International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:6, No:4, 2012

An approach to the solving Non-Steiner minimum link path problem

V. Tereshchenko and A. Tregubenko

Abstract—In this study we survey the method for fast finding a minimum link path between two arbitrary points within a simple polygon, which can pass only through the vertices, with preprocessing.

Keywords—minimum link path, simple polygon, Steiner points, optimal algorithm.

I. Introduction

THE *relevance of the problem.* Nowadays there are a lot of problems, which are related to the shortest paths finding. Algorithms for finding optimal paths, which use the Euclidean metric, are applied almost in all computer games. Link-distance metric is used in robotics, when rotation is much more expensive compared with motion itself. Link-distance metric without using of Steiner points is used in problems, which are related to transmitting straight wireless signals, where it is necessary to have opportunity to define quickly, how to transmit a signal from one point of area to another using the least number of transmitters.

Analysis of the latest researches. For objectives of link-distance metric in a simple polygon a concept of window turns out to be very useful [1]. This structure is explained better by using the idea of visibility. All points that are directly visible from the point s have distance 1. We call this set V_1. The boundaries between visible points and invisible points we denote as a window. Points from $P \backslash V_1$, which are visible from any point from the window V_1, have distance 2. Repetition of this construction will give all windows partition of polygon. For a fixed point s data structure "windows tree" [1] makes it possible to do search using $O(\log N)$ time and $O(N)$ of memory. When both of points are set by request, was proposed data structure by Arkin, Mitchell and Suri [2] that allows to do search for $O(\log N)$, but requires $O\left(N^3\right)$ time and $O\left(N^3\right)$ of memory for preprocessing. In a paper of W. McAveney it is proposed the algorithm that operates for $O\left(E(n)\alpha(n)\log^2 n\right)$ [3].

N*Novelty and idea.* Unfortunately, a prohibition to the use Steiner points don't allow to use windows partition to find the shortest link path. In this paper we consider a more complicated partition of polygon into visible regions, the point localization in these regions and fast finding of the shortest path between the regions.

V. Tereshchenko and A. Tregubenko are with the Faculty of Cybernetics, National Taras Shevchenko University of Kyiv, Kyiv, Ukraine, 02095 e-mail: vtereshch@gmail.com, anton.tregubenko@gmail.com

Manuscript received February 20, 2012; revised February 29, 2012.

II. Constructing the algorithm

Problem. Given a simple polygon P and a pair of points $s, t \in P$. Find minimal link path within P from s to t without using Steiner points, Figure 1.

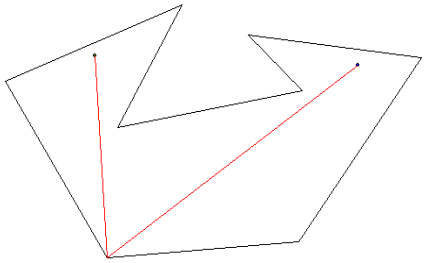

Fig. 1. Minimal link path without using Steiner points

A. Search without preprocessing

The problem of finding the minimum link path without using Steiner points is reduced to the problem of searching in the visibility graph of a polygon. To do this, at first you need to check points on the direct visibility for $O(N)$, reconsidering the intersection of link segment with polygon sides. If it is no intersection, the shortest path is 1. Otherwise, you need to build a visibility graph for vertices ($O(E)$, [4]), find the visible vertices from the points s and t ($O(N)$, [5]), find the tree of minimum paths from the vertex s by searching in width ($O(E)$, [6]) and reviewing all visible vertices of the polygon from t find that one, the distance from s to which is the smallest.

In general, the algorithm takes $O(E)$ time and memory.

B. Search with preprocessing (algorithm of visible regions)

For fast request realization of the shortest path it is necessary to include a construction of the tree of minimal paths to the preprocessing. To do this, we divide a polygon into the visibility regions, Figure 2.

*Definition.*A visibility region of a polygon is a such its part, that for all points from this part the set of visible vertices will be the same.

Preprocessing algorithm. At first we build visibility graph for vertices ($O(E)$, [4]). Then we build a convex hull of our polygon according the Lee algorithm and register it as a knot of the regions tree ($O(N)$, [7]). Find a list of all windows that form all vertices of the polygon ($O\left(N^2\right)$, [1]). Add to

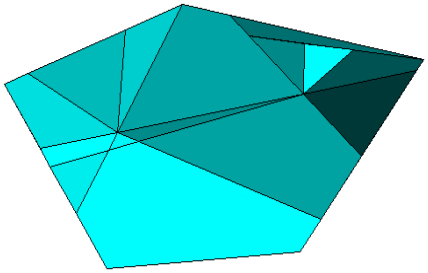

Fig. 2. Visibility regions in the convex hull

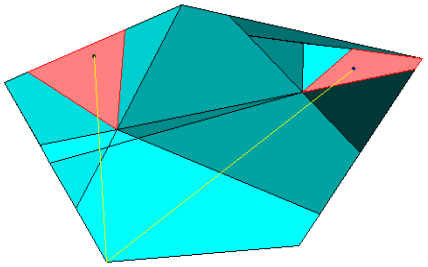

Fig. 3. Localization of points and finding the shortest path.

this list the initial polygon sides. All segments in the list will be stored in such a way that from the left side of the segment will be visible part, and from the right side - invisible. Let S be a number of segments.

Then randomly we will add these vertices to the regions tree in the following way: insert the segment into the knot of the regions tree. If a vertex is a leaf node, then we cut the segment into 2 regions and register formed parts into the proper successors of this vertex, and segment – as cut off line of this vertex. Otherwise, we check the position of the segment comparatively the cut off line in the vertex. If there isn't any intersection, we continue to search in that subtree, in which is the segment. If there is intersection, we divide a segment into 2 parts and insert these parts in the appropriate subtrees. All segments that overlap, we throw out. Let R be the total number of obtained regions.

Further, for each of obtained visibility regions we find the center and visible points from this center ($O(N)$, [5]). On the basis of this we build the tree of the shortest paths using search in width ($O(E)$, [6]).

Search algorithm. We need to check points on visibility. If there is no direct visibility - start your search.Locate points in the tree ($O(\log R)$). To do this we check the knot of the tree, from which side of the cut off line a point is situated and continue to search in the proper subtree, until you reach the leaf node. If the point lies on the cut off line, then accordingly to our construction, left side for a segment, which formed a cut off line, will be visible, so we continue search in the left subtree. After doing the localization of points we find that visible vertex from the visibility points of the second region to which the shortest path from the first region ($O(N)$), Figure 3. Accordingly (this length + 2) will be minimal link length, and the path can be derived for $O(W)$, where W- its length.

To achieve the request complexity of $O(\log N)$ during a preprocessing we find the map of minimal paths between all visibility regions ($O\left(R^2N\right)$ time, $O\left(R^2\right)$ memory). Then during the search it is necessary only to locate the points ($O(\log R)$). Then a derivation of the path will be made for $O(W)$, where W - its length.

III. Justification of correctness and complexity

Lemma 1. *Let S is a number of segments and R is the total number of obtained regions, then following statements take place:*

1)*All regions are convex.*
2)$S < N(N-1)$.
3)$R \leq \frac{(S+1)S}{2} + 1$

Proof: .

1). Initially a convex hull of a polygon is in the tree, and so a convex set. By induction we have that at each step we will cut a convex set with a line, and so we get convex sets too.

2).The window is assigned by two vertices of a polygon.

3). At first there is one region. $(i+1)$- cut off segment, which we add, cannot intersect more than $i$segments (lines), that have been already added, thus it forms no more than $i+1$ new regions. So $R \leq 1 + \sum_{i=1}^{S} i \leq \frac{(S+1)S}{2} + 1$.

■

Lemma 2. *The same set of vertices can be seen from all points of visibility region.*

Proof: Let suppose the contrary and then there are two points, one of which sees the vertex V, and another doesn't see. But then between them must be either a window, or a polygon side, that would have divided region into two parts.

■

An average estimation. For random polygons $R \sim N^2$.

The total complexity of the algorithm. The algorithm requires $O\left(RN^2\right)$ time and $O(RN)$ memory for preprocessing and $O(\log R + N)$ time on a request. Complicated algorithm requires $O\left(R^2N\right)$ time and $O\left(R^2\right)$ memory for preprocessing and $O(\log R)$ time on a request. Withdrawal of the minimum path is executed for $O(W)$, where W- its length.

IV. Implementation

The program is designed in Delphi 7 and does not require an installation of any additional modules, figure 4.

*Input.*Polygon inputting is implemented either manually (mouse), or is loaded from a file. Request points are entered manually after pretreatment of a polygon.

Output. On the screen are displayed a polygon itself (black), path (orange) and the numbers N, S, R and W. The program also allows you to display additional information such as the partition on the regions (blue-green), the visibility of request points (black) and localization of points in regions (red).

Main functions. Generation of partition on the regions, regions insertion in the tree and a point localization in the tree.

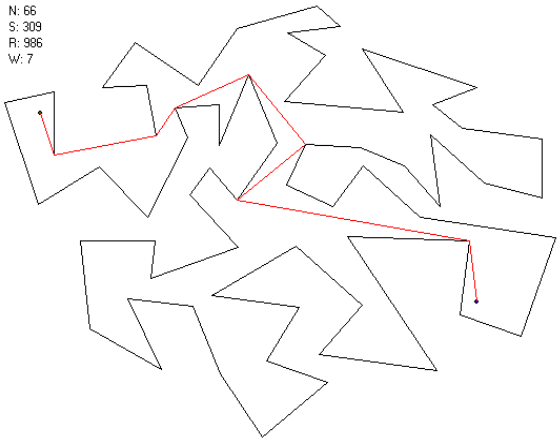

Fig. 4. Example of the algorithm realization.

V. Conclusions

In this paper we consider algorithms for finding the minimum link path in a polygon without using Steiner points. Forbidding to use additional points reduce problem to the search in visibility graph, for which there is no faster algorithms than common for unweighted graphs [8].

In the program implementation not all functions are made for the optimal time, because it does not affect the overall complexity of the algorithm, and is not the purpose of this work. As an improvement of a preprocessing algorithm we can suggest an ordered construction of trees of minimal paths for visibility regions. It is necessary to build a planar graph of regions connections where two regions are connected if they have a common edge. Then we organize regions by the number of visible vertices and we build trees in order in a following way. If there is no neighbors with defined tree of minimal paths in the region, we define a tree in the usual way. If there is a neighboring region, and a common edge does not change the set of visible vertices (i.e. not a window, but an extension of it), we record a tree of minimal paths that has been already found for neighbor. If it changes the set of visible vertices, it means in this region can be seen on a one vertex of polygon more (because we order by the number of visible vertices, so in this will be more). Updating of tree for a neighboring region by adding a new vertex can be done in linear time by simply watching all the paths and optimizing them if the path from added vertex is shorter (with a map of minimum path for vertices it can be done with constant time). Thus time complexity of preprocessing in a simple algorithm will be improved from $O\left(RN^2\right)$ to $O\left(RN\right)$, and in the complicated algorithm from $O\left(R^2N\right)$ to $O\left(R^2\right)$. As an improvement of a searching algorithm remains an open problem of construction of balanced tree of visibility regions without increasing the number of these regions.

References

[1] S. Suri. *On some link distance problems in a simple polygon.* IEEE Trans. Robot. Autom.,6, 1990, p:108-113.

[2] E.M. Arkin, J.S.B. Mitchell, and S.Suri. *Optimal link path queries in a simple polygon.* In Proc.3rd ACM-SIAM Sympos. Discrete Algorithms, 1992, p:269-279.

[3] W. McAveney. *Minimum link path finding methods in 2D polygons.* CS 633 Fall 2007.

[4] J. Hershberger. *An optimal visibility graph algorithm for triangulated simple polygons.* Algorithmica, 4,1989 p:141-155.

[5] B. Joe and R.B. Simpson. *Correction to Lee's visibility polygon algorithm.* BIT, 27, 1987, p:458-473.

[6] A Goldberg. *P-to-P shortest path algorithms with preprocessing.* Sofsem, 2007.

[7] D.T. Lee. *On finding the convex hull of a simple polygon.* IJCIS, Vol.12, No. 2, 1983, pages 87-98.

[8] M. Nouri, M.Ghodsi. *Space-query-time tradeoff for computing the visibility polygon.* LNCS, Vol.5598, 2009, pages 120-131.

Vasyl Tereshchenko In 1986 graduated from the Mathematics and Mechanics Faculty of Kyiv National Taras Shevchenko University. In 1992 graduated from graduate school and in 1993 defended PhD dissertation on the degree C.Sci. (Phys-Math.). In 2011 I defended a dissertation for the degree a doctor of Phys.-Math. sciences (theoretical bases of computer science and cybernetics).

Since 1994 - Associate Professor Faculty of Cybernetics KNTSU. Lecturer in computer graphics and in computational geometry, and also in databases and in the theory of algorithms.