A publication of the

United Kingdom Simulation Society

ISSN 1473-8031

Volume 17, Number 23, 2016

International Journal of Simulation Systems, Science & Technology

Editor-in-chief:

David Al-Dabass

International Journal of Simulation Systems, Science & Technology **Volume 17, Number 23, 2016**

Contents

An optimal algorithm for visiting disjoint segments

Lijuan Wang[1,2], Bo Jiang[1,*], Ansheng Deng[1]

1*College of Information Science and Technology*
Dalian Maritime University
Dalian, Liaoning, China
2*Dept of Information and Science*
Dalian Institute of Science and Technology
Dalian, Liaoning, China

***Abstract*—The shortest path of visiting disjoint segments in the plane will be discussed in this paper. Given a start point p, a target point q, and a sequence of n disjoint segments in the plane, how to find the shortest path between p and q which visits each segment in the given order is our focus. In this paper, based on the rubber-band algorithm, we present an improved algorithm with the O(n(log2n) 2) running time which improves upon the O(n2) time of rubber-band algorithm, where n is the total number of all segments. We adopt the convex chain division, binary-search tree storage and merging method to compute the local shortest path in order to improve the efficiency of algorithm. Particularly, we implement these two algorithms by programming and apply a large of randomly generated data to test them. The experiment results show that our algorithm is correct and efficient.**

Keywords-computational geometry; the shortest path; convex chain

I. INTRODUCTION

The shortest path problem is a typical problem in computational geometry [1]. Given a start point, a target point and a series of obstacles in the plane, finding the shortest path which visits all the obstacles from the start point to the target point is the main goal. These obstacles can be abstracted to points, segments and polygons, etc. In many applications, the shortest visiting path according to the given order of the obstacles always needs to be found, for example, in the Zoo keeper or Safari problems, the obstacles are the given order polygons[2]. In the Watchman Route Problem, the obstacles are the given order segments [3]. In the Traveling Salesman Problem, the obstacles are the given order points [4]. When the visited order of the obstacles is not specified, the problem is NP hard [4]. In this paper, visiting given order disjoint segments is mainly discussed. The solution of the problem will be contributed to the Watchman Route Problem.

The problem in this paper can be described as follows. Given two points p and q, and a sequence of disjoint segments S={s1,s2,…,sn}(n>0) in the plane. The goal is to find the shortest path from p to q which visits each segment in the given order. An example is shown in Fig.1, the solid lines are 3 given disjoint segments, and the dotted line is the calculated shortest path from p to q.

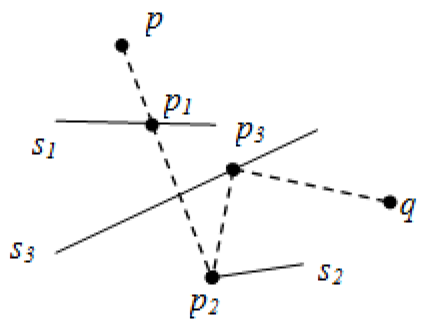


Figure1. The shortest paths of visiting disjoint segments in the plane (n=3).

The problem discussed in this paper is the logical model for solving many application problems, such as image analysis, robot motion planning and Watchman Route Problem. [1]. Many scholars have been studying this problem. In 1984, D. T. Lee and F. P. Preparata proposed Funnel algorithm[5], in 2007, Fajie Li and Reinhard Kettle proposed rubber-band algorithm with K(ε)=(L0 − L)/ ε running time[6-7], L is the shortest path length, L0 is the initial path length, and n is the number of the segments. When n is larger, the time complexity of rubber-band algorithm will be O (n2), which was shown in our previous work in 2011[8]. So far, no better algorithm has been reported.

In this paper, based on the rubber-band algorithm, an O(n(log2n) 2) time fast algorithm is put forward, which adopts binary-search tree to store local shortest path, and applies the convex chain division and combination optimization technique to compute the local shortest path to solve the problem. To verify the correctness and effectiveness of the algorithm, this paper randomly generates

100 groups of data, and each group of data contains 50000 segments. Based on the average running time of 100 groups, we make the contrast analysis of the improved algorithm and rubber-band algorithm. The experiment results show that the improved algorithm is superior to the rubber-band algorithm.

II. RUBBER-BAND ALGORITHM

The method of rubber-band algorithm is described as follows. At first, an initial path is formed by connecting the centers of all the segments. Next, the path point in each segment will be updated from the first path point according to the local optimization computation method, and then a new path is formed. Such a process of path-updated is called an iteration calculation. If the difference of two path lengths is less than the maximum possible numerical accuracy, the algorithm stops. Otherwise, the calculation will be done repeatedly.

```
Rubber-band algorithm.
Input: the segment set S={s1,s2,…,sn}, and two points p and q.
Output: d, which is the shortest path from p to q.
Procedure.
1. Let $\varepsilon$ =10-10 (the chosen accuracy).
2.Let pi be the center of si (for i=1,2, ···,n), and compute the length the initial path d={p,p1, p2,…,pn,q } ,denoted by d1.
3. Let b1=p and i=1.
4. While ( i < = n-1 ) do:
4.1. Let b3=pi+1.
4.2.Compute a point $b_2 \in s_i$ ,such that dist(b1, b2)+dist(b2, b3)=min{dist(b1, b′)+dist(b′, b3): b′∈si }.
4.3. Replace pi with b2 and Update d .
4.4. let b1=pi and i=i+1.
5. Let b1=pn-1 and b3=b.
6. Compute $b_2 \in s_n$ ,such that dist(b1, b2)+dist(b2, b3)=min{dist(b1, b′)+dist(b′, b3): b′∈sn }.
7. Replace pn with b2 and Update d.
8. Compute the updated path length, denoted by d2.
9. Let m=d1-d2。
10. if $m > \varepsilon$ then let d1=d2, and goto step 3.
11. Otherwise, stop.
```

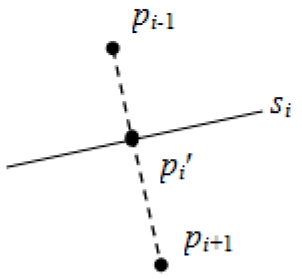
In rubber-band algorithm, from the first point, all path points will be updated in each iteration. Thus, large time will be spent for all the iteration calculations, which makes the efficiency of the algorithm lower. In this paper, we adopt the divide-and-conquer method to solve. The basic idea is the whole problem is divided into some sub-problems and the solution is got by merging the sub-problems. The combination optimization technique and merging method are applied to the improved algorithm in order to make the algorithm more efficient.

III. THE METHOD OF LOCAL PATH OPTIMIZATION

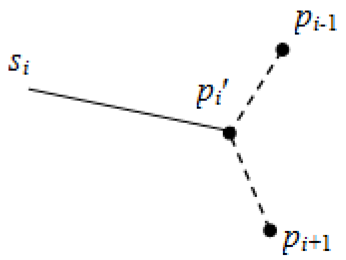
In the process of merging, the local optimal path points are updated according to the location relationship of the points and segments. So, here, we first look at the location relationship of them. Assuming that pi-1, pi and pi+1 are the path points on the segments si-1, si and si+1, which has the following three kinds of situations:

Case1. $\overline{p_{i-1}\, p_{i+1}}$ intersects with si or the extension of si

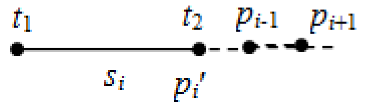
There are three situations in this case: 1) $\overline{p_{i-1}\, p_{i+1}}$ intersects with si at the point pi′, then pi′ is the new path point on si , and $\overline{p_{i-1}\, p_{i+1}}$ is the local optimal path, see Fig.2 (a). 2) $\overline{p_{i-1}\, p_{i+1}}$ intersects with the extension of si, then the nearest endpoint of si with $\overline{p_{i-1}\, p_{i+1}}$ will be chosen as the new path point of si,, see Fig.2 (b). 3) $\overline{p_{i-1}\, p_{i+1}}$ lies on si or the extension of si , then we can easily find a point pi′ on si such that the path from pi-1 to pi is the shortest, see Fig.2 (c).



(a) $\overline{p_{i-1}\, p_{i+1}}$ intersects with s_i



(b) $\overline{p_{i-1}\, p_{i+1}}$ intersects with the extension of s_i

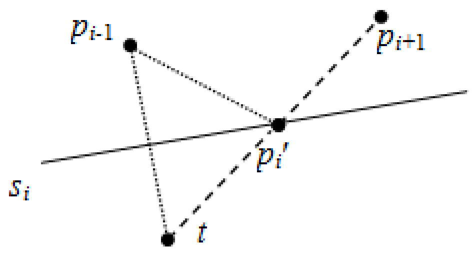


(c) $\overline{p_{i-1}\, p_{i+1}}$ *lie*s on the extension of s_i

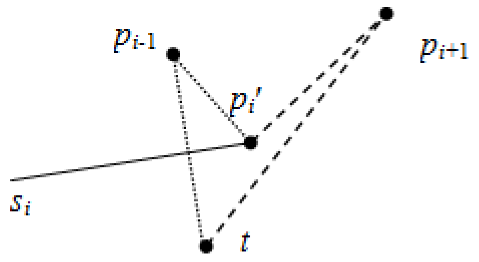
Figure 2. $\overline{p_{i-1}\, p_{i+1}}$ intersects with si or the extension of $_{\text{si}}$.

Case2. $\overline{p_{i-1}\, p_{i+1}}$ doesn’t intersect with si or the extension of si

In this case, we can compute the local optimal points on si by the reflection principle. We first compute the symmetrical point t of pi-1 or pi+1 with the segment si, then the computation is the same as Case 1, see Fig3 (a) and Fig3 (b).



(a) $\overline{p_{i+1}\,t}$ intersects with s_i



(b) $\overline{p_{i+1}\,t}$ doesn't intersect with s_i

Figure 3. $\overline{p_{i\text{-}1}\,p_{i+1}}$ doesn't intersect with si or the extension of $_{\mathrm{si}}$.

Here, we call the local optimal path computed each time convex chain, and it is worth mentioning that the local optimal path is not a strictly convex chain.

IV. IMPROVED ALGORITHM

In this paper, we adopt the combination optimization technique to solve this problem. At first, we divide the problem into many small sub-problems and compute the local shortest path which is called convex chain of every sub-problem. Then we merge the local shortest path into the global optimal path gradually. Finally, we obtain the solution of the problem. In the merging process, we adopt the binary-search tree to store the local optimal path (convex chain) and shorten the iteration process constantly by the method of local path optimization until the shortest path is formed. The key is as follows.

Step1. Initialize the local optimal path and Initialize the binary-search tree

1.1. Divide the segments set S={p, s1, …, sn, q} into many sub-sets by putting the two adjacent segments in a sub-set from the first segment s1, and the starting point p and the target point q is taken as a special segment respectively. When the total number of line segments is odd, only one line segment is contained in the final sub-set,

1.2. Compute the shortest path for the every sub-set and form the local optimal path (convex chain).

1.3. Apply the binary-search tree to storage each convex chain and the leaf nodes of binary-search
tree are formed.

Step2. Merge the leaf nodes of the binary-search tree by two with the method of local path optimization. When there is only one sub-tree, take it as the left sub-tree of the binary-search tree.

Step3. Running setp2 repeatedly, until one binary search tree is formed at last. Output the shortest path.

A. The merging process of local optimal pathy

Here,we discuss the binary-search tree merging process.

Assuming that the left sub-tree is denoted by L, called L chain, and the right sub-tree is denoted by R, called R chain. According to the local optimal method, we check whether the last path point pi of L chain and the first path point pi+1 of R need to be updated and do the corresponding processing. The method is as follows.

1. If the points pi and pi+1 needn't to be updated, connect pi with pi+1 directly and form the local shortest path. Then, merge the two sub-trees and store it to the parent node of binary-search tree. The merge operation stops.

2. If the point pi in L chain needs to be updated, do recursive processing with left and right sub-trees of L chain until the path point pi no longer needs to be updated.

3. If the point pi+1 in R chain needs to be updated, do recursive processing with left and right sub-trees of R chain until the path point pi+1 no longer needs to be updated.

4. Goto step1 to check whether the path points pi and pi+1 need to be updated again. Here, because the update of the points in the R chain may cause the update of the point in L chain again.

The merging method and binary search tree storage technique described above can improve the efficient of the algorithm. Take the Fig.4 for example, L={ pp1p2p3} ,R={ p4p5p6 q}, when merging the L and R, no matter how the path point p3 in segment s3 changes, the path point p4 in segment s4 needs not be updated. Compared with rubber-band algorithm, the characteristics of the local shortest path in our algorithm greatly simplifies the computation of shortest path, and greatly reduces the iteration process. In addition, in this paper, since the given segment is ordered, and the update of points in the process of merging is also in accordance with the given order, so adopting binary-search tree storage will make the calculation easy.

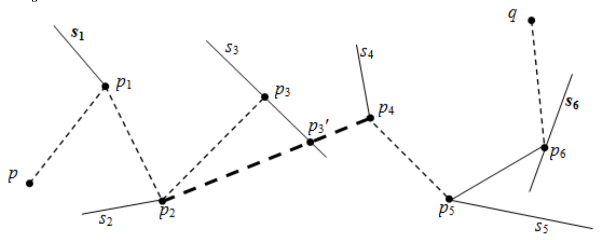


Figure 4. The example of merging two local paths.

B. The improved algorithm

The improved algorithm in this paper contains two procedures which are main procedure and merging procedure. The main procedure first initializes the local shortest path, and then calls the merging procedure. The Merging procedure finishes the computation of merging local optimal path and forms a new local path. The two procedures can be described as follows.

Main procedure.

1. Take the given line segment sequence s={p,s1,s2,…,sn,q} into two sub-sets every two, here, p and q are taken as two segments, and then calculate the local shortest path of each sub-set, which are stored as the leaf nodes of a binary search tree.
2. Select two leaf nodes of the trees which have not been merged, and call the merging procedure.
3. Goto step2 until the root nodes of tree is generated.
4. Output the globe shortest path L={p,s1,s2,…,sn,q}.
5. Stop.

Merging procedure.

Here, the L,R,pi and pi+1 are the same as the definition in section 4.1. Let mergedPath be a pointer to root node, pathNodes be the sequence of local path points, let parentsNode, lChild, rlChild be the parent node, left child node and right child node of current merged-tree respectively.

```
Input: L , R.
Output : mergedPath.
LSPMerge(L, R)
{
if(pi and pi+1 are not updated)
{ mergedPath. pathNodes ={ p,p1,p2,…,pn, q };
mergedPath.lChild= L;
mergedPath.rChild= R;
L. parentsNode = mergedPath;
R. parentsNode = mergedPath;
return mergedPath; }
else{
if(pi is updated)
{ while(L.rChild!=Null)
L = L.rChild;
    update pi by the new path point;
while(L.parentsNode!=Null)
{ tempNode=LSPMerge(L.parentsNode.lChild, L);
L= tempNode; }
}
if(pi+1 is updated)
{ while(R.lChild!=Null)
R = R.lChild;
update pi +1;
while(R.parentsNode!=Null)
{ tempNode=LSPMerge(R,R.parentsNode.rChild);
R = tempNode; }
}
mergedPath= LSPMerge(L, R);
return mergedPath;
}
}
```

C. *The computational complexity of the improved algorithm*

The improved algorithm adopts the binary-search tree storage method and applies it to the merging process. If no local paths are updated in the merging process, the depth of merging is no more than the height (log_n) of the binary tree, and the tree nodes can be computed easily, which is the best case, and the time complexity is $O(nlog_n)$. Otherwise, the path can be calculated iteratively in merging processing which can be carried out with log_n times at most. Thus, the time complexity is no more than $O(n(log_2n)_2)$.

V. THE ANALYSIS OF EXPERIMENT RESULT

In this paper, we implement the rubber-band algorithm and improve algorithms by programming. We first randomly generate 100 groups of test data, which contains 50000 disjoint segments in each group, and we calculate the average running time of 100 groups of data respectively, which is shown in Table1, here, only a part of the data is listed. According to the running results of Table1, we make the time curve chart which can be seen in Fig.5. We can see that the fitting curve equation of rubber-band algorithms is y=0.0015x2-14.175x+27994, which implies the time complexity of rubber-band algorithm is O(n2), while the fitting curve equation of the improved algorithm is no more than the result by the theory analysis, which is O(n(log2n) 2).

TABLE I THE AVERAGE RUNNING TIME OF 100 GROUPS TEST DATA OF TWO ALGORITHMS

the number of segment	the average running time of improved algorithm(ms)	the average running time of rubber-band algorithm(ms)
1000	437	453
3000	2147	2758
5000	3664	12901
10000	12462	50812
15000	30186	134931
20000	70980	317202
25000	132780	605769
30000	187864	851911
35000	260852	1289564
40000	352840	1888752
45000	454293	2282355
50000	597361	3185165

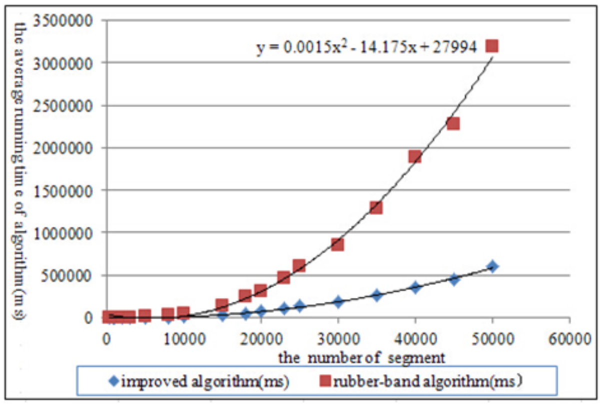


Figure 5. The time curve chart and fitting curve equation .

Furthermore, we present the running result of improved algorithm. To make the result clear,the shortest path of only 10 segments can be shown in Fig.6. The segments numbered is the randomly generated initial disjoint segments, here, the start point and target point is numbered by 1 and 12, and the bold lines are the shortest path from the start point to the target point. The experiment shows that our result is correct and efficient.

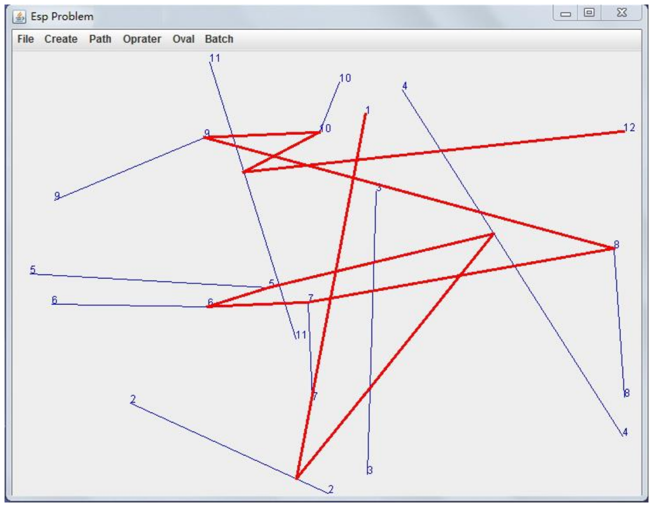


Figure 6. The running result of 10 segments.

VI. CONCLUSION

In this paper, an O O(n(log2n) 2) fast algorithm with the convex chain division technique and binary-search tree storage for the shortest path is put forward. When the given segment in the plane may intersect, the problem becomes more complicate, whether the method can be adapted to the case of segment intersected by some minor modifications is our further studied [9]. In addition, this problem in 3D space will be also further studied [10].

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China (No.61173034, 61272171), General Project of Liao Ning Province Science and Research (No. L2015105) and the General Project of Dalian Institute of Science and Technology (No. KYZ1416).

REFERENCES

[1] M D BERG, M V KREVELD, M OVRMARS, et al. ,“ Computational Geometry: Algorithm and Applications (Second Edition) ,”Springer-Verlag Berlin Heidelberg New York, Berlin,1997.

[2] J HERSHBERGER, J SNOEYINK,“ An efficient solution to the Zookeeper’s problem,” in: CCCG,pp. 104–109, 1994.

[3] A DUMITRESCU , J S. B. MITCHELLAND, et al. ,“Watchman Route for Line and segments,”Algorithm Theory – SWAT 2012 Lecture Notes in Computer Science, vol.7357, pp. 36-47, 2012.

[4] A DUMITRESCU,“The traveling saleman problem for lines and rays in the plane,”The proceeding of the 22nd Canadian Conference on Computational Geometry, Cannada, pp. 257-260, 2012.

[5] D. T. Lee and F. P. Preparata,“ Euclidean shortest paths in the presence of rectilinear barriers,”Networks, vol 14,pp. 393–410, 1984.

[6] F J LI, R KLETTE,“ Rubberband algorithms for solving various 2D or 3D shortest path problems,”In Proc.Computing:Theory and Applications, Platinum Jubilee Conference of the Indian Statistical Institute, pp. 9–18, 2007.

[7] F J LI, R KLETTE,“ Exact and approximate algorithms for the calculation of shortest paths,”IMA journal of management mathematics, vol. 17, No. 1-4, pp. 134–138, 2006.

[8] L J WANG, L HUO, D D He,“ An Improved Algorithm for Euclidean Shortest Paths of Visiting Line Segments in the Plane,”Journal of Convergence Information Technology, vol. 6, No. 06, pp. 119-125,2011.

[9] L J WANG, B JIANG, A S Deng, Q WEI,“ Fast Computation of Shortest Path for Visiting Segments in the Plane,”The Open Cybernetics & Systemics Journal, vol 8, pp. 224-229, 2014.

[10] J. CHOI, J. SELLEN, C.-K. YAP,“Approximate Euclidean shortest path in 3-space,”In Proc.ACM Conf. Computational Geometry,pp. 41–48,1994.

IJS^{3}T

Volume 17, Number 23, 2016

ISSN 1473-8031