The Shortest Path of Touring Lines given in the Plane

Lijuan Wang[1,2], Dandan He[2,*], Ansheng Deng[1] and Tao Ning[3]

[1]*College of Information Science and Technology, Dalian Maritime University, Dalian, Liaoning, 116026, P.R. China;*
[2]*Dept of Information and Science, Dalian Institute of Science and Technology, Dalian, Liaoning, 116052, P.R. China;*
[3]*Software College, Dalian Jiaotong University, Dalian, Liaoning, 116045, P.R. China*

Abstract: Given two points p, q and a sequence of n lines (n>1) in the plane, we want to find the shortest path of touring all the given lines that starts at p and ends at q. In this paper, we solve the problem by reducing it to the problem of finding the shortest path that tours all the segments in a convex polygon from p to q. We first introduce how to construct the convex polygon. Then, we propose the solution to process the intersections of two adjacent segments by crossing over two segments. Finally, based on rubber-band algorithm, a new algorithm is proposed which can find the shortest path of touring segments in a convex polygon, and the O(n^2) running time can be obtained for this problem, which improves the previous O(n^3logn) time. Our algorithm is simple and efficient.

Keywords: Convex polygon, rubber-band algorithm, shortest path.

1. INTRODUCTION

Path planning is one of the central problem areas in computational geometry. The shortest touring path problem is the most classical example of path planning, for example, finding the shortest path of touring a given series of obstacles between two points in the plane [1]. In this paper, we focus on the solution to find the shortest path of touring given lines in the plane from p to q. The problem is as follows.

Assume that there are two points p, q and n>1 line l_i (for i=1,2,… , n) in a plane Π, we want to find the shortest path that tours all the given lines l_i from the start point p to the end point q, see Fig. (**1**).

The problem given above is abstracted from the shortest watchman route problem, and it is also the key to solve the shortest watchman route problem [2, 3]. The watchman route problem is derived from the well-known art gallery problem, and the goal is to find a shortest closed route in a simple polygon G such that one can see each point in the polygon G from at least one point of the route.

Hakan Jonsson has presented an algorithm for finding the shortest path that intersects n lines in the plane in O(n^5) time [4]. No more optimal results have been reported recently. Many scholars have been studying on the touring segments problems and made some effective results. For example, for the case in which the given segments don't intersect, in 1984, D. T. Lee and F. P. Preparata proposed Funnel algorithm [5]. In 2007, Fajie Li and Reinhard Kettle proposed rubber-band algorithm [6, 7] (denoted by R algorithm) with K(ε)·O(n) time, where K(ε)=(L_0-L) /ε, L_0 is the initial shortest path length of touring all segment set S, L is the actual shortest path, and n is the number of segments. In 2011, R algorithm is obtained in O(n^2) time by Wang and Huo by the experiment. Furthermore, they improved the algorithm and obtained the O(n) time by introducing divide and conquer into R algorithm [8]. For the case in which the given segments possibly intersecting, the O(n^3logn) time algorithm for touring a sequence of possibly intersecting line segments is known [9].

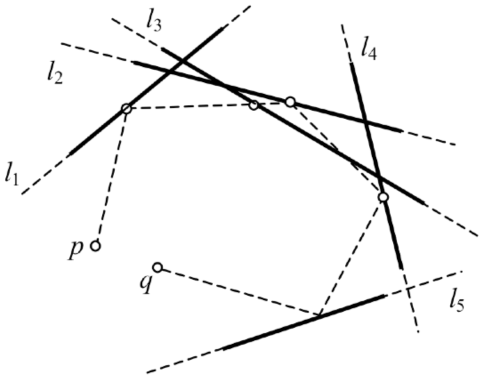


Fig. (1). The shortest path of touring 5 lines in the plane.

In our paper, the problem of finding the shortest path for lines can be converted to the problem of finding the shortest path for segments in a convex polygon. We first introduce how to construct the convex polygon. Then, on basis of R algorithm, a new algorithm is proposed which processes the degradation problem caused by the intersection by crossing over two segments when two adjacent segments intersect. Finally, we have implemented it with C++ program and obtained the O(n^2) time by experiment. The result shows that the algorithm is efficient.

1874-110X/15 2015 Bentham Open

2. THE METHOD OF CONSTRUCTING CONVEX POLYGON

We use OPT(L) to denote the shortest path of touring given line set L from p to q, since OPT(L) is the shortest path, the parts between consecutive points of tour L are line segments. Moreover, since a non-convex route in the plane is strictly longer than the boundary of its convex hull, and the convex hull of a route intersects any target line, we conclude that OPT(L) is convex [4]. It implies that OPT(L) visits each target line in one of three ways. If the angle of OPT(L) coming into a line l_i with l_i (incoming angle) is equal to the angle of OPT(L) going away from a line l_i with l_i (outgoing angle), OPT(L) makes a perfect reflection on the line l_i (see Fig. **2a**). If OPT(L) passes the line l_i twice, OPT(L) makes a crossing contact with l_i (see Fig. **2b**). The case in which OPT(L) shares a portion of lines with l_i can be thought of a special reflection (see Fig. **2c**).

Here, we respectively take the starting point p and the ending point q as the intersection of two lines, thus the number of lines in set L is n+4. We bound the region in the plane where OPT(L) visits the set L, and consider the convex hull C of all crossings(see Fig. **3a**). For sets L containing no parallel target lines, it can be shown that OPT(L) lies in C, and the point p and q lies in C or on the boundary of C. However, if there are parallel lines in set L, OPT(L) may be not contained in the convex hull C (see Fig. **4**). Since C is convex, each of its tangents partitions the plane into two half-planes of which the active half-plane totally contains C. We make all tangents of C that are normal to target lines, denoted by the set N, and consider the intersection G of the active half-planes of all tangents in set N (see Fig. **3b**), we can conclude OPT(L) is contained in G [4]. Thus, we can reduce this problem to compute the shortest path between p and q which visits its segments in the convex polygon G.

3. COMPUTE THE SHORTEST PATH OF SEGMENTS IN A CONVEX POLYGON

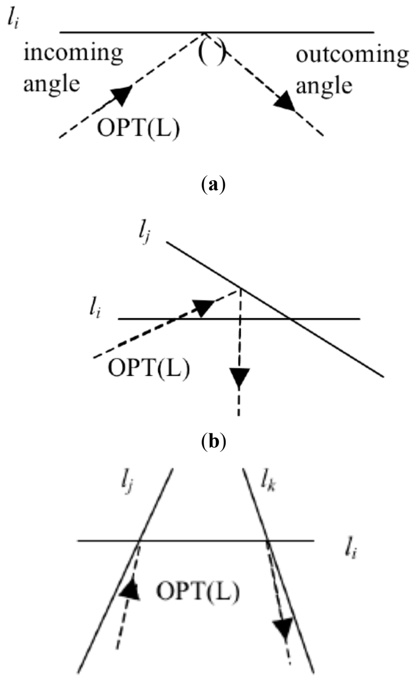
For the case in which the segments in a convex polygon are disjoint, the solution has been described in detail in references [6-8]. For the case in which the segments possibly intersect, a new algorithm has been introduced as follows.

3.1. The Method of Dealing with the Intersected Segments

When the segments intersect, suppose that the segment s_i intersects with s_{i+1} at the point C, and q_{i-1} is the path point on s_{i-1} and q_{i+2} is the path point on s_{i+2}. There are three cases between $\overline{q_{i-1}q_{i+2}}$ and the segments s_i and s_{i+1}, which are as follows.

Case 1 $\overline{q_{i-1}q_{i+2}}$ intersects with s_i and s_{i+1}

In this case, obviously, $\overline{q_{i-1}q_{i+2}}$ is the local shortest path, denoted by d, $d=\overline{q_{i-1}q_{i+2}}$, and the order of d passing through s_i and s_{i+1} is the touring order, if d first passes through s_i, the touring order is s_i and s_{i+1}, otherwise, that is s_{i+1} and s_i, see Fig. (**5**).

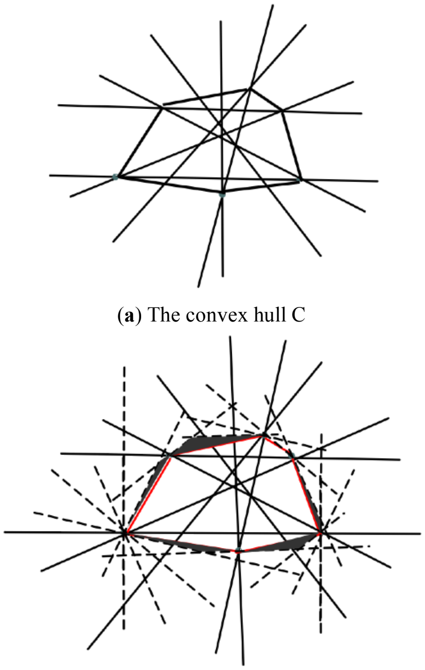


(**a**)

(**b**)

(**c**)

Fig. (2). Three types of contacts between OPT(L) and a line l_i.



(**a**) The convex hull C

(**b**) The polygon G (shaded and containing C)

Fig. (3). Constructing convex polygon.

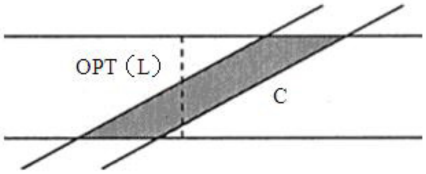
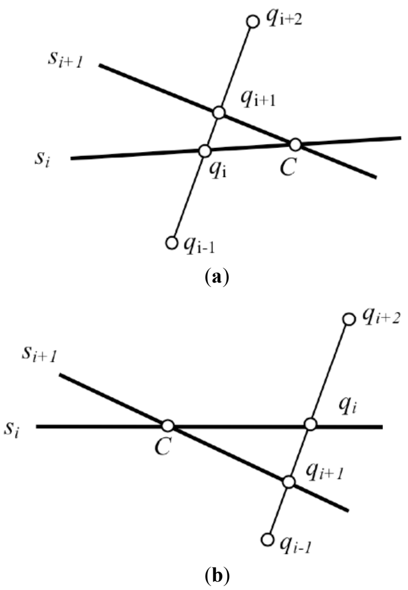


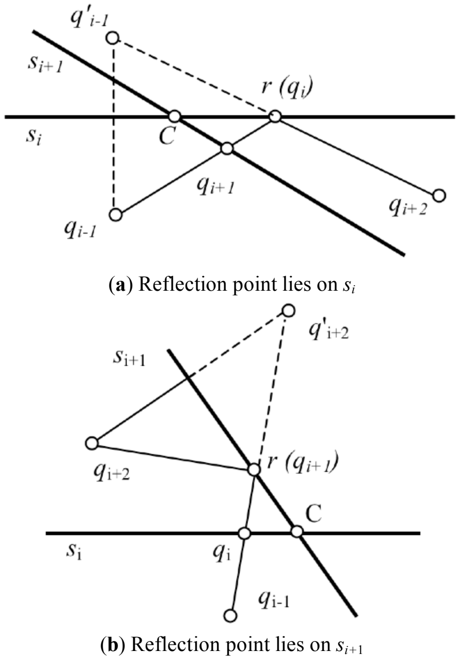
Fig. (4). OPT(L) is not contained in C.



(**a**)

(**b**)

Fig. (5). $\overline{q_{i-1}q_{i+2}}$ intersects with s_i and s_{i+1}.



(**a**) Reflection point lies on s_i

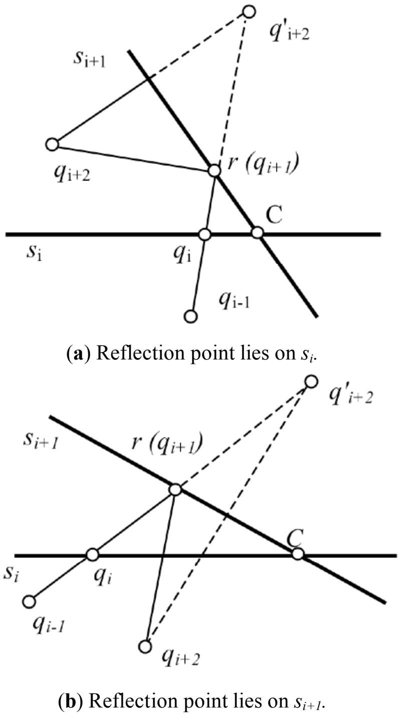
(**b**) Reflection point lies on s_{i+1}

Fig. (6). $\overline{q_{i-1}q_{i+2}}$ intersects with s_i or s_{i+1}.

Case 2 $\overline{q_{i-1}q_{i+2}}$ intersects with s_i or s_{i+1}

In this case, to find the shortest path, computing the reflection point r on the line segment s_i or s_{i+1} is needed (see Fig. **6**).

The approach in the iteration process is as follows. If $\overline{q_{i-1}q_{i+2}}$ only intersects with s_i or s_{i+1}, the reflection point r lies on the segment which does not intersect with $\overline{q_{i-1}q_{i+2}}$, $d = \overline{rq_{i-1}} + \overline{rq_{i+2}}$, and the touring order is the order of d passing through s_i and s_{i+1}, if d first passes through s_i, the touring order is s_i and s_{i+1}, otherwise, that is s_{i+1} and s_i.



(**a**) Reflection point lies on s_i.

(**b**) Reflection point lies on s_{i+1}.

Fig. (7). $\overline{q_{i-1}q_{i+2}}$ doesn't intersect with s_i and s_{i+1}.

Case 3 $\overline{q_{i-1}q_{i+2}}$ doesn't intersect with s_i and s_{i+1}

In this case, there are two cases which are shown in Figs. (**7**) and (**8**).

(1) the case of computing one reflection

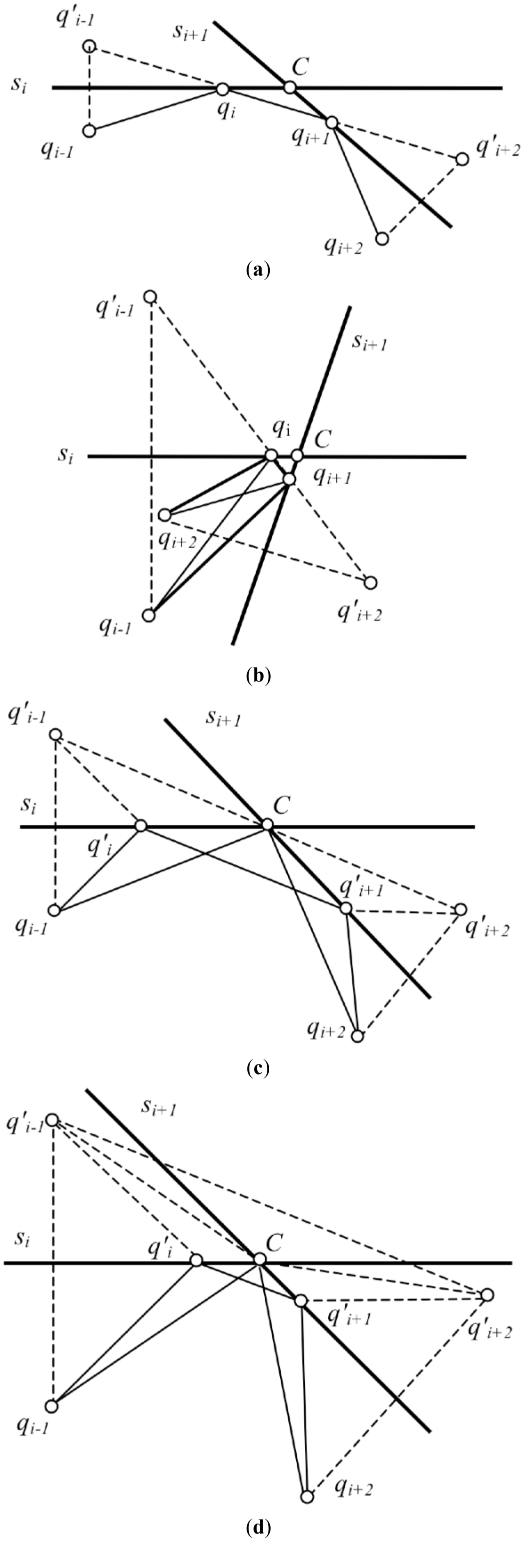
The two segments can be visited by computing a reflection point (see Fig. **7**). In this case, obviously, $d = \overline{rq_{i-1}} + \overline{rq_{i+2}}$, here, when d passes through s_i or s_{i+1} twice, we can deal with it as above, if d first passes through s_i, the touring order is s_i and s_{i+1}, otherwise, that is s_{i+1} and s_i.

(2) the case of computing two reflections

When the two segments can't be toured by computing a reflection point, it needs to compute two reflections. The approach is as follows. We make the reflection points, denoted by q'_{i-1} and q'_{i+2}, of q_{i-1} and q_{i+2} with respect to s_i and s_{i+1} respectively.

In this case, there are also three situations when computing the local shortest path, see Fig. (**8**).

(1) if $\overline{q'_{i-1}q'_{i+2}}$ intersects with s_i at the point q_i and intersects with s_{i+1} at the point q_{i+1}, and the two path points q_i and q_{i+1} are on the same side of q_{i-1} and q_{i+2}, they are the two reflections and also the path points. If $\overline{q_{i-1}q_i}$ does not intersect with $\overline{q_{i+1}q_{i+2}}$, $d = \overline{q_{i-1}q_i} + \overline{q_iq_{i+1}} + \overline{q_{i+1}q_{i+2}}$ is the local shortest path, and the order of d passing through s_i and s_{i+1} is the touring order (see Fig. **8a**). Otherwise, we need to change the touring order of s_i and s_{i+1} in order to get the local shortest path, then $d = \overline{q_{i-1}q_{i+1}} + \overline{q_{i+1}q_i} + \overline{q_iq_{i+2}}$, see Fig. (**8b**).



(**a**)

(**b**)

(**c**)

(**d**)

Fig. (8). q_{i-1} and q_{i+2} lie on the same side of s_i and s_{i+1} (two reflection points).

(2) if $\overline{q'_{i-1}\,q'_{i+2}}$ passes the point C, the reflection points coincide with C and they are also the path points(q_i=q_{i+1}=C), $d = \overline{q_{i-1}C} + \overline{Cq_{i+2}}$, and the order of d passing through s_i and s_{i+1} is the touring order. We can make two auxiliary points q'_i and q'_{i+1} on s_i and s_{i+1} and prove that easily, see Fig. (**8c**).

(3) if $\overline{q'_{i-1}\,q'_{i+2}}$ doesn't pass the point C and $\overline{q'_{i-1}\,q'_{i+2}}$ is on the different side of $\overline{q_{i-1}C}$ and $\overline{Cq_{i+2}}$, $d = \overline{q_{i-1}C} + \overline{Cq_{i+2}}$, the path points q_i and q_{i+1} also coincide with C, and the order of d passing through s_i and s_{i+1} is the touring order, see Fig. **8d**). It can be proved as follows. In Fig. (**8d**), we select a point q'_i on segment s_i and a point q'_{i+1} on s_{i+1}, since $\overline{q'_{i-1}\,q'_i} + \overline{q'_i\,q'_{i+1}} + \overline{q'_{i+1}\,q'_{i+2}} = \overline{q_{i-1}q'_i} + \overline{q'_i\,q'_{i+1}} + \overline{q'_{i+1}\,q_{i+2}}$, $\overline{Cq'_{i-1}} + \overline{Cq'_{i+2}} = \overline{Cq_{i-1}} + \overline{Cq_{i+2}}$, obviously, $\overline{q'_{i+2}\,q'_{i+1}} + \overline{q'_{i+1}\,q'_i} + \overline{q'_i\,q'_{i-1}} > \overline{Cq'_{i+2}} + \overline{Cq'_{i-1}}$, then $\overline{q_{i+2}q'_{i+1}} + \overline{q'_{i+1}\,q'_i} + \overline{q'_i\,q_{i-1}} > \overline{Cq_{i+2}} + \overline{Cq_{i-1}}$, $d = \overline{Cq_{i+2}} + \overline{Cq_{i-1}}$ follows.

3.2. Algorithm

In this paper, on basis of R algorithm, we apply a new algorithm to compute the shortest touring path, with crossing over two segments to process the intersection of them when these two segments intersect. The algorithm is as follows. Assume that s_i interests with s_{i+1}, we can compute the path point q_i on s_i and the path point q_{i+1} on s_{i+1} according to the position relation between points and segments descried above. Next, we judge the position relationship of s_{i+1} and s_{i+2}, if $s_{i+1} \cap s_{i+2} = \varnothing$, we compute the path point q_{i+2} on s_{i+2} and q_{i+3} on s_{i+3}. Otherwise, we compute the path point q_{i+1} on s_{i+1} and q_{i+2} on s_{i+2}.

The algorithm is composed of four functions which are Main function, UpdatePoint function, SkipOne function and SkipTwo function. Main function calculates the shortest path by calculating all the iteration process. UpdatePoint function calculates the shortest path in every iteration. SkipOne function calculates the shortest path with crossing over one segment. SkipTwo function calculates the shortest path with crossing over two segments. SkipTwo function is the key to the algorithm. In reference [8], Main function, UpdatePoint function, SkipOne function have been described in detail. Here, we mainly give the SkipTwo function, it is described below.

SkipTwo ($q_{i-1}, s_i, s_{i+1}, q_{i+2}$) function.

In this function, s_i and s_{i+1} denote segments, q_{i-1} and q_{i+2} denote path points calculated on segments s_{i-1} and s_{i+2} respectively. The function is used to calculate the local optimal path of touring s_i and s_{i+1} from q_{i-1} to q_{i+2}, and the path points q_i and q_{i+1} can be obtained.

Denote by $r = \overline{q_{i-1}q_{i+2}}$, L the path calculated by SkipOne function, and s_i intersected with s_{i+1} at q.

Case 1 $q_{i-1}q_{i+2}$ intersects with s_i and s_{i+1}

If ($r \cap s_i \neq \varnothing$ && $r \cap s_{i+1} \neq \varnothing$), we consider that s_i intersects with r at c_1 and s_{i+1} intersects with r at c_2 respectively, let $q_i = c_1$ and $q_{i+1} = c_2$, and if ($\left|\overline{q_{i-1}r_2}\right| < \left|\overline{q_{i-1}r_1}\right|$), call change (s_i, s_{i+1}) and call change (q_i, q_{i+1}).

Case 2 $\overline{q_{i-1}q_{i+2}}$ intersects with s_i or s_{i+1}

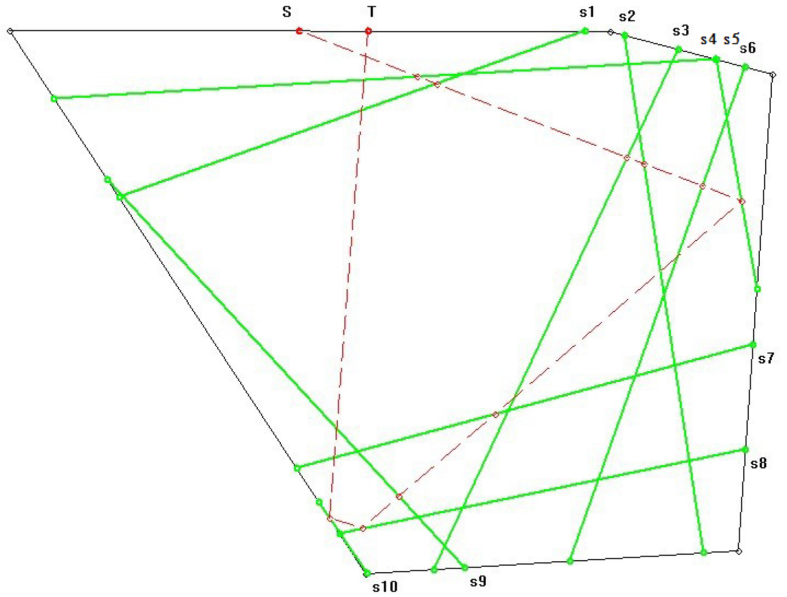


Fig. (9). The running result of 10 segments in a convex polygon.

if ($r \cap s_i \neq \varnothing \parallel r \cap s_{i+1} \neq \varnothing$)

if ($r \cap s_i \neq \varnothing$), call SkipOne ($q_{i-1}, s_{i+1}, q_{i+2}$), and we consider that s_i intersects with L at c_1 and s_{i+1} intersects with L at c_2 respectively.

Let $q_i = c_1, q_{i+1} = c_2$ and $rTmp = \overline{c_2 q_{i+2}}$,

if ($rTmp \cap s_i \neq \varnothing$), call chang (s_i, s_{i+1}),

and call change (q_i, q_{i+1}).

else call SkipOne (q_{i-1}, s_i, q_{i+2}),we consider that s_i intersects with L at c_1 and s_{i+1} intersects with L at c_2 respectively.

Let $q_i = c_1, q_{i+1} = c_2$ and $rTmp = \overline{c_1 q_{i-1}}$.

if ($rTmp \cap s_{i+1} \neq \varnothing$) call change (s_i, s_{i+1}), and call change (q_i, q_{i+1}).

Case 3 $\overline{q_{i-1} q_{i+2}}$ doesn't intersect with s_i and s_{i+1}

Call SkipOne (q_{i-1}, s_i, q_{i+2}). Denote by L intersecting with s_i at c_1 and $rTmp = \overline{c_1 q_{i-1}}$.

if ($rTmp \cap s_{i+1} \neq \varnothing$), and the intersection is c_2, then let $q_i = c_1, q_{i+1} = c_2$, call change (s_i, s_{i+1}) and call change (q_i, q_{i+1}).

else call SkipOne ($q_{i-1}, s_{i+1}, q_{i+2}$). Denote by L intersecting with s_{i+1} at c_2 , and $rTmp = \overline{r_2 q_{i+2}}$.

if ($rTmp \cap s_i \neq \varnothing$), and $rTmp$ intersects with s_i at c_1, else we compute the reflection points, denoted by q'_{i-1} *and* q'_{i+2}, of q_{i-1} and q_{i+2} with respect to s_i and s_{i+1} respectively, let $rTmp = \overline{q'_{i-1} q'_{i+2}}$, and denote s_i intersecting with $rTmp$ at r_1 and s_{i+1} intersecting with $rTmp$ at r_2 .

if (q and q_{i-1} lie on the same side of $rTmp$), let $q_{i+1} = q_i = q$.

else let $q_i = c_1$ and $q_{i+1} = c_2$.

if ($\overline{r_1 q_{i-1}} \cap \overline{q_{i+1} q_{i+2}} \neq \varnothing$), call change (s_i, s_{i+1}) and call change (q_i, q_{i+1}).

4. THE ANALYSIS OF RUNNING RESULT

In this paper, we have implemented the algorithm with C++ program. Here, we give an example of touring 10 segments in a convex polygon. In Fig. (**9**), the solid lines are the segments which the lines intersect with the convex polygon, and the intersections are the endpoints of segments. The path shown in dotted lines is the shortest path of touring the 10 lines from S to T.

We can see that the order of segments is from s_1 to s_{10}, while the output order is different from that of input. Moreover, the three cases of path points and segments described above have been contained. q_1, s_3, s_2, q_6 is the case 1, q_2, s_6, s_5, q_7 is the case 2, and q_7, s_8, s_{10}, T is the case 3.

The running time of the algorithm is complex and difficult to analyze, so in this paper, we adopt the method of analysis by running time counting to analyze the algorithm's time complexity. We have tested the algorithm applying a large of segment sets (for example 500), and the segments number is 1000 in every segment set. Based on the results, we have got the fitting curve equation of the algorithm, that is y= 1E-05x^2−0.011x + 3.810, see Fig. (**10**). The results show that the algorithm is in O(n^2) time, and it is superior to

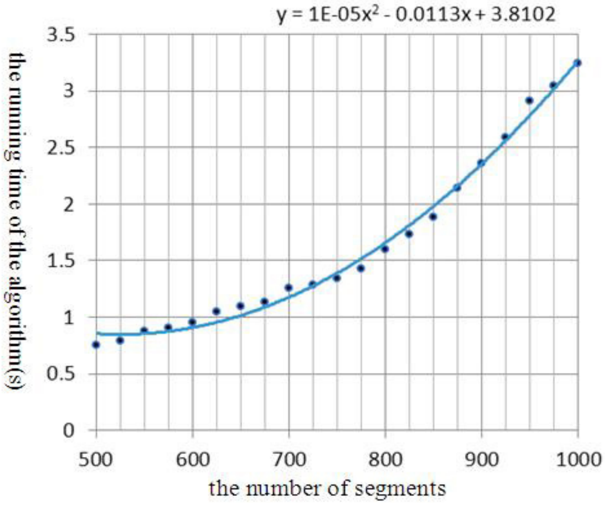


Fig. (10). The fitting curve equation of the algorithm.

the O(n^{3}logn) time of reference [9]. It can be seen that our algorithm is efficient.

CONCLUSION

In this paper, we convert the problem of computing the shortest touring path for lines to the problem of computing the shortest touring path for segments in a convex polygon, and we present an algorithm for computing the shortest touring path of segments in a convex polygon, especially, we have implemented it. The result shows that our algorithm is efficient.

This research has made preliminary results. If the segments haven't been given the order, it has been proved to be a NP hard problem, if we study the shortest path of touring rays, which is open and lies in between the problems concerning lines and segments. Those problems need to be further studied.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

This work is supported by the National Natural Science Foundation of China (No. 61173034, 61272171), the General Project of Liaoning Province Education Department (No. L2014183) and the General Project of Dalian Institute of Science and Technology (No. KYZ1416).

REFERENCES

[1] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational Geometry: Algoriths and Applications*, (3rd ed.), Springer-Verlag Publishers: Berlin, 2009, pp. 10-11.

[2] X. Tan, "Fast computation of shortest watchman routes in simple polygons", *Information Processing Letters*, vol. 77, no. 1, pp. 27-33, 2001.

[3] A. Dumitrescu, J. S. B. Mitchell, and P.. Żyliński, "Watchman route for line and segments", *Algorithm Theory - SWAT 2012 Lecture Notes in Computer Science*, vol. 7357, pp. 36-47, 2012.

[4] H. Jonsson, "The traveling salesman problem for lines in the plane", *Information Processing Letters*, vol. 82, pp. 137-142, 2002.

[5] D. T. Lee, and F. P. Preparata, "Euclidean shortest paths in the presence of rectilinear barriers", *Networks*, vol. 14, no. 3, pp. 393-410, 1984.

[6] F. Li, and R. Klette, "Exact and approximate algorithms for the calculation of shortest paths", *IMA Journal of Management Mathematics*, vol. 17, no. 1, pp. 134-138, 2006.

[7] L. Fa-jie, and K. Reinhard, "Shortest path algorithms for sequences of polygons", *CAAI Transactions on Intelligent Systems*, vol. 3, no. 1, pp. 23-30, 2008.

[8] L. Wang, L. Huo, and D. He, "An improved algorithm for euclidean shortest paths of visiting line segments in the plane", *Journal of Convergence Information Technology*, vol. 6, no. 6, pp. 119-125, 2011.

[9] M. Dror, A. Efrat, A. Lubiw and J.S.B. Mitchell, "Touring a sequence of polygons", In: *Proceedings 35th Annual ACM Symposium Theory Computation*, California, USA, 2003, pp. 473-482.

Received: September 16, 2014 Revised: December 23, 2014 Accepted: December 31, 2014