Shortest Paths in Polygonal Domains with Polygon-Meet Constraints

Ramtin Khosravi* Mohammad Ghodsi†

Department of Computer Engineering
Sharif University of Technology
P.O. Box: 11365-9415,
Tehran, Iran.
Tel: +98 (21) 600-5616

Abstract

In this paper, we study the problem of finding the shortest path in a polygonal domain in which the path should meet (touch or cross) a simple polygon in the domain. Our method uses the continuous Dijkstra paradigm and reflected wavefronts to solve the problem in worst-case optimal time $O(n \log n)$.

1 Introduction

Finding the shortest path between two points is a basic problem in computational geometry and has many applications in different areas such as motion planning and navigation. The problem is studied over various geometric domains such as simple polygons, polygonal domains, polyhedral surfaces, etc. Also several variations exists according to the metric for computing distances, and different constraints applied to the solution path. Examples of such restrictions are curvature constraints [2] or altitude constraints [1].

We study a special kind of constraints called polygon-meet constraints, in which the shortest path from source to destination should meet a given polygon in a polygonal domain. By meeting a polygon, we mean the path should have non-empty intersection with the closure (interior plus boundary) of a polygon, i.e. either touch the boundary of the polygon and reflect, or cross a part of the polygon.

One possible applications are resource-collection in which an object moving from a source point to a destination point has to collect some resources found in a certain region. Another application is visibility-related constraints in which the polygon to be met is the visibility polygon of a point or an object in the domain. Such a case may arise when direct visibility is needed between the moving object and the viewpoint, such as in communications, or guarding applications.

Our approach here is to use the *continuous Dijkstra* paradigm for finding the shortest polygon- meeting path between two points. As a result of using this paradigm, a subdivision of the domain can be built to answer single-source shortest polygon-meeting path queries. The idea of the method is to propagate a wavefront from the source until it touches the boundary of the target polygon. Upon touching the boundary, the original wavefront is marked as *met* and is propagated ignoring future contacts with the target polygon. Besides the original wavefront, its *reflected* version is also propagated in the opposite direction which is also marked *met*. By *met* wavefront we mean the set of points in the free space to which the length of the shortest polygon-meeting path is δ (Figure 1). The reflected part corresponds to those point whose shortest polygon-meeting path from source has been touched the boundary of the target polygon and has been reflected. If we use the method of Hershberger and Suri [3] for wavefront propagation, we obtain a time-bound of $O(n \log n)$ and $O(n \log n)$ space for our problem which is worst-case optimal. As the size of the constructed map is linear, each query can be answered in $O(\log n)$ time. To the best of our knowledge, no other result has been available on this problem.

*ramtin@mehr.sharif.edu
†ghodsi@sharif.edu

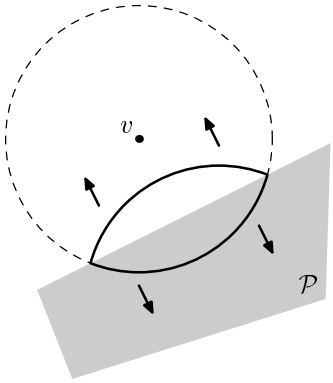

Figure 1: *met* wavefront propagation: the dashed circle is the wavelet generated by vertex v, the gray area belongs to the target polygon $\mathcal{P}$, the solid arcs are *met* wavelets. Arrows show the direction of wavefront expansion.

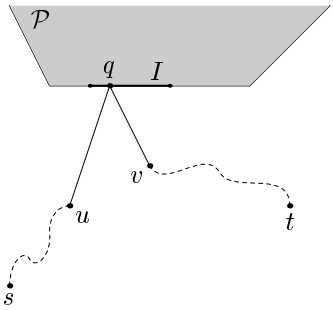

Figure 2: The linearity interval I (shown with heavy solid line) is a connected subset of the boundary of $\mathcal{P}$ such that the shortest path from s (resp. t) to its points passes u (resp. v) as the last vertex. $q \in I$ is the point with minimum total shortest path distance to s and t.

A similar problem is studied in [5] where the constraint is to meet the visibility region of a point. The problem is considered in several domains: simple polygons, polygonal domains, and polyhedral surfaces. The method is based on partitioning the boundary of the target polygon (the polygon to be met) into *linearity intervals* (L-intervals), such that the points of each interval has the property that their shortest paths to both source and destination has the same combinatorial structure. Because of this property, one can find a point $q(I)$ on an L-interval I that has the minimum total distance to source and destination in constant time. So, finding the shortest polygon-meeting path can be done by taking minimum over all L-intervals.

In the case of polygonal domains, the method mentioned yields to the time bound of $O(n \log n)$ which is similar to the bound obtained using the algorithm presented in the current paper. The advantage of the current method is in generating the shortest polygon-meeting path map which cannot be generated using the previous method.

2 Preliminaries

In this section we review some related terminology which is borrowed from [3], since our method relies on the shortest path algorithm given there. Suppose $\mathcal{O} = \{O_1, O_2, \ldots, O_k\}$ be the planar set of polygonal obstacles in the domain with distjoint closures, and s be the source point. Also suppose $\mathcal{P}$ be the target polygon to be met by the path. The total number of vertices in the obstacle polygons as well as the target polygon is assumed to be n. The plane minus the interior of all obstacle polygons is called the *free space*. Given a query point t, our goal is to find the shortest path between s and t which resides completely in the free space, having non-empty intersection with the closure of the target polygon $\mathcal{P}$.

The *shortest path map* of the source point s, denoted by $SPM(s)$, is a linear-size subdivision of the free space into regions (*cells*) such that the shortest path to all the points in one cells has the same

combinatorial structure, i.e., has the same sequence of obstacle vertices along the path. The last obstacle vertex along the path to the points in a cell is called the *root* of that cell. Each cell is star-shaped with respect to its root, which lies on the boundary of the cell. The boundaries of cells consist of portions of obstacle edges, extension segments (extensions of visibility graph edges incident on the root), and bisector curves. The bisector curves are, in general, hyperbolic arcs that are the locus of points that have shortest distance from two roots.

2.1 The Shortest Path Algorithm

Here we review the method of Hershberger and Suri for constructing shortest path map for a point in polygonal domain, which can be done in worst-case optimal time $O(n \log n)$, based on the continuous Dijkstra method. The method simuates the expansion of a wavefront form a point source in the presence of polygonal obstacles. The boundary of the wavefront is a set of cycles, each composed of a sequence of circular arcs. Each arc, called a *wavelet*, is generated by an obstacle vertex already covered by the wavefront; the vertex is called the *generator* of its wavelet. The meeting point between two adjacent wavelets sweeps along a bisector curve, which is either a straight line or a hyperbola. Simulating the wavefront requires processing events that change its topology. These events fall into two categories: wavefront-wavefront collisions and wavefront-obstacle collisions.

To speed up detecting and processing these events quickly, a special subdivision of size $O(n)$ is built on the vertices, temporarily ignoring the line segments between them. Each cell of this subdivision, called a *conforming subdivision*, has a constant number of straight line edges, contains at most one obstacle vertex, and satisfies the following crucial property: for any edge e of the subdivision, there are $O(1)$ cells within distance $2|e|$ of e. Then the obstacle line segments are inserted into the subdivision, but maintaining both the linear size of the subdivision and its conforming property—except now a non-obstacle edge e has the property that there are $O(1)$ cells within shortest path distance $2|e|$ of the edge. These cells form the units of the propagation algorithm: in each step, the wavefront is advanced through one cell. Since each cell has constant descriptive complexity, the propagation in a cell can be done efficiently. When propagating the wavefront across a boundary edge of a cell, instead of keeping track of the exact wavefront, two *approximate wavefronts* approaching the edge from opposite sides are maintained. Using approximate wavefronts, one can detect wavefront–wavefront collisions in a small neighborhood of their actual locations, marking those cells of the subdivision during the propagation phase. At the end of the propagation phase, the edges of the shortest path map are computed exactly for each cell from the collision information stored during propagation.

3 Computing Polygon-Meeting Paths

In this section we present our method for computing the shortest path with polygon-meet constraint. As mentioned earlier, the method is based on the wavefront propagation techniques in computing the shortest path map of the source point s. At the end of the execution, the algorithm generates the shortest polygon-meeting path map of the source point s, which can be used to answer queries in logarithmic time.

3.1 Geometric Properties

Suppose we have a line l, and a source point x in the plane (with no obstacles). Let H be the half-plane generated by l in which x lies. For a point $y \in H$, we define the shortest reflective distance from x to y, as the length of the shortest path that starts from x, meets some points of l, and continues to the point y. It is easy to see that the locus of points y that are of shortest reflective distance δ from x is the part of a circle of radius δ and center $\overline{x}$ that lies in H, where $\overline{x}$ is the reflection of x about l. If we consider a segment e instead of the line l, but with the same requirement for the shortest path, there will be two additional circular arcs generated from the two end-points of e.

Based on the above observations, we use the idea of reflecting a wavelet when colliding the boundary of $\mathcal{P}$, as well as letting it continue inside $\mathcal{P}$. By *original* wavefront we mean the wavefront resulting from the shortest path algorithm, ignoring the target polygon $\mathcal{P}$. Original wavelets are defined similarly. Upon collision of an original wavelet with the boundary of $\mathcal{P}$, a met wavelet of one of these kinds may be generated (Figure 3):

1. A wavelet that is essentially the part of the original wavelet that enters $\mathcal{P}$.

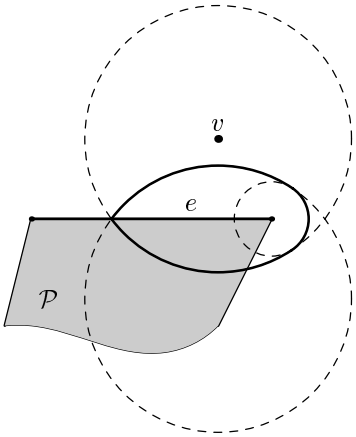

Figure 3: Three kinds of *met* wavelets are shown using solid arcs. The lower arc is part of the original wavefront generated by v, the upper arc is its reflected version, and the right arc (part of the small circle) is generated upon collision of the original wavefront by the right end-point of the edge e of $\mathcal{P}$.

2. A wavelet that is the reflection of the original wavelet about the edge of $\mathcal{P}$ with which the collision is occured.

3. A wavelet of zero radius. This happens when the original wavelet collides a vertex of $\mathcal{P}$.

We call these kind of wavelets (which are generated right after collision of original wavelets with the boundary of $\mathcal{P}$), *primary met wavelets*. In all of the above three forms, the primary wavelet is a circular arc: for the first kind, the center of the arc is the same as that of original wavelet, for the second kind, it is the center of the original wavelet reflected about the boundary edge, and for the third kind, it is the boundary vertex. Right after generation, primary met wavelets continue to propagate according to known rules of wavefront propagation in the Continuous Dijkstra paradigm. New wavelets may be added to the wavefront upon collision of met wavefronts with obstacle vertices, or may be deleted upon collision of met wavefront with each other. We call the newly generated wavelets, *secondary met wavelets*. Note that the collision of met wavelets with original wavelets are not considered during simulation.

Lemma 3.1 *The number of primary met wavelets is $O(n)$.*

Proof. As stated earlier, each primary met wavelet is genereated when an original wavelet first collides the boundary of $\mathcal{P}$. Since each cell of SPM is sweeped with one original wavefront, the portions of boundary that lie in one cell of SPM may be collided by only one original wavefront. In general, if we overlap SPM and the boundary of $\mathcal{P}$, the number of segments created on the boundary is $O(n^2)$, but as we will show, only $O(n)$ of these segments first collide with original wavelets. Others collide with wavelets that have collide these $O(n)$ segments before and therefore considered as met wavelets, not original ones.

To show that the number of segments on the boundary of $\mathcal{P}$ which generate primary met wavelets is linear, consider a cell f of the SPM with root r. If we connect r to vertices of $\mathcal{P}$ lying in f and continue until reaching the boundary of f, we obtain a number of slabs all having the root r in common. Each slab is intersected with a number of edges of $\mathcal{P}$ (possibly zero), but contains no vertex from it (Figure 4). It is easy to see that only one intersecting segment, the closest to r, can generate primary wavelets, since the original wavelet from r reaches all others after passing that segment and therefore considered as a met wavelet, not an original one. So the number of generator segments is bounded by total number of slabs in the SPM. Since each vertex of $\mathcal{P}$ adds at most one slab to the cell containing it, $\mathcal{P}$ has $O(n)$ vertices, and SPM has $O(n)$ cells, this bound is $O(n)$. Since each generator segment may generate at most two primary met wavelets and each vertex of $\mathcal{P}$ generates at most one such wavelet, the total number primary met wavelets is $O(n)$.

□

By *shortest $\mathcal{P}$–meeting path map* ($\mathcal{P}$-SPM for short), we mean the decomposition of the free space of the polygonal domain $\mathcal{P}$ to cells such that the shortest $\mathcal{P}$-meeting path to all points in one cell has the same combinatorial structure. Corresponding to two types of met wavelets (primary and secondary),

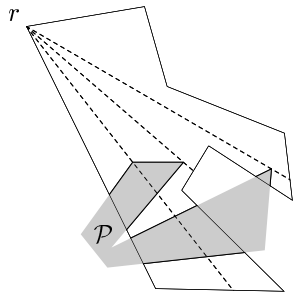

Figure 4: Proof of Lemma 3.1.

there are two kinds of cells in the $\mathcal{P}$-SPM, namely primary and secondary cells. Boundary of all primary cells have non-empty intersection with the boundary of $\mathcal{P}$. This intersection may be a segment, or a single vertex of $\mathcal{P}$.

Lemma 3.2 *The shortest polygon-meeting path map ($\mathcal{P}$-SPM) has linear size.*

Proof. From Lemma 3.1, we can easily conclude that the number of primary cells is linear. Since the secondary cells are generated as the result of propagation of circular arcs, they have the same properties as the cells of SPM, like the property stating that each obstacle vertex is the generator of at most one cell of the SPM. The same property holds for $\mathcal{P}$-SPM and we can conclude that the map has linear size. □

Finally, note that unlike the wavefront in the original shortest path algorithm, the met wavefront in our algorithm may be disconnected, i.e. consists of multiple closed chains of circular wavelets.

3.2 Computing $\mathcal{P}$-SPM

To compute $\mathcal{P}$-SPM, we use two extensions of the algorithm of [3] that the authors have mentioned: non-point sources and multiple sources with specified release times. The algorithm runs in two phases:

1. Computing the primary met wavelets.
2. Propagating the met wavefront.

In order to compute the primary met wavelets, we first use the original wavefront propagation algorithm to find the times at which the original wavefront meets the boundary of $\mathcal{P}$ generating primary met wavelets. To do this, we first consider $\mathcal{P}$ as an obstacle in the domain. This prevents original wavelets to enter the interior of $\mathcal{P}$. At the end of this phase, the following information should be obtained and stored:

1. The first time at which each wavelet collides the boundary $\mathcal{P}$. Note that if more than one wavelet collide a boundary edge, the times of all collisions should be stored.
2. The first time at which a wavelet collides a vertex of $\mathcal{P}$.

Note that considering $\mathcal{P}$ as an obstacle may cause some of the generated time labels be greater than the actual time if we let the wavefront enter $\mathcal{P}$ interior, but this does not make any problem, since in the second phase, when the simulation time reaches a time label, we check whether the generator of the event (either an edge or a vertex of $\mathcal{P}$) is already met. In this case we simpley ignore the event. The correctness of this decision follows from the fact that when we a collision occurs between an original wavelet and $\mathcal{P}$, we let the original wavelet propagete as a met wavelet. Therefore if the wavefront can reach a point of the boundary of $\mathcal{P}$ by entering its interior sooner than the time label, that point is no longer a generator.

In the second phase, we consider the problem as an extension of the standard shortest path algorithm in which there are multiple non-point sources with specified release time. Since the met wavefront may have several independent components, we should consider the propagation in the multiple-source model. As the times the primary met wavelets are generated may be different, these component have different release times. Considering the algorithm based on the non-point extension is due to the fact that the first

two kind of primary wavelets which are generated as a result of collision of original wavelets and $\mathcal{P}$ edges has non-zero radius when generated. This requires special attention for initialization of these wavelets.

In order to incorporate these characteristics into the original algorithm we should initialize the event priority queue of the shortest path algorithm with the information obtained from the first phase such that each primary met wavelet has one entry in the queue with an initial delay associated with it. When processing an event of this kind, the algorithm should compute the initial distance to neighboring conforming subdivision cells (which are constant in number). The rest of the processing is the same as what we have in ordinary shortest path algorithm. Note that the details of multiple-source version of the algorithm should be considered in this part to keep track of the wavefront colors and special case of overlapping well-covering regions of initial primary wavelets.

Using the above schema for the algorithm, we can conclude our main result:

Theorem 3.1 *For a polygonal domain of total complexity n, and a target polygon of size $O(n)$, the shortest polygon-meeting path map of the domain can be built in time $O(n \log n)$ and $O(n \log n)$ space.*

Proof. Since the target polygon has $O(n)$ edges, the first phase of our algorithm can be done in $O(n \log n)$ time and space using standard shortest path algorithm. The event queue can be also initialized in $O(n \log n)$ time. Since the total number of met wavelets is bounded by $O(n)$ and the size of the resulting map is also linear, the time required to construct the map is $O(n \log n)$ using extensions of the standard algorithm in [3]. □

4 Conclusions

We presented an algorithm for finding a special kind of shortest path map, in which the path is constrained to meet a target polygon, which may be a visibility polygon or a resource area in practical applications. The algorithm was based on a variation of the Continuous Dijkstra method for constructing shortest path maps. It is possible to study the problem in several extensions. One exension is to consider different metrics for distance calculations, such as link–distance.

Another extension is to consider the problem in other domains such as simple polygons or polyhedral surfaces. The case of simple polygon is particularly interesting since existence of linear time algorithms for finding shortest paths arises the question of solving the problem in linear time. Extending the method discussed in this paper to the case of polyhedral surfaces is also interesting since the subquadratic algorithm of Kapoor [4] for finding the shortest path in that domain is based on the Continuous Dijkstra paradigm.

If we consider the problem in the polygonal domain with the extension that the path should meet several polygons, it is called *TSP with neighborhoods* problem which is NP-hard and is studied in [6], but if we fix the order of meeting the target polygons, the problem seems to be less complex, yet we do not have any result for it.

References

[1] M. De Berg and M. Van Kreveld. Trekking in the alps without freezing or getting tired. *Algorithmica*, 18:306–323, 1997.

[2] Jean-Daniel Boissonnat, André Cérézo, and Juliette Leblond. Shortest paths of bounded curvature in the plane. *Internat. J. Intell. Syst.*, 10:1–16, 1994.

[3] John Hershberger and Subhash Suri. An optimal algorithm for Euclidean shortest paths in the plane. Manuscript, Washington University, 1995.

[4] S. Kapoor. Efficient computation of geodesic shortest paths. In *Proc. 32th Annu. ACM Sympos. Theory Comput.*, pages 770–779, 1999.

[5] Ramtin Khosravi and Mohammad Ghodsi. Shortest paths with single-point visibility constraints, Submitted for publication.

[6] C. Mata and J. S. Mitchell. Approximation algorithms for geometric tour and network design problems. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 360–369, 1995.