Dividing a simple polygon into two territories

THE TRANSACTIONS OF THE IECE OF JAPAN, VOL. E 69, NO. 4 APRIL 1986

LETTER

Dividing a Simple Polygon into Two Territories

Tetsuo ASANO†, *Member*

SUMMARY This paper considers the problem: Given two points u and v in a simple polygon P, divide P into three parts, locus of points closer to u, that closer to v, and that equidistant from u and v. An $O(n^2)$-time algorithm is presented where n is the number of vertices of the simple polygon.

Consider a problem of dividing a state into two parts based on the distance from two big cities in the state. A formal description of the problem is as follows:

Given two points u and v in the interior of a simple polygon P, divide the polygon P into three parts $N(u, v)$, $F(u,v)$, and $ED(u,v)$, where

$N(u,v)=\{x \in P|\text{dist}(x,u)<\text{dist}(x,v)\}$,

$F(u,v)=\{x \in P|\text{dist}(x,v)<\text{dist}(x,u)\}$, and

$ED(u,v)=\{x \in P|\text{dist}(x,u)=\text{dist}(x,v)$,

and $\text{dist}(x,y)$ is the geodesic distance between two points x and y, in other words, the length of the shortest path connecting x to y within the simple polygon P. An example is shown in Fig. 1. As is seen in the figure, the boundary lines consist of straight line segments and hyperbolic curve segments.

The key idea of the algorithm to be presented in this paper is to decompose a simple polygon P into disjoint regions so that for an arbitrary query point q the geodesic between q and u and between q and v can easily be computed. The decomposition algorithm is as follows.

[Algorithm Decomposition]

[input] A simple polygon P and a point x in its interior.

[output] Decomposition of P into disjoint regions P_0, $P_1, \cdots, P_m$. For each region P_i, r__point(P_i) and gdist (x,P_i) are computed:

r__point(P_i): A representative point of P_i that is nearest to the given point x within P_i.

gdist(x,P_i): The geodesic distance between x and the representative point of P_i.

Thus, for any point w in P_i, the geodesic distance dist(x, w) from x to w is given by the sum of gdist(x,P_i) and the length of the straight line segment between w and r__point(P_i).

Manuscript received January 16, 1986.
Manuscript revised February 24, 1986.

†The author is with the Faculty of Engineering, Osaka Electro-Communication University, Neyagawa-shi, 572 Japan.

```
begin
    vis__decom(x, 0, P)
end
procedure vis__decom(w, distance, S)
begin
    (1) Find the visibility polygon Vis(w,S) of the
        polygon S from the point w;
    (2) Enumerate all the reflexive vertices p1, p2, ...,
        pm of S on the boundary of Vis(w,S) such
        that each pi is adjacent to both a visible edge
        and an invisible edge in S;
    (3) Remove the region Vis(w,S) from S and
        then let P1, P2, ..., Pm be the resulting regions
        such that each Pi contains the vertex pi;
    (4) Let P0 be Vis(w,S) and let
          r__point(P0)=w;
          gdist(x,P0)=distance;
    (5) For each vertex pi and the polygon Pi, call
        vis__decom (pi, distance+dist(w,pi), Pi);
        where dist(w,pi) is given by the straight line
        distance between w and pi since pi is visible
        from w in S
end
```

It is easy to see that the above procedure decomposes the given simple polygon P with respect to the point x into disjoint regions such that for each region P_i

(1) there exists one representative point denoted

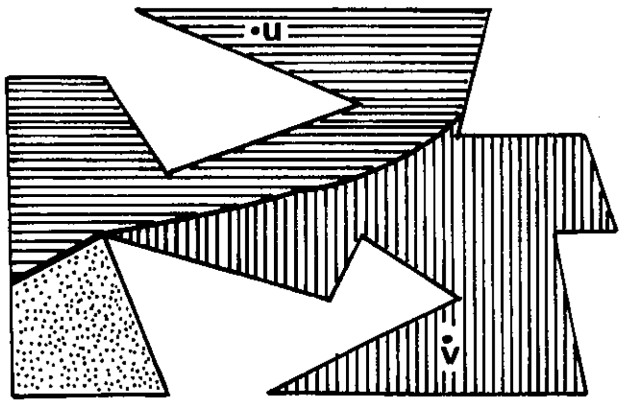

Fig. 1 Decomposition of a simple polygon into three parts $N(u, v)$, $F(u,v)$, and $ED(u,v)$, where $N(u,v)$ ($F(u,v)$, resp.) is the locus of points closer to u(v, resp.) than to v (u, resp.) and $ED(u,v)$ is that of points equidistant from u and v, which is the dotted region in the figure.

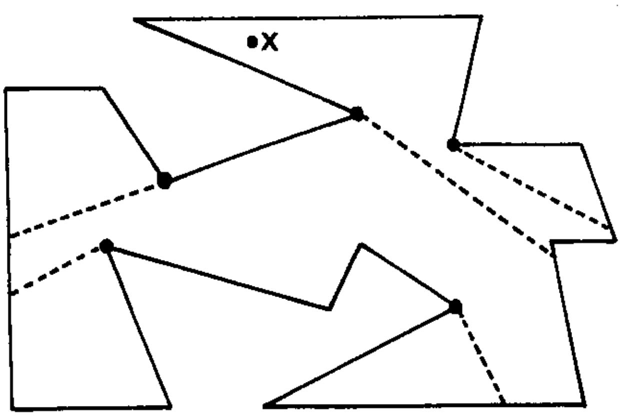

Fig. 2 Decomposition of a simple polygon based on the visibility from a point x using the procedure vis__decom.

by r__point(P_i) which is either a vertex of P or the point x,

(2) any point in P_i is visible from its representative point,

(3) the shortest path from x to any point v in P_i passes through the representative point of P_i and does not pass through any other vertex of P on the way from r__point(P_i) to v, and thus

(4) the geodesic distance between x and any point v in P_i is given by the sum of the straight line distance between v and r__point(P_i) and the geodesic distance between r__point(P_i) and x.

It is also easy to see that the above decomposition is done in $O(n^2)$ time by using at most n times a linear time algorithm[(1),(2)] for computing the visibity polygon from a point in a simple polygon.

Figure 2 shows a decomposition of a simple polygon by the above procedure.

Based on the decomposition described above, we decompose a simple polygon P into three parts with respect to two specified points u and v, the locus of points closer to u than to v, that closer to v than to u, and that equidistant from u and v. First of all, we decompose a given simple polygon P with respect to the point u and then with respect to v. Let P_0, P_1, $\cdots$, P_M and Q_0, Q_1, $\cdots$, Q_N be resulting polygons with respect to u and v, respectively, where P_0 and Q_0 are the visibility polygons from u and v, respectively. At the same time for each vertex v_i we find the region P_j and Q_k which contains v_i. Next, we compute the intersection of P_0, P_1, $\cdots$, P_M and Q_0, Q_1, $\cdots$, Q_N. This results in a finer decomposition of P. Let R_0, R_1, $\cdots$, R_K be the resulting regions, where R_0 is the intersection of P_0 and Q_0 if it is not empty.

Next, we find the boundary of the region $ED(u,v)$. We propose a brute-force algorithm as follows. The first step is to compute the distance to the two points u and v from each vertex v_i of each region defined above. In constant time we can find regions P_j and Q_k which contain the vertex v_i. Thus, constant time is enough to compute the distances from v_i to u and v. Since there are at most $O(n^2)$ such vertices, the above computation is done in $O(n^2)$ time.

Let (s,t) be an edge of some region R_i. Then, the necessary and sufficient condition that the boundary of $ED(u,v)$ crosses the edge, is that one endpoint is closer to u than to v and the other closer to v than to u. In this way we can enumerate all the edges intersecting the boundary of $ED(u,v)$. Let R_i be a region which contains more than one such edge. Assume that R_i is the intersection of P_j and Q_k. Then, for any point q in R_i we have

$$\mathrm{dist}(q,u)=\mathrm{dist}(q,r_j)+\mathrm{dist}(r_j,u),$$

and

$$\mathrm{dist}(q,v)=\mathrm{dist}(q,u_k)+\mathrm{dist}(u_k,v),$$

where $r_j=r$__point(P_j) and $u_k=r$__point(Q_k). The boundary of the region $ED(u,v)$ is characterized by a set of those points q which satisfy the equation

$$\mathrm{dist}(q,u)=\mathrm{dist}(q,v), \text{ that is,}$$

$$\mathrm{dist}(q,r_j)+\mathrm{dist}(r_j,u)$$
$$=\mathrm{dist}(q,u_k)+\mathrm{dist}(u_k,v).$$

Since dist(r_j,u) and dist(u_k,v) may be regarded as constants, the boundary in R_i is a straight line segment (more precisely, the bisecting line between the two points r_j and u_k) if dist(r_j,u)=dist(u_k,v), and a hyperbolic curve segment(s) otherwise. Especially, if r__point(P_j) coincides with r__point(Q_k) and gdist(x,P_j) is also equal to gdist(x,Q_k), then the whole region of R_i is included in $ED(u,v)$.

In this way we can compute the region $ED(u,v)$ for any pair of points u and v in the given simple polygon in $O(n^2)$ time. Each internal boundary of $ED(u,v)$ consists of at most $O(n^2)$ (staright line or curve) segments. The two parts $N(u,v)$ and $F(u,v)$ are obtained by removing the region $ED(u,v)$ from the simple polygon P.

As an application of the cecomposition algorithm described in this letter, we can devise an efficient algorithm for the following problem:

We are given a simple polygon P and a point x in its interior. Given a query point q in the interior of P, compute the geodesic distance between x and q.

Given a simple polygon P and a point x, we decompose P with respect to x using the decomposition algorithm. Then we have a planar subdivision with at most $O(n)$ edges where n is the number of vertices of P. Therefore, given a query point q in the interior of P, we can find the region P_i that contains q in its interior in $O(\log n)$ time with $O(n \log n)$-time preprocessing, using a point-location algorithm[(3),(4)]. Then, we find the representative point r__point(P_i) together with geodesic distance between r__point(P_i) and x. Since the shortest internal path from x to q passes through the point r__point(P_i) and q is visible from r__point(P_i). The

geodesic distance dist(q,x) between q and x is given by the sum of the geodesic disatnce between x and r_point(P_i), which is already obtained as dist(P_i), and the straight line distance between r_point(P_i) and q.
[Lemma] We are given a simple polygon P with n edges and point x in its interior. Given a query point q in the interior of P, we can compute the geodesic distance from q to x in $O(\log n)$ time with $O(n^2)$-time preprocessing and $O(n)$ space.

It follows from the lemma that we can solve the following problem in $O(m \log n)$ query time with $O(mn^2)$-time preprocessing.
(Problem) We are given a simple polygon P with n edges and m points $u_1, \cdots, u_m$ in the interior of P. Given a query point q in the interior of P, find a point among m given points that is closest to q.

References

(1) H. El Gindy and D. Avis: "A Linear Algorithm for Computing the Visibility Polygon from a Point", J. Algorithms, 2, 2, pp. 186-197 (1981).
(2) D. T. Lee: "Visibility of a Simple Polygon", Computer Vision, Graphics and Image Processing, 22, pp. 207-221 (1983).
(3) D. G. Kirkpatrick: "Optimal Search in Planar Subbivisions", SIAM J. Comput., 12, pp. 28-35 (1983).
(4) R. J. Lipton and R. E. Tarjan: "Applications of Planar Separator Theorem", Proc. 18th IEEE Symp. on Foundations of Computer Science, Providence, pp. 162-170, Rhode Island, (1977).