ELSEVIER

Computational Geometry 16 (2000) 223–233

Computational Geometry
Theory and Applications

www. elsevier.nl/locate/comgeo

Computing simple paths among obstacles

Qi Cheng[a,1], Marek Chrobak[a,*,1], Gopalakrishnan Sundaram[b]

[a] *Department of Computer Science, University of California, Riverside, CA 92521, USA*
[b] *Environmental Systems Research Institute Inc., Redlands, CA 92373, USA*

Communicated by K. Mehlhorn; submitted 1 January 1997; revised 1 January 1999; accepted 1 May 2000

Abstract

Given a set X of points in the plane, two distinguished points $s, t \in X$, and a set Φ of obstacles represented by line segments, we wish to compute a simple polygonal path from s to t that uses only points in X as vertices and avoids the obstacles in Φ. We present two results: (1) we show that finding such simple paths among arbitrary obstacles is NP-complete, and (2) we give a polynomial-time algorithm that computes simple paths when the obstacles form a simple polygon P and X is inside P. Our algorithm runs in time $O(m^2n^2)$, where m is the number of vertices of P and n is the number of points in X. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Computational geometry; Polygon; Simple path; NP-completeness

1. Introduction

The research we describe in this paper was motivated by polygon generation problems. Suppose that given a set X of points in the plane, we wish to generate all simple polygons whose vertices are in X. A simple, iterative approach to this problem is to start at an arbitrary point $x \in X$ and successively extend the path. Given the current path L ending at a point y, let Y denote the set of points of X that are not on L. We can extend L to all points $z \in Y \cup \{x\}$ for which the interior of the line segment $[y, z]$ does not intersect L. For some sequences of choices, however, we can reach situations when y cannot be connected to any $z \in Y \cup \{x\}$ without intersecting L. Thus such a path L is not a feasible partial solution. In order to avoid this difficulty, and reduce the backtracking, at any step of this process, given the current (feasible) path L, we would like to verify which extensions of L are feasible, that is, to compute the set of all z's in Y for which the extended path $L' = Lz$ can be completed to a simple polygon. One can think of this question as the problem of finding a simple path that avoids given obstacles and uses given points as

* Corresponding author.

E-mail addresses: qcheng@cs.ucr.edu (Q. Cheng), marek@cs.ucr.edu (M. Chrobak), gsundaram@esri.com (G. Sundaram).

[1] This research supported by NSF Grant CCR-9503498.

vertices: we want to find a simple path from z to x that does not intersect $L' - \{x, z\}$ and whose vertices are in $Y \cup \{x\}$.

A similar idea can be applied to the problem of generating a "random" polygon with vertices in X. In this case the next point z can be picked, say, uniformly at random from $Y \cup \{x\}$. Note, however, that the probability distribution on the polygons resulting from this process is not necessarily uniform, that is, different polygons may be generated with different probabilities. It is an open problem whether there is a polynomial-time algorithm that uniformly generates simple polygons with vertices from a given set X (see, for example, [2,15]).

Let us now define our problem formally. Given points $x_0, x_1, \ldots, x_k$, by a *path* $L = x_0 x_1 \ldots x_k$ we mean the union of line segments $[x_i, x_{i+1}]$, for $i = 0, \ldots, k-1$. L is called *simple* if it is homeomorphic to a line segment (in other words, L does not "intersect itself"). We refer to the points x_i as the *vertices* of L. Points x_0 and x_k are called the *start point* and the *end point* of L, respectively, and are usually denoted by s and t. When X is a finite set of points and $x_0, \ldots, x_k \in X$ then L is called an (s, X, t)*-path*. If Φ is any subset of the plane, then we say that L *avoids* Φ if $L \cap \Phi \subseteq \{s, t\}$. In other words, L does not intersect Φ, except for possibly at points s and t.

Our task can be stated as follows. We are given a finite set of points X, two distinguished points $s, t \in X$, and an obstacle set Φ represented by a union of line segments, and we wish to compute a simple (s, X, t)-path that avoids Φ, if it exists, or to report that such path does not exist, otherwise.

We present two results. In Section 2 we prove that it is NP-complete to decide whether there is a simple (s, X, t)-path that avoids Φ. In Section 3 we consider the case when the obstacle set is a simple polygon P and X is inside P. We give a polynomial time dynamic-programming algorithm for computing simple (s, X, t)-paths in P. Our algorithm works in time $O(m^2 n^2)$, where n is the number of points in X and m is the number of vertices of P.

To the best of our knowledge, no research on the above problem has been reported in the literature. A related problem of computing arbitrary (s, X, t)-paths avoiding Φ, not necessarily simple, can be solved using visibility graphs. Let $\mathcal{G}_\Phi(X)$ be the visibility graph of X with respect to Φ defined as

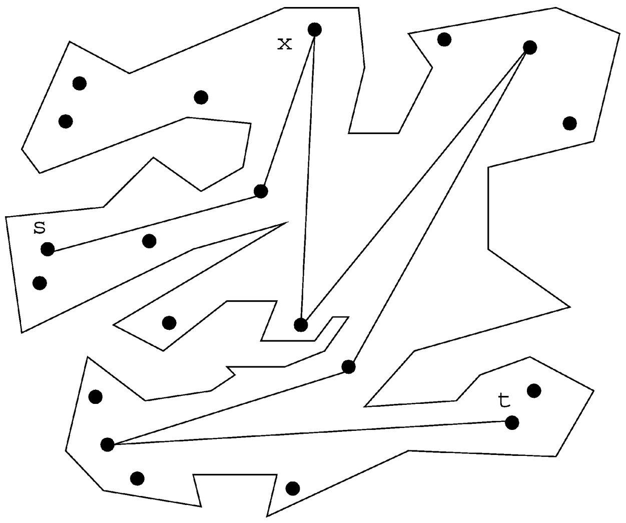

Fig. 1. Example of a simple (s, X, t)-path.

follows. The vertices of $\mathcal{G}_P(X)$ are the points in X, and two points $x, y \in X$ are connected by an edge in $\mathcal{G}_\Phi(X)$ if the line segment $[x, y]$ does not intersect Φ. (More precisely, $[x, y] \cap \Phi \subseteq \{s, t\}$.) Then, finding an (s, X, t)-path that avoids Φ reduces to the problem of finding an st-path in $\mathcal{G}_\Phi(X)$. The latter problem can be solved in time $O(n + e)$, where $n = |X|$ and $e = O(n^2)$ is the number of edges in $\mathcal{G}_\Phi(X)$. The graph $\mathcal{G}_\Phi(X)$ can be computed using the algorithm by Ghosh and Mount [7] that works in time $O(e + (n + m)\log(n + m))$, where m is the number of line segments in Φ.

Requiring that the (s, X, t)-path is simple makes the problem substantially harder. Fig. 1 shows an example of a simple (s, X, t)-path inside a polygon. Note that if we remove point x, no simple (s, X, t)-path will exist, but there still exists a self-intersecting (s, X, t)-path. We leave it as an exercise for the reader to construct a similar example in which X has only four points.

2. Arbitrary obstacles

In this section we show that the simple path problem is NP-complete for arbitrary obstacles. The input consists of an obstacle set Φ represented by a union of line segments, a set X of points, and two points $s, t \in X$. Our goal is to determine if there is a simple (s, X, t)-path that avoids Φ. It is easy to see that this problem is in NP. The proof of NP-hardness is by reduction from the problem HP3PD, the Hamiltonian path problem for 3-regular planar digraphs, which is known to be NP-complete, see [12].

First we draw the given 3-regular planar digraph $G = (V, E)$ on the rectilinear $O(n^2)$ size grid, with at least one bend in each edge (see Fig. 2). This can be achieved in polynomial time by slightly modifying the algorithms from [6,14]. Next, magnify the drawing and replace each vertex v with a corresponding gadget A_v. For each arc, replace one bend w on this arc with a gadget B_w and all other bends z with gadgets C_z. Each gadget consist of a number of obstacles and points.

Let $|V| = n$. We think of A_v as a box that has three groups of "entrance points" corresponding to the edges incident to v. For example, if there is an arc from v to u (or from u to v), A_v has a group of entrance points $e^v_{u,i}$, $1 \leqslant i \leqslant n$. Each A_v is drawn in the plane so that the points $e^v_{u,i}$, for $i = 1, \ldots, n$, are ordered clockwise around the center of A_v. Gadget A_v is shown in Fig. 3, in which solid dots are points of X, solid lines represent the obstacles, and dashed lines represent the visibility lines between points in X. Any path with the vertices in X must follow these visibility lines.

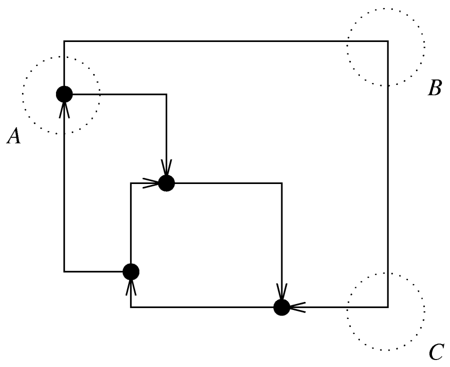

Fig. 2. A rectilinear drawing of a 3-regular planar digraph.

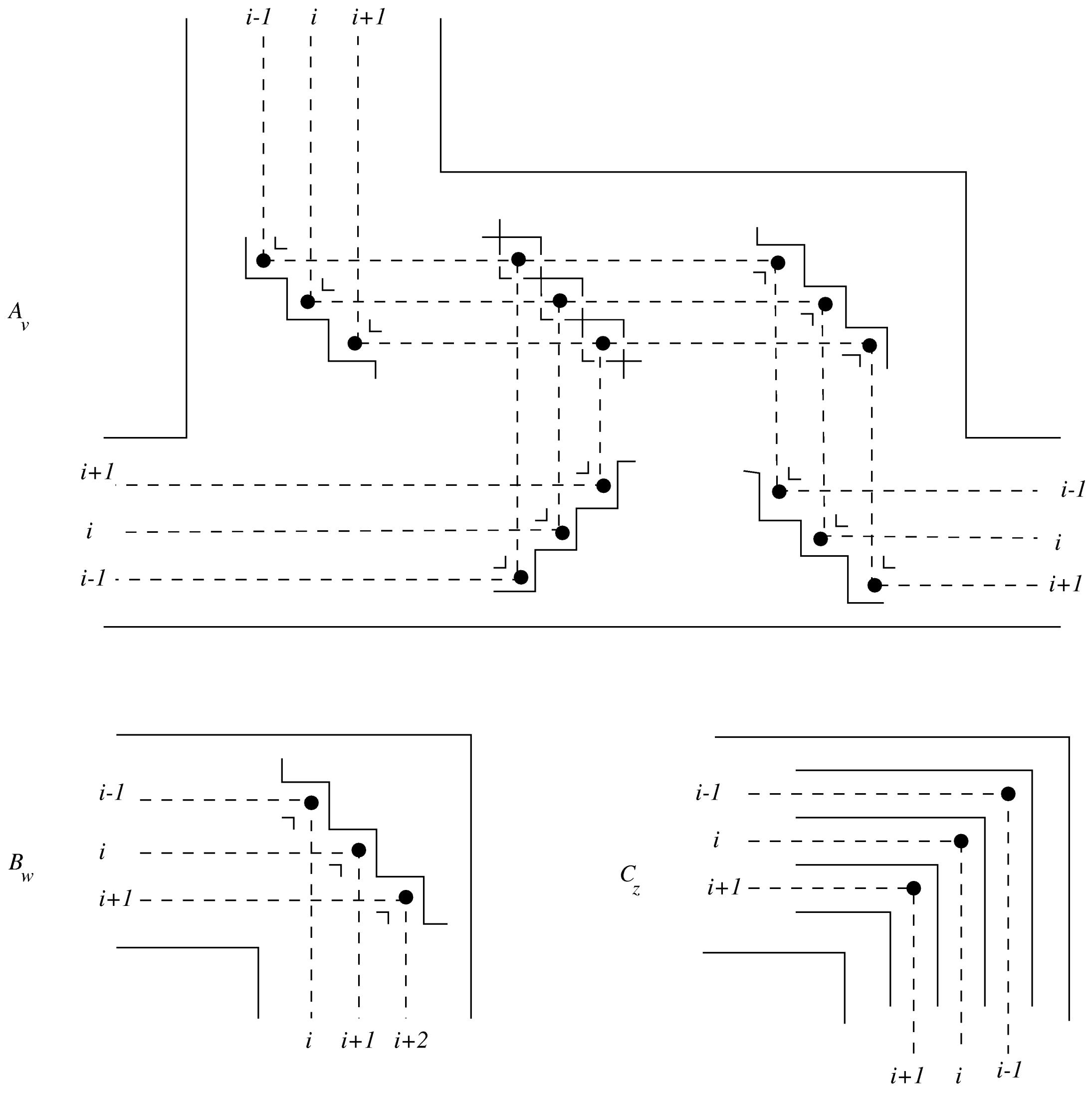

Fig. 3. Gadgets A_v, B_w and C_z.

We also use appropriate gadgets B_w and C_z, so that the following condition holds: if there is an arc (v,u) in G, an entrance $e^v_{u,i}$ in A_v can only see (indirectly) the entrance $e^u_{v,i+1}$ in A_u. The indices $\dots i-1, i, i+1, \dots$ on the visibility lines in Fig. 3 are the indices of the corresponding entrance points.

Gadget C_z simply bends the visibility lines at the right angle. Gadget B_w is similar, but in addition to bending the direction of the visibility lines, it also reverses and shifts the order of their indices. It increases the indices by one when traversing along the direction of the arc, with the exception of index n for which we reach a dead-end. A symmetric statement holds for traversing an arc in the opposite direction.

Let the neighbors of v be u_1, u_2, u_3. The crucial property of gadget A_v is that if A_v is entered through entrance $e^v_{u_a,i}$ then the only way to leave it, without self-intersecting the path, is through an entrance $e^v_{u_b,i}$ for some $b \neq a$. Additionally, once we visit A_v this way, it is impossible to visit it again without crossing edges. In this case we say that A_v is visited in *mode i*.

Gadget A_s looks similar to A_v in Fig. 3. To construct A_s, remove the $2n - 2$ interior (non-entrance) points whose visibility lines have indices greater than 1, and name any of the remaining two interior vertices as s_0. We construct gadget A_t, with an interior vertex t_0, analogously.

The above construction can be performed in polynomial time and, in addition, the resulting grid has polynomial size. Let X be the set of points and Φ the set of obstacles created above. It remains to show that G has a Hamiltonian path from s to t if and only if there is a simple (s_0, X, t_0)-path that avoids the obstacles in Φ.

If we have a Hamiltonian path H in G from s to t, then we can go from s_0 to t_0 by visiting the gadgets A_v in the same order as we visit the vertices v of G in H. Let v_i denote the ith vertex on H, for $i = 2, \ldots, n-1$. We traverse A_{v_i} in mode i, that is we go from $e^{v_i}_{v_{i-1},i}$ to $e^{v_i}_{v_{i+1},i}$. Between gadgets A_{v_i} and $A_{v_{i+1}}$, for $i = 1, \ldots, n-1$, we go from $e^{v_i}_{v_{i+1},i}$ to $e^{v_{i+1}}_{v_i,i+1}$. That the resulting path is simple follows from the properties of the gadgets A_v, B_w and C_z.

Suppose now that there is a simple path L from s_0 to t_0 that avoids Φ. Each gadget can be visited at most once. We leave A_s in mode 1, we enter A_t in mode n, and the mode between visiting two gadgets can only increase or decrease by 1. Since we can visit each gadget A_v at most once, the only way it can happen is when we visit all gadgets A_v and increase the mode at every step. Recall that increasing the mode corresponds to following the direction of an arc. Then the permutation of the vertices corresponding to the order in which we visit the gadgets determines a Hamiltonian path in G from s to t.

We summarize our result in the following theorem.

Theorem 1. *The problem of finding a simple (s, X, t)-path that avoids a given set Φ of obstacles is strongly NP-complete.*

3. Paths in simple polygons

In this section we concentrate on computing simple (s, X, t)-paths inside a simple polygon P. We view P as a closed bounded region, whose boundary, ∂P, constitutes the set Φ of obstacle line segments. We let V_P denote the set of m vertices of P. We can assume that $X \subseteq P$ and $X \cap \partial P \subseteq \{s, t\}$.

For the sake of simplicity, throughout the rest of the paper we assume that all the points in $V_P \cup X$ are in a general position, that is, no three points in $V_P \cup X$ are collinear. Otherwise, we can appropriately perturb some points without changing the solution.

By (x, y) we denote the open line segment between x and y, $(x, y) = [x, y] - \{x, y\}$. Similarly, $[x, y) = [x, y] - \{y\}$ and $(x, y] = [x, y] - \{x\}$. If $[x, y] \subseteq P$, we say that x, y *see each other* in P, or that y is *visible* from x in P.

For arbitrary points x, y on a path L (not necessarily vertices), by $L[x, y]$ we denote the sub-path of L between x and y. If x, y are non-consecutive vertices of L that see each other in P, then $[x, y]$ is called a *shortcut* of L. Path L is called *shortcut-free* if it has no shortcuts. If L is simple and $[x, y]$ is a shortcut whose interior does not intersect L, then $[x, y]$ is called a *simple* shortcut. In other words, “simple” means here that we can take the shortcut without violating the simplicity of L.

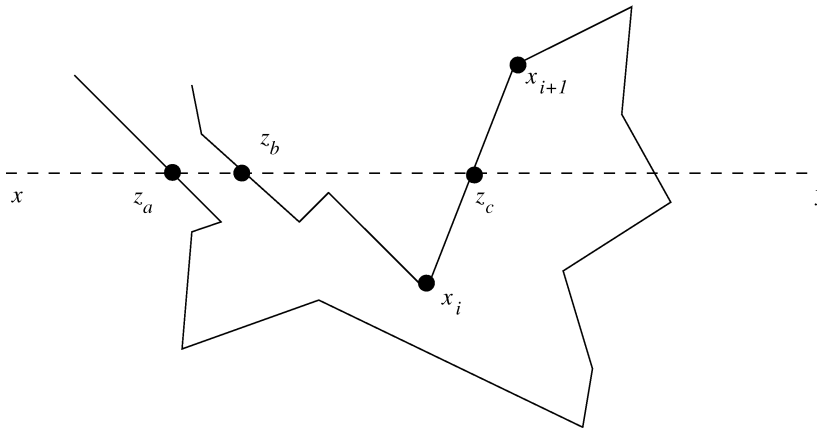

Fig. 4. The construction of z_a, z_b, z_c in the proof of Lemma 1.

Lemma 1 (The Consecutive Intersection Property). *Let $L = x_0x_1 \dots x_k$ be a simple path in P without simple shortcuts, and $x, y \in P$ be two points that see each other in P. Let $z_1, \dots, z_j$ be the intersection points of L with $[x, y]$, listed in the order in which they appear on L. Then $z_1, \dots, z_j$ are consecutive on $[x, y]$, that is, z_i is between z_{i-1} and z_{i+1} for all $i = 2, \dots, j-1$.*

Proof. Consider a permutation π of $1, 2, \dots, j$, such that, going from x to y, the intersection points are ordered $z_{\pi(1)}, z_{\pi(2)}, \dots, z_{\pi(j)}$. If the lemma is false, then $j \geqslant 3$ and there is a b, $1 \leqslant b \leqslant j$, that is a local extremum of π, and such that z_b is neither closest to x nor to y. By symmetry, assume b is a local minimum. This means that if z_a and z_c are the intersection points immediately to the left and to the right of z_b, then z_b appears before z_a and z_c on L. Again, by symmetry, we can assume that z_c appears before z_a on L.

Let $z_c \in [x_i, x_{i+1})$, and consider a simple polygon Q whose boundary is $\partial Q = L[z_b, z_a] \cup [z_a, z_b]$. Suppose first that $x_0 \notin Q$. That means that L crosses $[x, y]$ at z_b and z_a in the opposite direction (see Fig. 4). In Q, either x_i and x_{i+2} can see each other, or x_{i+1} can see some vertex u of Q. By the construction, u cannot be z_a nor z_b. Thus, in both cases, we obtain a simple shortcut of L – a contradiction.

If $x_0 \in Q$, then $L[x_0, z_b]$ is inside Q. In this case, essentially the same argument as before shows that either x_i sees x_{i+2} in Q, or x_{i+1} sees a vertex of Q other than z_a or z_b. □

Obviously, for the purpose of computing simple (s, X, t)-paths we can restrict our attention to simple paths without simple shortcuts. The lemma below states that we can in fact consider only shortcut-free paths.

Lemma 2. *If there exists a simple (s, X, t)-path in P, then there also exists such a path that is shortcut-free.*

Proof. Let $L = x_0x_1 \dots x_k$ be a simple (s, X, t)-path in P. Without loss of generality, we can assume that L has no simple shortcuts. Suppose that it has a shortcut $[x_i, x_j]$, for $i < j$. By Lemma 1, $L[s, x_i]$ and

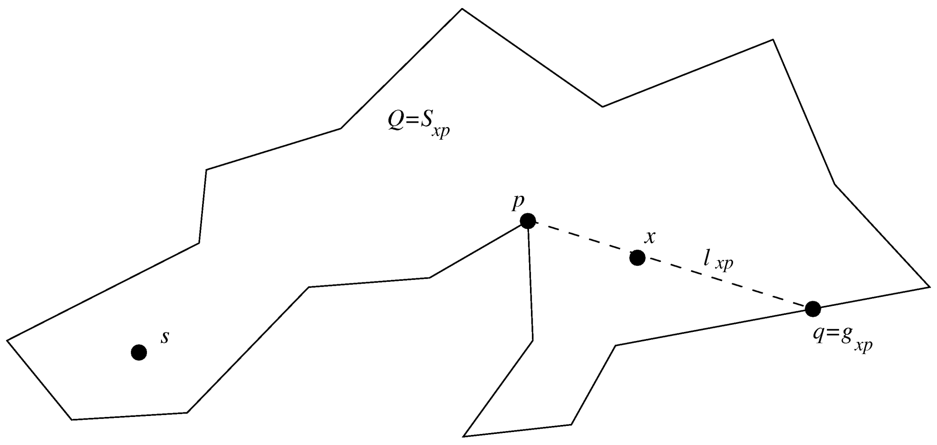

Fig. 5. An example of a cut.

$L[x_j,t]$ do not intersect (x_i,x_j). But then we can replace the subpath $L[x_i,x_j]$ by the segment $[x_i,x_j]$. By repeating this process we could eliminate all shortcuts. $\square$

If $p,q\in\partial P$ see each other in P then the line segment $[p,q]$ is called a *cut*. Each cut $[p,q]$ partitions P into two polygons Q, Q' that are disjoint except for sharing edge $[p,q]$. If $p\in V_P$, $s\in Q$, and $x\in(p,q)$, then we introduce the following notation: $\ell_{xp}=[p,q]$, $g_{xp}=q$ and $S_{xp}=Q$ (see Fig. 5). Sometimes, to avoid double subscripts, we will write $\ell(x,p)$, $S(x,p)$ and $g(x,p)$ instead of ℓ_{xp}, S_{xp} and g_{xp}, respectively. If $x=s$ then S_{sp} is not uniquely defined. In this case, we choose S_{sp} to be the one of Q, Q' that does not contain t.

Lemma 3. *If t is not visible from s then there is a vertex $r\in V_P$ visible from s such that any shortcut-free simple (s,X,t)-path L does not intersect the line segment (s,g_{sr}).*

Proof. Since t is not visible from s, there is a vertex $r\in V_P$, visible from s, such that the line segment ℓ_{sr} is tangent to ∂P at r. If we extend ℓ_{sr} beyond r, it will intersect ∂P at a point h_r. Any simple (s,X,t)-path L intersects the segment $[r,h_r]$. Then, by Lemma 1, L cannot intersect (s,g_{sr}). $\square$

Lemma 4. *Suppose that $L=x_0x_1\ldots x_k$ is a simple shortcut-free (s,X,t)-path in P. Then there exist vertices $p_0,p_1,\ldots,p_k\in V_P$, that satisfy the following conditions:*
(a) *$S(x_{i-1},p_{i-1})\subset S(x_i,p_i)$ for all $i=1,\ldots,k$,*
(b) *$L[x_0,x_i]\subset S(x_i,p_i)$ for $i=0,\ldots,k$.*

Proof. The lemma is trivial for $k=0,1$, so we assume $k\geqslant 2$. We construct the points p_i one by one. First we show how to construct p_0. Let ξ be a mobile point that is initially x_1. We slide ξ towards x_2 until $[s,\xi]$ hits ∂P at a vertex u. While we move ξ, the segment (s,ξ) cannot intersect L, since otherwise we would obtain a shortcut (this intersection would occur at a vertex of L other than x_1). Furthermore, by Lemma 1, L does not intersect (s,g_{su}). Thus we can set $p_0=u$.

Suppose we already constructed $p_0,\ldots,p_{i-1}$. Let Q be the polygon consisting of the points z such that (z,x_i) does not intersect neither ∂P nor $\ell(x_{i-1},p_{i-1})$. Let ξ be initially x_{i-1}. We will move point

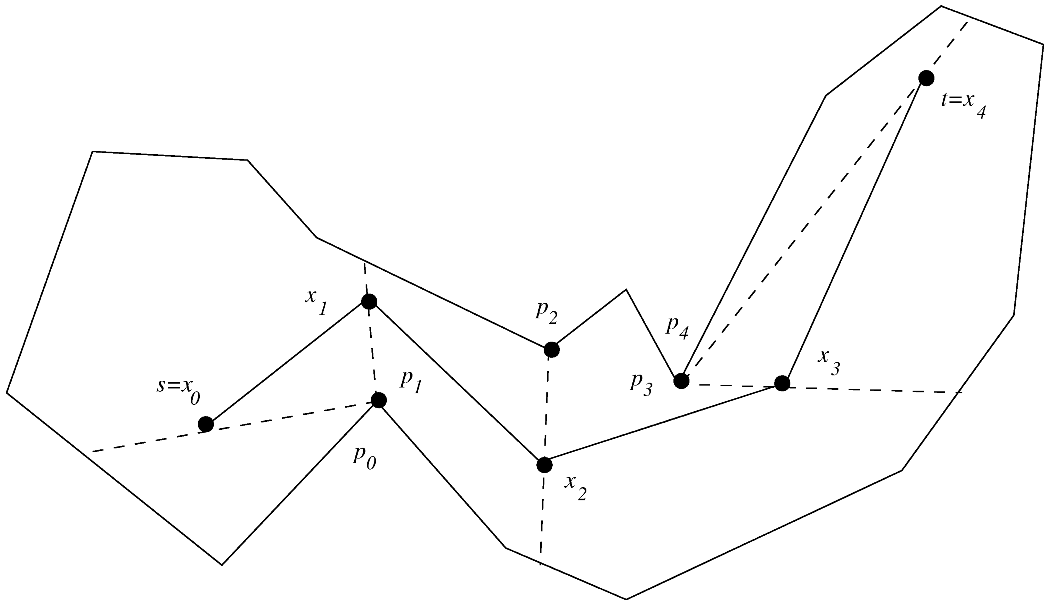

Fig. 6. Example of a simple (s, X, t)-path and a corresponding sequence of cuts.

ξ along the edges of ∂Q and examine the intersection points of $\ell(x_i, \xi)$ with ∂P and L. We distinguish three cases.

Suppose, first, that $i = k$. In this case, move ξ along ∂Q in the direction of p_{i-1}, and stop when $[x_i, \xi]$ touches a vertex u of P. The cut $\ell(x_i, u)$ does not intersect $\ell(x_{i-1}, p_{i-1})$ (except possibly at p_{i-1}, if $u = p_{i-1}$). So we can set $p_k = u$.

Throughout the rest of the proof we can assume that $i < k$. Let α be the line through x_{i-1} and x_i. Suppose now x_{i+1} is on the same side of α as p_{i-1}. We move ξ along ∂Q in the direction of p_{i-1}. Similarly as before, $[x_i, \xi]$ must touch a vertex u of P. Throughout this movement (x_i, ξ) does not intersect L, since otherwise we would obtain a shortcut. If $(x_i, g(x_i, u))$ does not intersect L, we can set $p_i = u$ and we are done.

Otherwise, if $(x_i, g(x_i, u))$ intersects L, then let $\eta \in [x_j, x_{j+1})$, for $j > i$, be the intersection point closest to x_i. Now we move η towards x_{j+1}. Segment $[x_i, \eta]$ must intersect ∂P before η reaches x_{j+1}, since otherwise we would have a shortcut $[x_i, x_{j+1}]$. Let v be this intersection point. By the construction, L does not intersect (x_i, v) and, by Lemma 1, L does not intersect $(x_i, g(x_i, v))$. Thus we can set $p_i = v$.

The remaining case is when x_{i+1} is on the same side of α as $g = g(x_{i-1}, p_{i-1})$. Let q the endpoint of the edge of P that contains g which does not belong to $S(x_{i-1}, p_{i-1})$. In this case, we move ξ along ∂Q in the direction of q. Since the internal angle in Q at g is less than 180°, and since q is a vertex of P, the segment $(x_i, \xi]$ cannot intersect L before $[x_i, \xi]$ touches some vertex u of P. The rest of the argument is the same as in the previous case. □

Algorithm SimplePath. Without loss of generality assume that t is not visible from s in P. We consider first the *boundary case*, when $s \in V_P$. For convenience, we think of s as consisting of three points: s, s' and s'', where s is treated as a point of X and s' and s'' as vertices of P, and we introduce the cut $\ell_{ss'}$, for which $S_{ss'} = \{s\}$ and $g_{ss'} = s''$.

First, create a directed graph $D = (C, A)$, where C is the set of all cuts ℓ_{xp}, for $x \in X$ and $p \in V_P$. The arcs in A are determined as follows: given any two cuts ℓ_{xp} and ℓ_{yq}, we create an arc $\ell_{xp} \to \ell_{yq}$ if y is visible from x in P and $S_{xp} \subset S_{yq}$. Obviously, D is acyclic.

If there is no path from $\ell_{ss'}$ to some ℓ_{tu} in D, then report that there is no simple (s, X, t) path in P. Otherwise, let K be a path from $\ell_{ss'}$ to ℓ_{tu} in D. The actual (s, X, t)-path L is reconstructed from K in D in the following way. If ℓ_{xp} is on K then x is on L, and if ℓ_{yq} is the first vertex on K after ℓ_{xp} then y follows x on L.

Correctness. If $[x, y]$ is an edge of L then $\ell_{xp} \to \ell_{yq}$ is an arc of D for some $p, q \in V_P$. This means that y is visible from x in P. Therefore L does not intersect ∂P. We also have $S_{xp} \subset S_{yq}$, which implies that (x, y) does not intersect $L[s, x]$. Thus, L is simple. On the other hand, by Lemma 4, if there exists any simple (s, X, t)-path in P, then some path will be found by our algorithm.

Time complexity. Let n be the number of points in X and m the number of vertices in P. We now estimate the time complexity. The visibility graph $\mathcal{G}_P(X)$ of X in P can be computed in time $O((n+m)^2)$. We have at most m^2n^2 pairs ℓ_{xp}, ℓ_{yq} of cuts. For each such pair we want to determine if there is an arc $\ell_{xp} \to \ell_{yq}$ in D. To implement it efficiently, for each cut ℓ_{xp} compute the point g_{xp}. With $O(m)$ preprocessing, this point can be determined in time $O(\log m)$ using the ray shooting algorithm from [4]. Then we sort all the points g_{xp} based on their order of appearance when we traverse the polygon clockwise, starting from s' and ending at s''. Using this ordering, we can in time $O(1)$ determine if $\ell_{xp} \to \ell_{yq}$ is an arc: simply check whether (a) y is visible from x, and (b) whether $\min(p, g_{xp}) < q, g_{yq} < \max(p, g_{xp})$.

General case. Suppose now that $s \in P$ is arbitrary. We construct another polygon P' in which s is a vertex. Let r be the vertex from Lemma 3, and let $[u, v]$ be the edge of P that contains g_{sr}. Introduce two points $u', v' \in [u, v]$ such that the ordering of the points along $[u, v]$ is u, u', g_{sr}, v', v, and u', v' are close enough to g_{sr} so that the triangle (u', s, v') does not contain any points of X. Create polygon P' obtained from P by replacing edge $[u, v]$ by four edges $[u, u']$, $[u', s]$, $[s, v']$ and $[v', v]$.

The correctness of the algorithm follows from Lemma 3, which implies that P has a simple (s, X, t)-path if and only if P' has a simple (s, X, t)-path. Polygon P' can be easily computed in time $O(m + n)$ using the visibility polygon of s [5], and point location algorithms (see [13, pp. 45–67]). We summarize the discussion above in the following theorem.

Theorem 2. *Algorithm SimplePath computes simple (s, X, t)-paths inside a simple polygon P in time* $O(m^2n^2)$.

4. Final comments

We presented the NP-completeness result for arbitrary obstacles and a polynomial-time algorithm for the case when Φ is a simple polygon.

There are several possible directions for further research. The first question is whether it is possible to compute simple (s, X, t)-paths in P substantially faster than $O(m^2n^2)$. In particular, is it possible to do it in time $O(mn(m + n))$?

With some minor modifications to Algorithm SimplePath, we can obtain an $O(m^2n^2)$-time algorithm to compute simple (s, X, t)-paths *outside* a simple polygon P. (Lemma 1 remains true in this case. Using this lemma, we can reduce the "outside" case to the "inside" case.) One possible application of this algorithm is to generate all simple polygons with vertices in X with only polynomial-time overhead (that is, the time complexity is bounded by a polynomial multiplied by the number of generated polygons). In order to do so, in the incremental algorithm for generating polygons, the path Q generated so far is treated as a thin polygon, and we use Algorithm SimplePath to check if there is a simple path from the last vertex on Q to the first vertex on Q that avoids Q. Note that in this application the given set X and the given polygon P are subject to only local changes at each step. Thus it would be interesting to develop a dynamic algorithm for this problem.

Sometimes we may wish to generate a polygon that uses *all* points of X, not just a subset. This naturally leads to the problem of computing *simple Hamiltonian (s, X, t)-paths* (that is, simple (s, X, t)-paths that visit all points of X) that avoid Φ. It is easy to see that the problem is NP-complete for arbitrary obstacles, so we restrict our attention to the case when $\Phi = P$ is a simple polygon and X is inside (or outside) P. When P is convex, a simple Hamiltonian (s, X, t)-path that avoids P always exists and can be computed in time $O(n \log n)$ by using angular orderings of the points in X in an appropriate fashion. However, the status of this problem when P is an arbitrary simple polygon remains open.

It is an open problem whether simple polygons whose vertex set is X can be counted (or generated uniformly) in polynomial time. Auer and Held [2] considered several heuristics for this problem. One of their heuristics is, in essence, the same incremental method as the one we described in the introduction. They reduce the backtracking by storing the inventory of usable edges, that is the edges that do not intersect the current path. An efficient algorithm that computes Hamiltonian simple (s, X, t)-paths outside a given polygon P could be used to eliminate all useless branching.

We would also like to mention the result of Alsuwaiyel and Lee [1] that finding a Hamiltonian (s, X, t)-path (not necessarily simple) in a simple polygon P is NP-complete. Their proof works even in the special case when X is restricted to be the vertex set of P. (Note that the boundary of P is not a feasible solution if s and t are not consecutive.)

Acknowledgements

We would like to thank David Eppstein for helpful discussions and pointing out several references. We would also like to thank an anonymous referee for comments that helped us improve the presentation.

References

[1] M.H. Alsuwaiyel, D.T. Lee, Minimal link visibility paths inside a simple polygon, Computational Geometry 3 (1993) 1–25.
[2] T. Auer, M. Held, Heuristics for the generation of random polygons, in: F. Fiala, E. Kranakis, J.-R. Sack (Eds.), Proc. 8th Canadian Conference on Computational Geometry, Carleton University Press, Ottawa, Canada,12–15 August 1996, pp. 38–44.
[3] M. Babikov, D. Souvaine, R. Wenger, Constructing piecewise linear homeomorphisms of polygon holes, in: Proc. 9th Canadian Conference on Computational Geometry, 1997, pp. 6–10.
[4] B. Chazelle, H. Edelsbrunner, M. Grigni, L. Guibas, J. Hershberger, M. Sharir, J. Snoeyink, Ray shooting in polygons using geodesic triangulations, Algorithmica 12 (1994) 54–68.

[5] H. ElGindy, D. Avis, A linear time algorithm for computing the visibility polygon from a point, J. Algorithms 2 (1981) 186–197.
[6] S. Even, G. Granot, Rectilinear planar drawings with few bends in each edge, Technical Report CS0797, Computer Science Department, Technion, Israel Institute of Technology, 1994.
[7] S. Ghosh, D. Mount, An output-sensitive algorithm for computing visibility graphs, SIAM J. Comput. 20 (1991) 888–910.
[8] M.R. Garey, D.S. Johnson, Computers and Intractability, A Guide to the Theory of NP-Completeness, Freeman, New York, 1979.
[9] L.J. Guibas, J. Hershberger, Optimal shortest path queries in a simple polygon, J. Comput. System Sci. 39 (1989) 126–152.
[10] L. Guibas, J. Hershberger, D. Leven, M. Sharir, R. Tarjan, Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons, Algorithmica 2 (1987) 209–233.
[11] J. Hershberger, An optimal visibility graph algorithm for triangulated simple polygons, Algorithmica 4 (1989) 141–155.
[12] J. Plesnik, The NP-completeness of the Hamiltonian cycle problem in planar digraphs with degree bound two, Inform. Process. Lett. 8 (1979) 199–201.
[13] F.P. Preparata, M.I. Shamos, Computational Geometry, An Introduction, Springer, Berlin, 1985.
[14] R. Tamassia, I.G. Tollis, Planar grid embeddings in linear time, IEEE Trans. Circuits Systems 9 (1989) 1230–1234.
[15] C. Zhou, G. Sundaram, J. Snoeyink, J.S.B. Mitchell, Generating random polygons with given vertices, Computational Geometry 6 (1996) 277–290.