ELSEVIER

Information Processing Letters 77 (2001) 23–26

Information Processing Letters

www.elsevier.com/locate/ipl

Shortest zookeeper's routes in simple polygons

Xuehou Tan

School of High-Technology for Human Welfare, Tokai University, 317 Nishino, Numazu 410-0321, Japan

Received 1 September 1999; received in revised form 11 July 2000
Communicated by F.Y.L. Chin

Abstract

Let P be a simple polygon, and let $\mathcal{P}$ be a set of disjoint convex polygons inside P, each sharing one edge with P. The *zookeeper's route problem* asks for a shortest route inside P that visits (but does not enter) each polygon in $\mathcal{P}$. We present an $O(n^2)$ time algorithm for computing a shortest zookeeper's route, on which no starting point is specified. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Computational geometry; Zookeeper's routes; Unfolding; Adjustments

1. Introduction

Shortest paths are of fundamental importance in robotics and computational geometry. The *zookeeper's route problem*, introduced by Chin and Ntafos [3], is defined as follows: Given a simple polygon P (the *zoo*) with a set $\mathcal{P}$ of disjoint convex polygons (the *cages*) inside it, each sharing one edge with polygon P, find a shortest route inside P that visits (without entering) each polygon in $\mathcal{P}$. One may consider it as minimizing the route for a zookeeper to feed animals. In some sense, the zookeeper's route problem looks like the well-known *Traveling Salesperson problem* if we consider cages as cities. But, we actually know that the shortest zookeeper route has to visit cages in the order they appear in the boundary of P, since otherwise it would cross itself and could be shortened [3].

Let n denote the total number of edges of polygon P and polygons in $\mathcal{P}$. With the unfolding and adjusting techniques, Chin and Ntafos gave an $O(n^2)$ time algorithm for computing the shortest *fixed* zookeeper's route, i.e., the route is forced to pass through a starting point s on the boundary of P [3]. This result was later improved to $O(n \log^2 n)$ by Hershberger and Snoeyink [7], using a complicated data structure for shortest-path queries in a simple polygon [5,6]. In this note, we present an $O(n^2)$ time algorithm for computing a shortest zookeeper's route, on which no starting point is specified. In some cases, the shortest zookeeper's route without any restrictions can be much shorter than the restricted one.

2. The main result

Let us first give a brief review of Chin and Ntafos' algorithm for computing the shortest fixed zookeeper's route [3]. Suppose that we have a set E of edges, one per cage. The locally optimal zookeeper's route, which makes contacts with the edges of E, can be computed by triangulating the interior of $P - \mathcal{P}$ (Fig. 1(a)), unfolding the triangulation using edges in E as mirrors and finding the shortest path between s and its image s' in the unfolded polygon (Fig. 1(b)), and finally

E-mail address: tan@wing.ncc.u-tokai.ac.jp (X. Tan).

0020-0190/01/$ – see front matter
PII: S0020-0190(00)00144-7

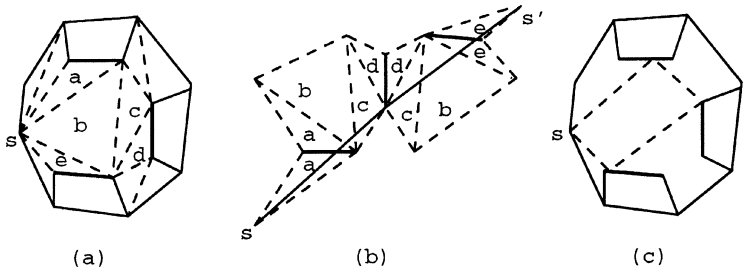

Fig. 1. Construction of a zookeeper's route.

folding back the shortest path to get the zookeeper's route (Fig. 1(c)). See Fig. 1 for an example, where the edges in E are drawn in bold lines. This process takes linear time, since a simple polygon can be triangulated in linear time [1] and the shortest path between two points in a triangulated polygon can be found in linear time [4]. Specially, we call the edges of E, *active edges*.

Chin and Ntafos' algorithm first selects an initial set E_0 of edges, one per cage, and computes the initial zookeeper route R_0 using the unfolding method. The shortest fixed zookeeper's route R is then found by repeatedly adjusting the current route R_i until it becomes the shortest one. Let e_x and e_y denote two adjacent edges of a cage P_j, and let $p_{x,y}$ denote the common point of e_x and e_y. Route R_i is *adjustable* on the active edge e_x if the contact of R_i with e_x is only the point $p_{x,y}$ and the incoming (outgoing) angle formed by R_i with the line containing edge e_y is smaller than the outgoing (incoming) angle formed by R_i with the line containing edge e_y. Fig. 2 shows two different cases in which an adjustment occurs. An adjustment involves a change in the set of active edges (i.e., e_y is substituted for e_x in the new set E_{i+1}) and thus calls the unfolding procedure once to compute the new and shorter zookeeper's route. In the adjusting process, the length of the current route decreases monotonically. When route R_i cannot be adjusted any more, it gives the shortest fixed zookeeper route. The initial edge set E_0 is so selected that if one unfolds both R_0 and the shortest fixed zookeeper's route R, then R_0 lies in the same side of R throughout its extent. Thus, the current route R_i moves along the boundary of any cage P_k in a single direction, i.e., the contact points of the computed routes are

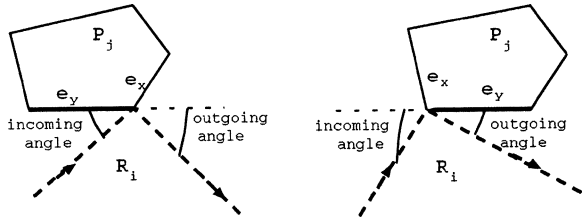

Fig. 2. Definition of adjustments.

well ordered on the boundary of P_k in the adjusting direction. Since at most n adjustments are performed, Chin and Ntafos' algorithm simply runs in $O(n^2)$ time [3]. (Later, Hershberger and Snoeyink showed that an adjustment can be done in $O(\log^2 n)$ time, which leads to an $O(n \log^2 n)$ time algorithm [7].)

In the following, we give a method to remove the starting point constraint. First, we place the point s at the leftmost vertex of some cage P_i, and then move s along the boundary of P_i. The general zookeeper's route problem (without giving any starting point) is solved if we can find the shortest zookeeper's route forced to pass through each point of the boundary of P_i. Although there is an infinite number of such routes, we show that only a linear number of points (*event points*) on the boundary of P_i needs to be considered and each event can be dealt with in linear time. (It is somewhat similar to the work of Chen and Daescu for maintaining the visibility of a moving point in a simple polygon [2].)

Let R_s denote the shortest fixed zookeeper's route passing through s. We are interested in topological changes occurred to R_s when the point s is slid from the leftmost of P_i to the rightmost. Since R_s is the shortest zookeeper's route passing through s, the unfolded version of the route R_s is the shortest

path between s and its image s'. Thus, it makes turns only at the vertices of the unfolded polygon. Since the sliding effect is blocked by the first and last turn points, only the lengths of the first and last segments of the unfolded route R_s change in the sliding process. Hence, topological changes to the route happen if one of the followings occurs.

(1) The first (last) segment of the unfolded route R_s has to be split into two. It occurs when the end vertex of an active edge or a vertex of polygon P is encountered.
(2) The first (last) two segments of the unfolded route R_s become collinear. It occurs at the starting vertex of an active edge or a vertex of polygon P.
(3) The point s reaches the end vertex of the active edge of P_i.
(4) The first segment and the last segment of the unfolded route R_s have the same angle to the edge of P_i containing s. This event leads to a locally optimal route, which is likely the shortest of all zookeeper's routes.

We call the points of s on the boundary of P_i where topological changes to the route happen the *event points* of the sliding process. Four types of event points are shown in Fig. 3. Note that if the number of cages is odd, the arrows on the top and bottom in Fig. 3 should have different directions.

Lemma 1. *There are* $O(n)$ *event points in the sliding process.*

Proof. Since the direction of contact points slid on any active edge need never be changed (Fig. 3), the number of Type 1 and Type 2 event points is bounded by the number of vertices of polygon P and polygons in $\mathcal{P}$. Note that a vertex of a polygon in $\mathcal{P}$ may act as two event points (a Type 1 event and a Type 2 event). Clearly, the number of Type 3 event points is bounded by the number of vertices of P_i. There may be an infinite number of Type 4 event points in an edge of P_i. But, their fixed zookeeper's routes are of the same length. So we need to select only one representative point, or an event point as we called. Thus, the number of Type 4 event points is bounded by the size of P_i. □

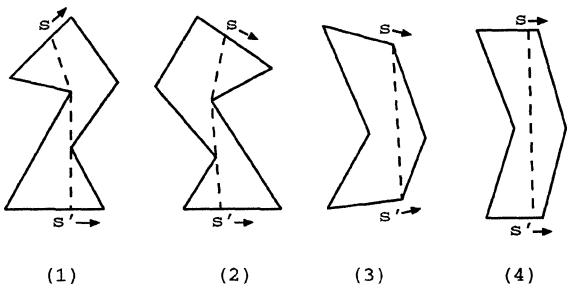

Fig. 3. Four types of event points.

When an event point is encountered on the boundary of P_i, what kind of changes should be made to the route R_s and what is the time taken to maintain the shortest zookeeper's route passing through the new event point? We answer these questions in the following lemma.

Lemma 2. *The shortest fixed zookeeper's route* R_s *can be maintained in linear time when a new event point is encountered in the sliding process.*

Proof. Note first that for any event point s, we need to record the length of the shortest fixed zookeeper's route R_s. For a Type 1 event point, if it corresponds to a vertex of polygon P, there are no changes in the set of active edges. If it corresponds to the end vertex of an active edge, this edge is deleted from the set of active edges and the next edge (if it exists) is inserted into the set, and then we compute the new route R_s (i.e., the shortest zookeeper's route passing through the new event point) by the unfolding method. Analogously, a Type 2, Type 3 or Type 4 event point can be dealt with.

Consider now how to find the nearest event point. We will describe a method to compute four potential event points in the edge containing s, one for each type of events. The nearest event point can then be computed from these (at most) four candidates. For the next Type 1 event point, we first consider the case where the unfolded route R_s is a straight line segment. If the point s and its image s' have opposite sliding directions in the unfolded polygon (i.e., the number of cage is odd), we take the middle point of the unfolded route R_s as the center and rotate R_s in the sliding direction. Otherwise, the whole route R_s is translated in the sliding direction. When a vertex of the unfolded polygon is first encountered by this rotation or translation, the next Type 1 event point occurs. If the unfolded route R_s makes turns at some vertices of the

unfolded polygon, the computation can be similarly done by rotating the first and last segments of route R_s, fixing centers at the vertices where R_s makes the first and last turns, respectively.

To find the next Type 2 event point, we rotate the first and last segments of the unfolded route R_s in the sliding direction and report the place of s where the first or last two segments of R_s become collinear. It is trivial to find the next Type 3 event point, as it is the end vertex of the edge containing s.

For the next Type 4 event point, we first note that there may be an infinite number of Type 4 event points in a continuous interval of an edge of P_i and we need to pick up only one representative point. If the unfolded route R_s is a straight line segment, the position of s that minimizes the distance between the edge (e_1, e_2) and its unfolded image (e'_1, e'_2) gives the potential Type 4 event point, where e_1 and e_2 are two endpoints of the edge containing s. (Given (e_1, e_2) and (e'_1, e'_2), the position of s minimizing the edge-distance can be computed in constant time. This is because the unfolded polygon needn't be considered in the computation.) If the unfolded route R_s is not a straight line segment, we put together two unfolded polygons: one is used to compute the route R_s which visits all cages in the clockwise order and placed to the left of the edge containing s, and the other is used to compute the route R_s which visits all cages in the counterclockwise order and placed to the right of the edge containing s. Then draw a line segment to connect two first turn points in different unfolded polygons. If it wholly lies in the union of two unfolded polygons, the intersection of the line segment with the edge containing s gives the next Type 4 event point. Otherwise, no Type 4 event points exist.

Since any type of next event point can be found in linear time, the lemma is proved. □

Theorem 1. *The general shortest zookeeper's route in a simple polygon* (*without giving any starting point*) *can be found in* $O(n^2)$ *time.*

Proof. First, the initial shortest fixed zookeeper's route can be computed in $O(n \log^2 n)$ time [7]. In the sliding process, we maintain the length of the shortest zookeeper's route passing through each event point. Since the length of a route is either monotonically decreasing or increasing between two consecutive event points, the shortest one has to pass through one of the event points. The shortest zookeeper's route is then the shortest of the routes passing through the event points. Since there are $O(n)$ event points (Lemma 1) in the sliding process and each event takes $O(n)$ time (Lemma 2), the time complexity of our algorithm is $O(n^2)$. □

3. Conclusions

We have given an $O(n^2)$ time algorithm for computing a shortest zookeeper's route, on which no starting point is specified. The time bound of our algorithm might be further reduced (say, to $O(n \log^2 n)$). Note that the data structure of Hershberger and Snoeyink [7] can be used to compute the shortest zookeeper's route at an event point. Whether or not the next event point can be found in sublinear time is left as an interesting open problem.

References

[1] B. Chazelle, Triangulating a simple polygon in linear time, in: Proc. 31th Annual IEEE Symp. Foundations of Comput. Sci., 1990, pp. 220–229.
[2] D.Z. Chen, O. Daescu, Maintaining visibility of a polygon with a moving point of view, Inform. Process. Lett. 65 (1998) 269–275.
[3] W.P. Chin, S. Ntafos, The zookeeper route problem, Inform. Sci. 63 (1992) 245–259.
[4] L. Guibas, J. Hershberger, D. Leven, M. Sharir, R. Tarjan, Linear time algorithms for visibility and shortest path problems inside simple triangulated polygons, Algorithmica 2 (1987) 209–233.
[5] L. Guibas, J. Hershberger, Optimal shortest path queries in a simple polygons, J. Comput. System Sci. 39 (1989) 231–235.
[6] J. Hershberger, A new data structure for shortest path queries in a simple polygon, Inform. Process. Lett. 38 (1991) 231–235.
[7] J. Hershberger, J. Snoeyink, An efficient solution to the zookeeper's problem, in: Proc. 6th Canadian Conf. on Computational Geometry, 1994, pp. 104–109.