of Lemma 15 is a topological one, and we can create a topological domain in which the argument applies.

Every point $p \in \partial P$ outside of V_{k-1} is labeled by a $(k-1)$-crossing distance $d_{k-1}(p)$. If p belongs to an obstacle H, and there exists some $q \in \partial H$ such that $d_{k-1}(q) + |\overline{qp}| < d_{k-1}(p)$, then $\pi_k(p)$ may reach p by passing through H. The wavefront that determines $SPM_{=k}$ will be initialized with a weighted source that reaches p by "elevator" passing through H. If

$q \in \partial H$ minimizes $d_{k-1}(q) + |\overline{qp}|$, then the predecessor of q on $\pi_{k-1}(q)$ is the generator of the wavelet that first reaches p in the wavefront. We partition each edge of ∂H into maximal sub-edges with the same predecessor. For each sub-edge with predecessor v, we construct a triangular "flap" by drawing the segments from the sub-edge endpoints to v. Shortest paths propagate from v toward the kth garage floor inside the flap, and in the pseudo-polygonal domain obtained by gluing all the flaps onto the boundary of free space, each shortest path map region is connected to its predecessor. If these flaps were projected into the plane, they would likely overlap, but topologically they do not alter the structure of the domain, and they add only two edges per flap.

▶ **Lemma 16.** *Let P be a polygonal domain with n vertices and h holes. If P is extended by gluing at most m triangular flaps to its boundary, then the shortest path map of m sources weighted by their delays in this extended polygonal domain has $f \leq m + n + h \leq m + 2n$ faces and total complexity $O(m + n)$.*

The preceding lemma applies to the propagation of shortest paths on each floor of the k-garage and also to propagation inside the obstacles (elevators). In both cases the key to bounding the complexity of an iterated construction is bounding the number of sources that propagate into the next level, whether elevator or garage floor. In each elevator and on each garage level $i > 0$, the sources are located on the domain boundary. For simplicity we partition the sources at obstacle vertices, so each source is a maximal (sub-)edge ℓ on some obstacle boundary ∂H, with an associated generator (v, w). We refer to such a source as a *boundary source* and represent it by the triple (v, w, ℓ). Shortest paths from a source (v, w, ℓ) enter the domain through edge ℓ, and their predecessor is vertex v with weight (delay) w. As noted above, each boundary source defines a triangular flap glued onto the boundary of the propagation domain; the flap is the convex hull of ℓ and v.

When boundary sources propagate into some domain (either P or the interior of an obstacle), they define a shortest path map S in the domain. We say that if the region of S corresponding to a source $s = (v, w, \ell)$ intersects a domain edge, then s *claims* the intersection interval on that edge. An *entry claim* of a source (v, w, ℓ) is a claim on edge ℓ itself; entry claims can be ignored for further propagation, since a path that enters the domain through ℓ and exits through the same edge can be shortened. *Exit claims* (ones on edges other than ℓ) define the sources for the next level of shortest path propagation. Within any edge, a maximal sequence of exit claims with the same source is called an *exit claim cluster*. If an exit claim cluster on an edge e has source (v, w, ℓ), then the corresponding boundary source at the next level is (v, w, ℓ'), where ℓ' is the minimal subsegment of e containing the cluster. As noted, entry claims inside ℓ' do not affect shortest path propagation at the next level.
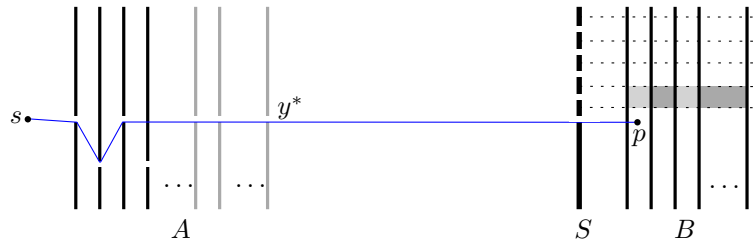
▶ **Lemma 17.** *Let S be the shortest path map obtained by propagating m boundary sources into a polygonal domain with n vertices. Then the number of exit claim clusters of S is at most $m + O(n)$.*

We are now ready to bound the complexity of $SPM_{=k}$.

▶ **Lemma 18.** *The number of faces f_k in $SPM_{=k}$ is $O(n(k+1))$. The complexity of $SPM_{=k}$ has the same asymptotic bound.*

Proof. The proof is by induction. Our goal is to show that there exists a constant C such that the number of faces f_k in $SPM_{=k}$ is at most $Cn(k+1)$ for all $k \geq 0$.

We begin with the inductive step. Let m be the number of exit claim clusters in $SPM_{=(k-1)}$. This is the number of boundary sources in "elevator" propagation across the

Figure 4 A shortest k-path map with complexity $\Omega(nk)$. Bundle A has $2k$ black strips and k gray strips; bundle B has k strips. The thick strip S has $\Omega(n)$ openings. A shortest k-path $\pi(p)$ from s is shown. Observe that $\pi(p)$ crosses $(k-1)$ strips in bundle A and therefore can cross only the first strip in bundle B.

obstacle interiors, going from level $k-1$ to level k. By Lemma 17, the resulting number of exit claim clusters is $m' = m + O(n)$. But m' is the number of boundary sources in the construction of $SPM_{=k}$, and once again by Lemma 17, the resulting number of exit claim clusters is $m'' = m' + O(n) = m + O(n)$, that is, $m'' \leq m + c_1 n$ for some constant c_1.

To establish the base case, recall that a shortest path map with no crossings (SPM_0) has complexity $O(n)$, which implies that the number of exit claims on its boundary is $O(n)$, i.e., at most $c_2 n$ for some constant c_2. Combining the base case and inductive step, we have shown that the number of exit claim clusters on the boundary of $SPM_{=k}$ is at most $c_2 n + k \cdot c_1 n$. The number of faces of $SPM_{=k}$ is at most equal to the number of boundary sources, which is at most $Cn(k+1)$, for $C = \max(c_1, c_2)$. Lemma 14 establishes the total complexity bound. ◀

A Matching Lower Bound. We will now bound the size of SPM_k from below by constructing a map with $\Omega(nk)$ regions. We construct an arrangement of obstacles as shown in Figure 4. We start with two obstacle bundles A and B placed parallel to the y-axis. Within each bundle, the horizontal spaces between strips are infinitesimal, but they are shown enlarged for clarity. The source s lies on the x-axis with bundle A placed right next to it. Bundle A consists of $3k$ perforated strips. In the first $2k$ strips, the odd numbered ones have openings at $y = 0$ and the even numbered ones have openings at $y = -0.5$. The next k strips have an opening at $y = 0$. Bundle B is placed at a distance D to the right of A and consists of k strips with no openings.

The last k strips in bundle A ensure that shortest k-paths starting at s must exit from the opening of the last strip in A (denoted by y^*); a path that crosses the last strip in A at some point other than y^* can be shortened while preserving the same number of crossings. Observe that a shortest path starting at s can reach y^* with i crossings, where $0 \leq i \leq k$. However, each crossing avoided results in an additional length of 1 unit. Therefore a shortest path with i crossings at y^* has an additional length of $(k-i)$ units. Also note that a shortest path with i crossings prior to y^* can cross the first $(k-i)$ of the k strips in bundle B, but cannot cross any farther. Therefore, to the right of strip j in bundle B, we get a region with k-predecessor $(y^*, k-j)$ and a total path length (to a point on the x-axis) of $D + j$. This gives us a total of k regions.

We extend this construction to $\Omega(nk)$ regions by adding a vertical strip S, which acts as a *path splitter*. This special strip has a total of m single-point openings at $y = 0, 1, \ldots, m$, denoted by y_i. We place S at an infinitesimal distance to the left of bundle B, creating k new regions for each opening of S. Note that in the range $0 \leq y \leq m$, a path that crosses S other than at one of the perforations y_i can be shortened by detouring through the nearest y_i and

inserting one more crossing before y^*. Hence a shortest k-path always passes through one of the y_i. This gives a total of $O(mk)$ regions: the k-predecessor of the region at $y = i$ and to the right of strip j of bundle B will be $(y_i, k - j)$, with a total path length of $\sqrt{D^2 + i^2} + j$.

The total number of vertices in our construction is $3k \times 4 + k \times 2 + (m + 1) \times 2 = 14k + 2m + 2$. By choosing $m = (n - 14k - 2)/2$ and assuming $k < n/28$, we have $m = \Theta(n)$ and the total number of regions in SPM_k is $\Omega(nk)$. This gives us the following lemma.

▶ **Lemma 19.** *The worst-case complexity of SPM_k is $\Omega(nk)$.*

Combining Lemmas 11, 18, and 19, we get the main result of this section.

▶ **Theorem 20.** *The shortest k-path map SPM_k has size $\Theta(kn)$.*

4 Computing SPM_k

In this section we describe an $O(k^2 n \log n)$ algorithm to construct SPM_k. Recall from our discussion about the k-garage (Definition 13), we can construct $SPM_{=k}$ iteratively, one level at a time. To compute the map at each level, we propagate the sources from the previous level and then perform wavefront propagation at the current level. For this, we use the algorithm for shortest paths in the presence of polygonal obstacles by Hershberger and Suri [15] as a subroutine. Except for a few small modifications required for our setting, most of the algorithm carries over unchanged. In the following, we briefly review the key ideas and discuss the necessary modifications.

The Hershberger-Suri algorithm uses the continuous Dijkstra method, which simulates the propagation of a unit speed wavefront in free space. The wavefront is a collection of circular wavelets. It changes its shape as it propagates and hits obstacles. Each wavelet originates at a *generator*, which may be a point source or an obstacle vertex (an *intermediate source*). A generator for a wavelet γ is identified by the pair (v, w), where v is an input vertex and w is the time at which v starts emitting γ. The Hershberger-Suri algorithm simulates wavefront propagation over a planar subdivision called the *conforming subdivision* of free space. For each subdivision edge e, and every point $p \in e$, the algorithm identifies the generator whose wavelet first reaches p. Combining these results for all $p \in e$ gives the *wavefront for e*. The key idea of the algorithm is to localize interesting events (such as wavelet collisions) within a constant number of cells in the subdivision. Each free-space edge e of this subdivision is contained in the union of a constant number of cells, called its *well-covering region* $\mathcal{U}(e)$. The wavefront for edge e is computed by combining and propagating the wavefront through $\mathcal{U}(e)$. The computed wavefronts are then merged to compute the shortest path map. This is the main result relevant to our algorithm:

▶ **Lemma 21** ([15]). *Given a set of polygonal obstacles with n vertices and a set of $O(n)$ sources with delays, one can compute the shortest path map in $O(n \log n)$ time and $O(n \log n)$ space.*

From the discussion preceding Lemma 17, recall that the sources on floor i are identified by triples (v, w, ℓ), where ℓ is a (sub-)edge of some obstacle H, (v, w) is a weighted point source on some floor $j < i$, and the wavelet γ generated by (v, w) enters floor i from the interior of H (an elevator) passing through edge ℓ. Each source (v, w, ℓ) defines a triangular flap glued onto the boundary of free space at ℓ. Conceptually, we think of the wavelet γ from (v, w, ℓ) as propagating in the flap before it enters floor i. Algorithmically, we can ignore the flap and start the propagation in free space at edge ℓ. This calls for a slight modification in the initialization step of the Hershberger-Suri algorithm. In particular, we do the following for each edge e of the conforming subdivision:

1. Find all boundary sources (v, w, ℓ) such that the well-covering region $\mathcal{U}(e)$ contains ℓ.
2. Initialize *covertime*(e), which is the time at which e would be engulfed by the wavefront, minimizing over all boundary sources (v, w, ℓ) with $\ell \in \mathcal{U}(e)$, and for each such source considering paths from v with delay w, constrained to pass through ℓ.
3. For each source (v, w, ℓ) with $\ell \in \mathcal{U}(e)$, propagate its wavelet γ to e inside $\mathcal{U}(e)$.

In the following lemma we show how to compute the boundary sources for each step of wavefront propagation.

▶ **Lemma 22.** *Given m boundary sources in a polygonal domain with n vertices, we can compute the exit claims of the sources in $O((m + n) \log(m + n))$ time and space.*

Proof. We apply the Hershberger-Suri algorithm, modified for boundary sources as described above. The algorithm computes the shortest path map for the sources inside the polygonal domain in total time and space $O((m+n) \log(m+n))$. The shortest path map partitions the boundary into $O(m + n)$ intervals, each claimed by its own source. The boundary sources form another set of m intervals. Overlaying these two sets of intervals in additional linear time and space, we identify the exit claims, i.e., those with a claiming source from a different segment. ◀

With these primitives in place, we are ready to describe our algorithm. The input is a polygonal domain P with convex obstacles. We will use M to denote the set of boundary sources passed as input to the Hershberger-Suri algorithm. The algorithm computes two things: the $(k - 1)$-visibility region V and the $(= k)$-path map $SPM_{=k}$, which combined together form SPM_k. The length of the shortest path to any point p can then be easily computed by first locating the region containing p in the map SPM_k and then connecting p to the k-predecessor of this region as described in the beginning of Section 3.

Algorithm to construct SPM_k.
1. Set $M = \{s\}$ and call the Hershberger-Suri algorithm to compute SPM_0 for the polygonal domain P. Initialize V to be the empty region $\emptyset$.
2. Repeat for each $i \in 1, 2, \ldots, k$:
   a. Using Lemma 22, propagate the sources in SPM_{i-1} through the obstacles in P to compute the set of boundary sources M_{new} for $SPM_{=i}$.
   b. Identify all the regions in $SPM_{=(i-1)}$ for which the predecessor is s. Observe that this is precisely the region $V' = V_{i-1} \setminus V_{i-2}$. Set P to be the new polygonal domain with this region removed.
   c. If $V = \emptyset$, then set $V = V'$. Otherwise merge V with V' at the common vertices.
   d. Set $M = M_{new}$ and call the Hershberger-Suri algorithm to compute $SPM_{=i}$ for the polygonal domain P.
3. Merge $SPM_{=k}$ with V at the boundary of regions of $SPM_{=k}$ that have s as predecessor (i.e. $V' = V_k \setminus V_{k-1}$), to obtain SPM_k.

Observe that after Step 2c of iteration i, the region V is equal to V_{i-1}. Because V_{i-1} contains V_{i-2} and because both regions have linear size (by Lemma 11), Step 2c takes linear time. Therefore, the total running time is dominated by k calls to the Hershberger-Suri algorithm with $O(nk)$ sources (Theorem 20). We have the following result.

▶ **Theorem 23.** *If P is a polygonal domain bounded by convex obstacles with a total of n vertices, the shortest k-path map for P with respect to a source point s can be computed in $O(k^2 n \log n)$ time and $(kn \log n)$ space.*

5 Conclusion

In this paper, we studied the problem of finding shortest paths that are allowed to pass through a bounded number of convex obstacles. We showed that although two such k-paths may cross each other, they can be decomposed into non-crossing subpaths based on prefix-counts. This decomposition allows us to compute shortest k-paths efficiently, using the continuous Dijkstra framework. We showed that the size of the shortest k-path map is $\Theta(kn)$ and that it can be computed in worst-case time $O(k^2 n \log n)$ using $(kn \log n)$ space. Our algorithm's time complexity is optimal when $k = O(1)$.

References

1   M. Abellanas, A. García, F. Hurtado, J. Tejel, and J. Urrutia. Augmenting the connectivity of geometric graphs. *Computational Geometry*, 40(3):220–230, 2008.

2   R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice hall, 1993.

3   T. Asano. An efficient algorithm for finding the visibility polygon for a polygonal region with holes. *IEICE TRANSACTIONS (1976-1990)*, 68(9):557–559, 1985.

4   T. Asano, T. Asano, L. Guibas, J. Hershberger, and H. Imai. Visibility of disjoint polygons. *Algorithmica*, 1(1-4):49–63, 1986.

5   J.-L. De Carufel, C. Grimm, A. Maheshwari, and M. Smid. Minimizing the continuous diameter when augmenting paths and cycles with shortcuts. In *15th Scandinavian Symposium and Workshops on Algorithm Theory*, pages 27:1–27:14, 2016.

6   T. M. Chan. Low-dimensional linear programming with violations. *SIAM Journal on Computing*, 34(4):879–893, 2005.

7   D. Z. Chen and H. Wang. Computing shortest paths among curved obstacles in the plane. *ACM Trans. Algorithms*, 11(4):26:1–26:46, 2015.

8   H. Edelsbrunner, L. J. Guibas, and J. Stolfi. Optimal point location in a monotone subdivision. *SIAM Journal on Computing*, 15(2):317–340, 1986.

9   S. Eriksson-Bique, J. Hershberger, V. Polishchuk, B. Speckmann, S. Suri, T. Talvitie, K. Verbeek, and H. Yıldız. Geometric k shortest paths. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1616–1625, 2015.

10   M. Farshi, P. Giannopoulos, and J. Gudmundsson. Improving the stretch factor of a geometric network by edge augmentation. *SIAM Journal on Computing*, 38(1):226–240, 2008.

11   S. K. Ghosh and D. M. Mount. An output-sensitive algorithm for computing visibility graphs. *SIAM Journal on Computing*, 20(5):888–910, 1991.

12   L. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2(1-4):209–233, 1987.

13   S. Har-Peled and V. Koltun. Separability with outliers. *16th International Symposium on Algorithms and Computation*, pages 28–39, 2005.

14   J. Hershberger and J. Snoeyink. Computing minimum length paths of a given homotopy class. *Computational Geometry*, 4(2):63–97, 1994.

15   J. Hershberger and S. Suri. An optimal algorithm for Euclidean shortest paths in the plane. *SIAM Journal on Computing*, 28(6):2215–2256, 1999.

16   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*, pages 359–368, 2013.

17 S. Kapoor and S. N. Maheshwari. Efficient algorithms for Euclidean shortest path and visibility problems with polygonal obstacles. In *Proceedings of the Fourth Annual Symposium on Computational Geometry*, pages 172–182, 1988.

18 D. Kirkpatrick. Optimal search in planar subdivisions. *SIAM Journal on Computing*, 12(1):28–35, 1983.

19 D. T. Lee and F. P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks*, 14(3):393–410, 1984.

20 A. Maheshwari, S. C. Nandy, D. Pattanayak, S. Roy, and M. Smid. Geometric path problems with violations. *Algorithmica*, pages 1–24, 2016.

21 J. Matoušek. On geometric optimization with few violated constraints. *Discrete & Computational Geometry*, 14(4):365–384, 1995.

22 J. S. B. Mitchell. A new algorithm for shortest paths among obstacles in the plane. *Annals of Mathematics and Artificial Intelligence*, 3(1):83–105, 1991.

23 J. S. B. Mitchell. Shortest paths among obstacles in the plane. *International Journal of Computational Geometry & Applications*, 6(3):309–332, 1996.

24 J. S. B. Mitchell and C. H. Papadimitriou. The weighted region problem: finding shortest paths through a weighted planar subdivision. *Journal of the ACM (JACM)*, 38(1):18–73, 1991.

25 M. H. Overmars and E. Welzl. New methods for computing visibility graphs. In *Proceedings of the Fourth Annual Symposium on Computational Geometry*, pages 164–171, 1988.

26 H. Rohnert. Shortest paths in the plane with convex polygonal obstacles. *Information Processing Letters*, 23(2):71–76, 1986.

27 T. Roos and P. Widmayer. *k*-violation linear programming. *Information Processing Letters*, 52(2):109–114, 1994.

28 J. A. Storer and J. H. Reif. Shortest paths in the plane with polygonal obstacles. *Journal of the ACM (JACM)*, 41(5):982–1012, 1994.