Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information Processing Letters 91 (2004) 171–176
Information Processing Letters
www.elsevier.com/locate/ipl

Shortest paths in simple polygons with polygon-meet constraints ☆

Ramtin Khosravi [a,b,*], Mohammad Ghodsi [b,a]

[a] *IPM School of Computer Science, P.O. Box 19395-5746, Tehran, Iran*
[b] *Department of Computer Engineering, Sharif University of Technology, P.O. Box 11365-9517, Tehran, Iran*

Received 8 December 2003; received in revised form 21 April 2004
Available online 4 June 2004
Communicated by F.Y.L. Chin

Abstract

We study a constrained version of the shortest path problem in simple polygons, in which the path must visit a given target polygon. We provide a worst-case optimal algorithm for this problem and also present a method to construct a subdivision of the simple polygon to efficiently answer queries to retrieve the shortest polygon-meeting paths from a single-source to the query point. The algorithms are linear, both in time and space, in terms of the complexity of the two polygons.

Keywords: Computational geometry; Shortest path; Shortest path map; Polygon-meet

1. Introduction

In a number of applications, like resource-collection or guarding, it is required for a moving object to visit a particular region of the domain during its motion to the destination. Examples include TSP with neighborhoods [4,5], watchman route problem [1,12], zookeeper's problem [2,12], and safari route problem [11,13]. In this work, we study the problem of finding the shortest path between two points inside a simple polygon while the path is constrained to visit (i.e., has non-empty intersection with) a given polygonal region inside the simple polygon (we call it the *target polygon*). We provide a worst-case optimal algorithm for this problem in terms of n which is the total number of vertices in the simple polygon and the target polygon. Furthermore, we show how to construct a subdivision of the domain to answer single-source queries to retrieve the shortest polygon-meeting distances in logarithmic time.

A similar problem is studied in [9] where the domain of the problem is a polygonal domain (polygon with holes). The method is based on wavefront propagation and employs extensions to the shortest path algorithm of Hershberger and Suri [8]. The resulting algorithm works in optimal time $O(n \log n)$ and the same space (n is the total number of vertices in the polygonal domain and the target polygon). The problem we study in this paper is a special case of the mentioned problem, and for the case of convex target polygons

☆ This work has been supported by a grant from IPM School of CS (No. CS1382-2-02).
* Corresponding author.
E-mail addresses: ramtin@mehr.sharif.edu (R. Khosravi), ghodsi@sharif.edu (M. Ghodsi).

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2004.04.010

runs faster. Also in [10], the problem of finding the shortest path between two points lying on the surface of a polyhedron is considered while the path is constrained to meet the visibility region of some viewpoint on the surface. Recently, Dror et al. [3] have presented an algorithm for the problem of finding the shortest path that visits k given convex polygons in a pre-specified order. Also, they have shown that the problem is NP-hard for the case of non-convex polygons.

To find the shortest path constrained to visit the target polygon, we use the property that the constrained path has one of the two forms: either it is the same as the (unconstrained) shortest path that already passes through the interior of the target polygon, or it touches exactly one point on the boundary of the target polygon and bounces back. In the latter case, we compute the relevant portion of the boundary as the set of points that a constrained shortest path may visit for the first time (similar to *first contact set* in [3]). Using this set, we can find the point of contact efficiently. These concepts are covered in Section 2. The algorithm runs in $O(n)$ time and is based on the shortest path algorithm of Guibas et al. [6]. In Section 3, we show how to modify the method of Dror et al. [3] to construct a subdivision of the polygon (named shortest polygon-meeting path map) to answer single-source queries efficiently. This construction takes linear time and is asymptotically faster than the method of [3] applied to the special case of one target polygon.

2. Computing the constrained shortest path

Let $\mathcal{P}$ be the input simple polygon and $\mathcal{T}$ be the boundary of the target polygon lying completely inside $\mathcal{P}$. Assume that the total complexity of $\mathcal{P}$ and $\mathcal{T}$ is n. Our goal is to compute the shortest path between a given source point s and a destination point t in $\mathcal{P}$ such that the intersection of the path and $\mathcal{T}$ is non-empty. We call such a path $\mathcal{T}$*-meeting*. The following lemma states a basic property of the shortest polygon-meeting paths that enables efficient computation of such paths.

Lemma 2.1. *In a simple polygon $\mathcal{P}$, if the shortest path between s and t does not intersect the interior of the target polygon $\mathcal{T}$, neither does the shortest $\mathcal{T}$-meeting path between s and t.*

Proof. Let π^* be the shortest path between s and t, and π be an arbitrary path between s and t that crosses $\mathcal{T}$. It is easy to see that π is the same as π^* up to a vertex of $\mathcal{P}$ at which it deviates from π^*, enters $\mathcal{T}$, and joins π^* once again. The space enclosed between the two paths lies completely inside $\mathcal{P}$. Since π differs π^*, it has one or more *convex bends*. By convex bend we mean a bend that makes an angle less than 180° *inside* $\mathcal{P}$ that of course may be shortcut inside $\mathcal{P}$. We can repeatedly shortcut the convex bends of π, until it has no intersection with $\mathcal{T}$. (Note that this happens at some step, since the path will coincide π^* eventually.) Let $\triangle abc$ be the triangle formed by the last shortcut taken, where the segments ba and ac are replaced by bc. Obviously, either or both ba and ac have non-empty intersection with $\mathcal{T}$ while bc does not. Consider the point p on the part of $\mathcal{T}$ inside $\triangle abc$ which has the shortest perpendicular distance from bc. If we replace ba and ac with bp and pc respectively, we obtain a path which does not enter $\mathcal{T}$ (only touches it at p) and is shorter than π. This shows that if π^* does not intersect $\mathcal{T}$, then any path that goes inside $\mathcal{T}$ can be shortened to another path which does not. □

Assume that the shortest $\mathcal{T}$-meeting path between s and t does not enter the interior of $\mathcal{T}$. If the path visits the boundary in more than one point, then all of the contacts occur at reflex vertices of $\mathcal{P}$ (otherwise, the path can be shortened), and the path is the same as the shortest path. So, the shortest $\mathcal{T}$-meeting path has exactly one point in common with $\mathcal{T}$ (namely q). Obviously, the two subpaths from s to q and from q to t are optimal. So, the entire path is easy to compute as soon as we find q.

To find q, we define the notion of *gate* as follows. Consider the intervals on the boundary of $\mathcal{T}$ when intersected by the shortest path map of a point $p \in \mathcal{P}$. We pick those intervals that are visited for the first time when walking along the shortest paths from p to points inside $\mathcal{P}$ and call them *gates* of $\mathcal{T}$ with respect to p, or p-gates for short. It is easy to prove that the number of gates of $\mathcal{T}$ with respect to any point $p \in \mathcal{P}$ is $O(n)$ [9].

It is also easy to see that the point q defined above, belongs to both an s-gate and a t-gate. We take the intersection of the set of points on s-gates and t-gates and partition the result into a set of intervals $L_{s,t}$, according to the corresponding s-gate and t-gate. For

an interval $I \in L_{s,t}$, computation of the point $q(I) \in I$ which has the minimum total shortest distances to s and t among all points in I takes constant time: I has the property that the last vertices of the shortest paths from s and t to any point on I are the same. Let u (respectively v) be the last vertex on the shortest paths form s (respectively t) to the points of I. To find $q(I)$, we can reflect v about the line supporting I and connect the reflected point to u. If the segment obtained in this way intersects I, the intersection point will be $q(I)$. Otherwise, $q(I)$ will be one of the endpoints of I depending on which side of I lies the intersection point between the segment and the line supporting I. Having the set $L_{s,t}$, we can find the point q by taking minimum over the set $\{q(I) \mid I \in L_{s,t}\}$.

The main step in computing the set $L_{s,t}$ is to find the set of s-gates and t-gates efficiently as the following lemma shows.

Lemma 2.2. *Given a target polygon $\mathcal{T}$ inside a simple polygon $\mathcal{P}$, one can find the gates of $\mathcal{T}$ with respect to a point $p \in \mathcal{P}$ in* $O(n)$ *time.*

Proof. To find the set of p-gates, we choose among a number of intervals obtained by intersecting $\mathcal{T}$ with the edges of SPM(p). To avoid computation of the set of all intervals, we consider the interior of $\mathcal{T}$ as an obstacle and remove it from the free space and denote the result by $\overline{\mathcal{T}} = \mathcal{P} - \mathcal{T}$ which is a simple polygon with exactly one hole. We make a cut in $\overline{\mathcal{T}}$ along a segment from its outer boundary to its inner boundary to obtain a simple polygon which has $\mathcal{T}$ as a part of its boundary. However, care must be taken to choose the cut segment so that it does not block a shortest path from p to a point on a gate of $\mathcal{T}$. To find the appropriate cut, we choose an arbitrary point x on the boundary of $\mathcal{T}$ (Fig. 1). Using the standard shortest path algorithm for simple polygons, we find the last vertex v of $\mathcal{P}$ along the shortest path from p to x. The half line with endpoint v passing through x may have several intersections with the boundary of $\mathcal{T}$. We take the farthest intersection from v, namely y, and cut $\overline{\mathcal{T}}$ along the half line from y away from v until we reach the boundary of $\mathcal{P}$ (at point z). The result will be a simple polygon which we name $\overline{\mathcal{T}}_c$. If we choose an x that is not a vertex of $\mathcal{T}$ and vx that is not co-linear with any edges of $\mathcal{T}$ (which is an easy task), it is obvious that the yz cut is completely outside $\mathcal{T}$.

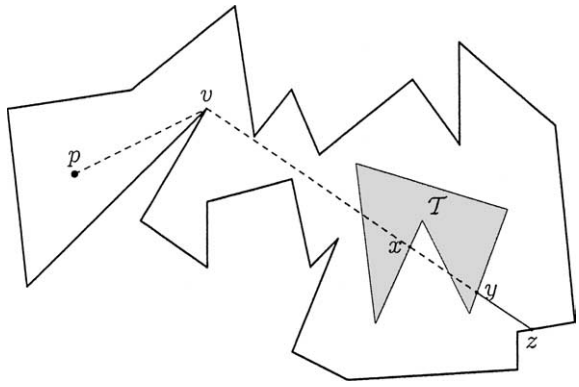

Fig. 1. Proof of Lemma 2.2. We cut $\mathcal{P} - \mathcal{T}$ along yz to obtain the simple polygon $\overline{\mathcal{T}}_c$.

Note that yz is a part of the shortest path from p to z. Due to the fact that no two shortest paths intersect, yz does not intersect any shortest path to a gate.

To find the gates of $\mathcal{T}$, we run the shortest path algorithm of [6] on $\overline{\mathcal{T}}_c$. This will partition $\mathcal{T}$ into a set of intervals which is a superset of the set of p-gates. Those intervals that belong to the cells of SPM(p) with their roots as vertices of $\mathcal{P}$ (not $\mathcal{T}$) are the gates of $\mathcal{T}$ with respect to the source point p. □

The following theorem summarizes the result and provides a time bound on the algorithm.

Theorem 2.1. *Given a pair of points s and t and a target polygon $\mathcal{T}$ inside a simple polygon $\mathcal{P}$, the shortest $\mathcal{T}$-meeting path from s to t inside the polygon can be found in* $O(n)$ *time and* $O(n)$ *space.*

Proof. In linear time, we can check whether either of s or t lies inside $\mathcal{T}$, or the shortest path between them intersects $\mathcal{T}$. In these cases, the shortest Euclidean path between s and t is the answer. If s and t and the shortest path between them lie outside $\mathcal{T}$, we use the shortest path algorithm of [6] to compute the shortest distances to vertices of $\mathcal{P}$ from both s and t. These distances are needed in the final stage of the algorithm to compute the shortest $\mathcal{T}$-meeting distances. Also, we compute the set of s-gates and t-gates using the technique mentioned in Lemma 2.2. All these steps can be done in $O(n)$ time and space.

To merge the two sets of gates efficiently, we must guarantee that the intervals in each set is in sorted order on $\mathcal{T}$. This is possible if we make the recursive

calls made during the DFS traversal in the shortest path algorithm sorted in some fixed direction (e.g., clockwise). Since the funnel structure used in the algorithm is stored in a *finger search tree* [7], the list of s-gates and t-gates obtained can be arranged in a sorted list in linear time. Thus, we can merge the end points of the two sets of gates in $O(n)$, obtaining the set $L_{s,t}$. The rest of algorithm is straightforward and can be done in linear time. □

3. Computing the shortest polygon-meeting path map

We now consider the problem of constructing a subdivision of the input polygon $\mathcal{P}$ such that for an arbitrary query point $x \in \mathcal{P}$, the shortest $\mathcal{T}$-meeting path from s to x can be reported efficiently. We call such a subdivision the shortest $\mathcal{T}$-meeting path map of $\mathcal{P}$ with respect to s ($\mathcal{T}$-SPM for short). We study the problem in two cases depending on $\mathcal{T}$ being convex or non-convex.

3.1. Convex target polygon

In this case, the problem can be considered as a special case of the *general touring polygons problem* (general TPP) studied by Dror et al. [3]. In that problem, the goal is to find the shortest path between two given points that visits k convex polygons in a given order. In its general form, the input to the problem contains a number of polygonal regions called *fences* and the path is constrained such that the subpath between any two consecutive polygons should lie inside the corresponding fence. The running time of the proposed algorithm is $O(nk^2 \log n)$. For $k = 1$ with $\mathcal{T}$ as the only polygon to be visited and $\mathcal{P}$ as the fence for all parts of the path, we can obtain $\mathcal{T}$-SPM in $O(n \log n)$ time. However, we show this can be reduced to $O(n)$. Since our method is closely related to the mentioned work, we borrow the related terminology from [3] and review some terms adapted to the notation used in this paper.

Consider the source point s and the target polygon $\mathcal{T}$. Let G_s be the set of points on s-gates of $\mathcal{T}$. It can be easily verified that this set is a polygonal chain (since $\mathcal{T}$ is convex). We define R as the set of rays leaving points of G_s along shortest $\mathcal{T}$-meeting paths to points in $\mathcal{P}$. R is a *starburst* with *source* G_s which means that if we ignore $\mathcal{P}$, each point in the plane is reached by exactly one ray (the proof is given in [3]). Associated with R is a subdivision $\mathcal{S}^R$ of the plane that groups together points reached by rays of R that leave from the same vertex of G_s, or leave from the same side of an s-gate (Fig. 2). $\mathcal{S}^R$ imposes a decomposition on the interior of $\mathcal{P}$ into a number of cells, such that the shortest $\mathcal{T}$-meeting path to all points in a cell leaves $\mathcal{T}$ from the same vertex of G_s or the same side of an s-gate. We define the *root* of a cell of this decomposition in the following way. If the cell corresponds to a vertex of G_s, that vertex is the root of the cell. If the cell corresponds to an s-gate g, we have two cases depending on whether the paths to the points of the cell *pass through* g or are *reflected* by g. In the first case, the root of the cell is the last vertex of the shortest paths from s to points on g and in the second case, the root is the mentioned vertex reflected about g. In Fig. 2, the root of the cell containing s is $\bar{v}$ which is v reflected about the gate bc.

To compute $\mathcal{T}$-SPM, we consider the cells of the mentioned decomposition separately, and for each cell, construct its SPM with respect to the root of the cell. Note that for the cells with roots other than vertices of G_s, we have to add the triangle to the cell which is obtained by connecting the root of the cell to the endpoints of the corresponding gate. In Fig. 2, to compute the map for the cell containing s, we add $\triangle \bar{v}bc$ to the cell. After computing the map

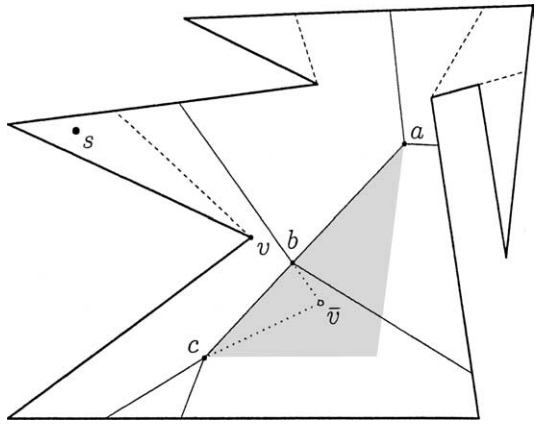

Fig. 2. The structure of the shortest $\mathcal{T}$-meeting path map: the solid thin edges (from $\mathcal{S}^R$) together with the dashed edges make the set of edges of $\mathcal{T}$-SPM.

using the standard shortest path algorithm, we discard the added triangle. Now, $\mathcal{T}$-SPM is obtained by taking the edges of the maps together with the edges of the decomposition. The following theorem states that computing this subdivision takes linear time.

Theorem 3.1. *Given a source point s and a convex target polygon $\mathcal{T}$ inside a simple polygon $\mathcal{P}$, the shortest $\mathcal{T}$-meeting path map with respect to s can be computed in* $O(n)$ *time and* $O(n)$ *space.*

Proof. To compute the map, we first compute the set of s-gates on the boundary of $\mathcal{T}$. According to Lemma 2.2, this takes $O(n)$ time. The method used guarantees that the set of gates are obtained in sorted order. Thus, we can compute the edges of $\mathcal{S}^R$ in sorted order in linear time. Assuming that we are given the edges of $\mathcal{P}$ in sorted order, we can find the decomposition of $\mathcal{P}$ imposed by $\mathcal{S}^R$ in $O(n)$ time. We start by picking up the first ray and find its first intersection with the edges of $\mathcal{P}$ and traverse both sequences of rays and edges in the same direction. (The notion of direction must be defined precisely, but we omit the details due to space limitations.) During the traversal, we consider two consecutive rays such as r_i and r_{i+1} assuming we have found the intersection of r_i with the edge e_j, which is the first edge in the sequence of edges that intersects the ray. (Note that this intersection is not necessarily a vertex of decomposition, since the ray may be blocked by an edge that we visit later in our traversal.) We pick the edges $e_{j+i}, e_{j+2}, \ldots$ one by one, and keep track of the last edge that intersects r_i, until we find an edge e_k that intersects r_{i+1}. At this time, the last intersection of r_i is identified as a vertex of the decomposition and we continue the process on the rays r_{i+1} and r_{i+2}.

The computed decomposition has $O(n)$ faces with the total complexity of $O(n)$. Computing the SPM for all faces can be done in $O(n)$ total time and the same space. Finally, these maps are merged into a single subdivision in linear time. □

3.2. Non-convex target polygon

In this case, the set of rays R defined above is no longer a starburst, which means there may be points in $\mathcal{P}$ that can be reached by more than one *locally* shortest $\mathcal{T}$-meeting paths which leave $\mathcal{T}$ differently. This introduces new kind of edges which are bisector curves dividing portions of $\mathcal{P}$ claimed by two or more parts of G_s. Also, having the set of s-gates in sorted order along the boundary of $\mathcal{T}$, does not imply that the intersections of $\mathcal{S}^R$ with $\mathcal{P}$ occur in the same order. This requires an additional sorting process which needs $O(n \log n)$ itself. In this case, we can use the algorithm presented in [9] to compute $\mathcal{T}$-SPM in polygonal domains in $O(n \log n)$ time.

4. Conclusion

We presented an algorithm to find the shortest path between two points in a simple polygon constrained to meet a polygonal region inside the simple polygon. The algorithm runs in linear time and space in terms of the complexity of the two polygons. Also, we gave a method to compute a subdivision of the simple polygon in linear time to answer single source queries in logarithmic time.

Our work was mainly an application of the shortest path algorithm in simple polygons to a constrained version of the problem. The problem is not much harder than the unconstrained version as the algorithms are worst-case optimal. A modified version of the problem is to constrain the path to meet several target polygons in a fixed order. As Dror et al. [3] have shown, this problem is NP-hard for non-convex target polygons. For convex targets in a simple polygon, one can immediately apply the algorithm of Dror et al. considering $\mathcal{P}$ as the fence for all parts of the path.

Acknowledgement

We would like to thank the anonymous referee for his/her valuable remarks.

References

[1] W.-P. Chin, S. Ntafos, Watchman routes in simple polygons, Discrete Comput. Geom. 6 (1) (1991) 9–31.
[2] W.-P. Chin, S. Ntafos, The zookeeper route problem, Inform. Sci. 63 (3) (1992) 245–259.
[3] M. Dror, A. Efrat, A. Lubiw, J.S.B. Mitchell, Touring a sequence of polygons, in: Proc. 35th ACM Symp. Theory Comput., 2003.
[4] A. Dumitrescu, J.S.B. Mitchell, Approximation algorithms for TSP with neighborhoods in the plane, in: Proc. Symposium on Discrete Algorithms, 2001, pp. 38–46.

[5] J. Gudmundsson, C. Levcopoulos, A fast approximation algorithm for TSP with neighborhoods and red–blue separation, in: Lecture Notes in Comput. Sci., vol. 1627, Springer, Berlin, 1999, pp. 473–482.
[6] L.J. Guibas, J. Hershberger, D. Leven, M. Sharir, R.E. Tarjan, Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons, Algorithmica 2 (1987) 209–233.
[7] L.J. Guibas, E. McCreight, M. Plass, J. Roberts, A new representation for linear lists, in: Proc. 9th Annual ACM Symp. Theory Comput., 1977, pp. 49–60.
[8] J. Hershberger, S. Suri, An optimal algorithm for Euclidean shortest paths in the plane, SIAM J. Comput. 28 (6) (1999) 2215–2256.
[9] R. Khosravi, M. Ghodsi, Shortest paths in polygonal domains with polygon-meet constraints, in: Proc. 19th European Workshop Comput. Geom., 2003, pp. 137–142.
[10] R. Khosravi, M. Ghodsi, M. Taghdiri, Shortest point-visible paths on polyhedral surfaces, in: Proc. of the 10th International Conference on Computing and Information (ICCI 2000), 2000.
[11] S. Ntafos, Watchman routes under limited visibility, Comput. Geom. Theory Appl. 1 (3) (1992) 149–170.
[12] X. Tan, Approximation algorithms for the watchman route and zookeeper's problems, Discrete Appl. Math. 136 (2–3) (2004) 363–376.
[13] X. Tan, T. Hirata, Finding shortest safari routes in simple polygons, Inform. Process. Lett. 87 (4) (2003) 179–186.