ELSEVIER

Information Processing Letters 59 (1996) 207-210

Information Processing Letters

Finding a shortest Hamiltonian path inside a simple polygon

Muhammad H. Alsuwaiyel [1]

Department of Information and Computer Science, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia

Received 1 February 1995; revised 1 June 1996
Communicated by S.G. Akl

Keywords: Computational geometry; Simple polygon; Hamiltonian path; Watchman route

1. Introduction

Given a set S of n points on the boundary of a Simple Polygon P, we are concerned with the problem of finding a Hamiltonian path on S (i.e., a path that visits each point in S exactly once) of minimum Euclidean length. We present an $O(n^3)$ time algorithm that finds $n-1$ optimal Hamiltonian paths from a fixed source point to each other point in S. Hence, by repeating this algorithm $n-1$ times, a Hamiltonian path of minimum Euclidean length with no fixed endpoints can be found in time $O(n^4)$. The algorithm presented works for any simply connected region provided that the inter-point distances are part of the input. In [1], Alsuwaiyel and Lee have shown that this problem is NP-complete if the Euclidean distance is replaced with the link-distance metric [2].

Given a simple polygon P, a watchman route for P is a route within P with the property that every point in P is visible from at least one point along the route. The problem of finding a watchman route of shortest total length is fundamental in computational geometry. This problem has first been investigated by Chin and Ntafos who gave a linear time algorithm for the special case when P is rectilinear [3]. In the case of general simple polygons, an algorithm that runs in $O(n^3)$ time was given in [2]. But the problem of finding a watchman *path* of minimum Euclidean length inside a simple polygon is still open. In general, to find such a path or a tour, the problem reduces to finding a set of convex chains on the boundary of the given input polygon which have to be intersected by a (polygonal) path of shortest length [1-3]. It is not obvious whether the algorithm presented here can be generalized to visiting a set of convex chains, instead of points, and thus proving that the problem of finding a shortest watchman path is indeed solvable in polynomial time. It is not even obvious in the case of rectilinear polygons in which the chains degenerate into line segments.

2. The algorithm

Let $S = \{a_0, a_1, \ldots, a_{n-1}\}$ be a set of n points on the boundary of a simple polygon P sorted in counterclockwise order. In what follows, we give an algorithm that finds $n-1$ paths $\pi_1, \pi_2, \ldots, \pi_{n-1}$, where

[1] Email: suwaiyel@ccse.kfupm.edu.sa.

[2] The link-distance between two points x and y in a simple polygon P is defined to be the minimum number of line segments in a polygonal path contained in P that connects x and y.

PII S0020-0190(96)00103-2

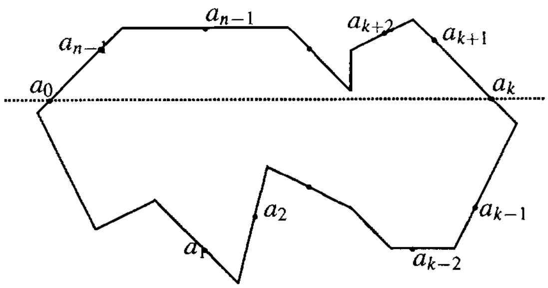

Fig. 1. Partitioning the set of points.

each path π_k, $1 \leqslant k \leqslant n-1$, is a Hamiltonian path of shortest Euclidean length that starts at the designated source point a_0 and ends at point a_k.

Consider computing a Hamiltonian path π_k of shortest length from a_0 to a_k, and assume without loss of generality that the line passing through a_0 and a_k is horizontal. Partition the set of points S into two sequences: an upper sequence $U = a_{n-1}, a_{n-2}, \ldots, a_k$ and a lower sequence $L = a_0, a_1, \ldots, a_{k-1}$ (see Fig. 1). We say that portions of π_k overlap or, equivalently, π_k intersects itself if, when tracing the path from a_0 to a_k, a point $p \in \pi_k$ is encountered more than once. It is not difficult to see that if π_k intersects itself, then it can be transformed into another path π'_k that is not self-intersecting and of strictly shorter length. As a consequence, we have the following lemma.

Lemma 1. *Let π_k be a Hamiltonian path of shortest length that starts at point a_0 and ends at point a_k. Then, π_k must visit the points $a_1, a_2, \ldots, a_{k-1}$ in this order, and the points $a_{n-1}, a_{n-2}, \ldots, a_{k+1}$ in this order.*

It follows that π_k oscillates between the upper and lower sequences, visiting the points in each sequence as described in Lemma 1. Suppose that, at some instant, an object that is traversing path π_k is at point a_j in the upper sequence, say, after it has already visited $a_{n-1}, a_{n-2}, \ldots, a_{j-1}$ and $a_0, a_1, \ldots, a_i$ in the upper and lower sequences respectively, where $0 \leqslant i < k < j \leqslant n-1$. Then, by Lemma 1, the next point to be visited by this object must be either a_{i+1} or a_{j+1}. This implies the following approach to compute π_k, $1 \leqslant k \leqslant n-1$.

With each point a_k, $1 \leqslant k \leqslant n-1$, we associate k subpaths $\Pi_0^k, \Pi_1^k, \ldots, \Pi_{k-1}^k$, where each Π_j^k is a Hamiltonian path of shortest length from a_0 to a_k that visits the points $a_{n-1}, a_{n-2}, \ldots, a_{k+1}$ and $a_1, a_2, \ldots, a_j$. For each pair k and j, $0 \leqslant j < k \leqslant n-1$, let A_j^k denote the length of Π_j^k. Thus, $\pi_k = \Pi_{k-1}^k$ and A_{k-1}^k is the length of π_k. In what follows, we will let A^k denote the array with the k entries $A_0^k, A_1^k, \ldots, A_{k-1}^k$.

The arrays A^k, $1 \leqslant k \leqslant n-1$, are computed in the order

$$A^{n-1},\ A^{n-2},\ \ldots,\ A^1.$$

For each pair i and j, $0 \leqslant i < j \leqslant n-1$, define

$$D_{i,j} = d_{i,i+1} + d_{i+1,i+2} + \cdots + d_{j-1,j},$$

where $d_{i,j}$ denotes the length of the shortest *geodesic* path from a_i to a_j.

We proceed to compute the arrays $A^{n-1}, A^{n-2}, \ldots, A^1$ inductively as follows:

$$A_0^{n-1} = d_{0,n-1},$$

$$A_1^{n-1} = D_{0,1} + d_{1,n-1},$$

$$A_2^{n-1} = D_{0,2} + d_{2,n-1},\ \ldots,$$

$$A_{n-2}^{n-1} = D_{0,n-2} + d_{n-2,n-1}.$$

For $k = n-2, n-3, \ldots, 1$, A^k is computed from A^{k+1} using the following equations:

$$A_0^k = A_0^{k+1} + d_{k+1,k},$$

$$A_1^k = \min\{A_0^{k+1} + d_{k+1,1} + d_{1,k}, A_1^{k+1} + d_{k+1,k}\},$$

$$A_2^k = \min\{A_0^{k+1} + d_{k+1,1} + D_{1,2} + d_{2,k},\ A_1^{k+1} + d_{k+1,2} + d_{2,k}, A_2^{k+1} + d_{k+1,k}\},$$

$$\vdots$$

$$A_j^k = \min\{A_0^{k+1} + d_{k+1,1} + D_{1,j} + d_{j,k},\ A_1^{k+1} + d_{k+1,2} + D_{2,j} + d_{j,k},\ \ldots,\ A_i^{k+1} + d_{k+1,i+1} + D_{i+1,j} + d_{j,k},\ \ldots,\ A_j^{k+1} + d_{k+1,k}\},$$

$$\vdots$$

$$A^k_{k-1} = \min\{A^{k+1}_0 + d_{k+1,1} + D_{1,k},\ A^{k+1}_1 + d_{k+1,2} + D_{2,k},\ \ldots,\ A^{k+1}_i + d_{k+1,i+1} + D_{i+1,k},\ \ldots,\ A^{k+1}_{k-1} + d_{k+1,k}\}.$$

The computation of the actual paths is postponed after all the A^k arrays have been computed in order to avoid an $O(n^3)$ storage ($O(n)$ for each subpath Π^k_j). For this purpose, we associate with each array entry A^k_j the index of the "split point" B^k_j of the subpath Π^k_j defined as follows: If in the computation of

$$A^k_j = \min\{A^{k+1}_0 + d_{k+1,1} + D_{1,j} + d_{j,k},\ A^{k+1}_1 + d_{k+1,2} + D_{2,j} + d_{j,k},\ \ldots,\ A^{k+1}_i + d_{k+1,i+1} + D_{i+1,j} + d_{j,k},\ \ldots,\ A^{k+1}_j + d_{k+1,k}\}$$

the quantity

$$A^{k+1}_i + d_{k+1,i+1} + D_{i+1,j} + d_{j,k}$$

minimizes A^k_j for some i, $0 \leqslant i \leqslant j$, then we set the split point $B^k_j = i$.

After all arrays $A^{n-1}, A^{n-2}, \ldots, A_1$ have been computed, their corresponding actual paths are constructed using the following recurrence:

$$\Pi^k_j = \begin{cases} a_0, a_1, \ldots, a_j, a_k & \text{if } k = n-1, \\ \Pi^{k+1}_j, a_k & \text{if } B^k_j = j \text{ and } k \leqslant n-2, \\ \Pi^{k+1}_{B^k_j}, a_{B^k_j} + 1, \ldots, a_k & \\ \quad \text{if } B^k_j < j \text{ and } k \leqslant n-2. \end{cases}$$

It is not hard to see that each path Π^k_j is constructed in $O(n)$ time and $O(n)$ space. The correctness of the algorithm follows from the following lemma.

Lemma 2. *The paths Π^k_j, $0 \leqslant j < k \leqslant n-1$, as computed by the algorithm above are of shortest length.*

Proof. By induction on k. Obviously, $\Pi^{n-1}_0 = a_0, a_{n-1}$, and by Lemma 1, the only shortest Hamiltonian path from a_0 to a_{n-1} that visits the points $a_1, a_2, \ldots, a_j$ is $a_0, a_1, a_2, \ldots, a_j, a_{n-1}$, for each j, $1 \leqslant j \leqslant n-2$. Assume that for some k, $1 \leqslant k \leqslant n-2$, and for all j, $0 \leqslant j \leqslant k$, A^{k+1}_j is minimized by the algorithm. We show that each A^k_j, $0 \leqslant j \leqslant k-1$, is also minimized by the algorithm. Clearly, $A^k_0 = A^{k+1}_0 + d_{k+1,k}$ is of minimum length. By construction, for any j, $1 \leqslant j \leqslant k-1$, Π^k_j is the concatenation of the two subpaths

$$\Pi^{k+1}_i \quad \text{and} \quad a_{k+1}, a_{i+1}, a_{i+2}, \ldots, a_j, a_k,$$

where $i = B^k_j$, the split point of Π^k_j. By induction, Π^{k+1}_i is a path of shortest length from a_0 to a_{k+1} that visits the points

$$a_{n-1}, a_{n-2}, \ldots, a_{k+1} \quad \text{and} \quad a_1, a_2, \ldots, a_i,$$

and, by Lemma 1, $a_{k+1}, a_{i+1}, a_{i+2}, \ldots, a_j, a_k$ is the *unique* optimal path from a_{k+1} to a_k that visits the points $a_{i+1}, a_{i+2}, \ldots, a_j$. Since the algorithm selects i that minimizes the sum of the lengths of these two subpaths, it follows that Π^k_j must be of shortest length as well. □

We analyze the time and space required by the algorithm as follows. First, the inter-point distances can be computed in $O(n^2)$ time [4]. Since the number of operations required to compute A^k_j is $O(j)$, the total cost of computing A^k is $\sum_{j=0}^{k-1} O(j) = O(k^2)$. Hence, the total time needed to compute all A^ks is $O(n^3)$. As noted before, the amount of time needed to construct all k Hamiltonian paths, if needed, is $O(n^2)$. It follows that the overall time taken by the algorithm is $O(n^3)$.

Since each array A^k consists of k entries, the amount of space needed to store these arrays is $O(n^2)$. Similarly, an amount of $O(n^2)$ is needed to store the values of $D_{i,j}$, $0 \leqslant i < j \leqslant n-1$. The amount of space needed to compute and store the actual paths, if needed, is $O(n^2)$. It follows that the total amount of space needed by the algorithm is $O(n^2)$.

Finally, to find a Hamiltonian path π of minimum total length that visits all the points in S with no restriction on its endpoints, we only need to repeat the above procedure $n-1$ times, once for each starting point (one point need not be considered as a source). It follows that the overall time complexity to find π, an optimal Hamiltonian path that connects all points in S, is $O(n^4)$. The following theorem summarizes the main result.

Theorem 3. *Given a set S of n points on the boundary of a simple polygon, it is possible to find $n-1$ Hamil-*

tonian paths $\pi_1, \pi_2, \ldots, \pi_{n-1}$ from a source vertex to every other vertex in $O(n^3)$ *time and* $O(n^2)$ *space. Thus, a Hamiltonian path of shortest length that visits all the points in S can be computed in* $O(n^4)$ *time and* $O(n^2)$ *space.*

Proof. Direct from Lemma 2 and the analysis of time and space complexities above. □

Acknowledgements

The author is grateful to King Fahd University of Petroleum and Minerals for their support, and to the anonymous reviewers for their valuable comments.

References

[1] M.H. Alsuwaiyel and D.T. Lee, Minimal link visibility paths inside a simple polygon, *Comput. Geometry Theory Appl.* **3** (1) (1993) 1-26.

[2] S. Carlsson, H. Jonsson and B.J. Nilsson, Finding the shortest watchman route in a simple polygon, in: *Proc. 4th Ann. Internat. Symp. on Algorithms Comput. (ISAAC93)*, Lecture Notes in Computer Science (Springer, Berlin, 1993).

[3] W.-p. Chin and S. Ntafos, Optimum watchman routes, *Inform. Process. Lett.* **28** (1988) 39-44.

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