Improvement of Dijkstra's Algorithm and Its Application in Route Planning

DongKai Fan
School of Transportation and Vehicle Engineering,
Shandong University of Technology
Zibo, Shandong, P.R.China, 255049

Ping Shi
School of Architecture Engineering,
Shandong University of Technology
Zibo, Shandong, P.R.China, 255049

Abstract: **In order to improve the efficiency of road network route planning，many experts and scholars have conducted some studies，Dijkstra's algorithm is a research hotspot. The Dijkstra's algorithm has its own shortcomings when seeking an optimal path between two points, but it has irreplaceable advantages. Through the analysis of strengths and weaknesses of the classic Dijkstra's algorithm, we can find that the main drawbacks can be summarized as two points: storage structure and searching area. Therefore, the paper has improved these two points, namely the improvement of data storage structure and the searching area of restricted algorithms. And its validity is obtained by analyzing the experimental results.**

Key words: **Dijkstra's algorithm; road network; route planning; storage structure; restricted searching area**

I. INTRODUCTION

Based on a topological structure of the road network, Path planning is a process to search the optimal route for a vehicle before or during traveling to get its destination and a specific application of the shortest path problem in vehicle navigation system [1]. Dijkstra's algorithm is the most classical and mature algorithm for searching a shortest path in the graph, however, this algorithm has the highly time complexity and takes up a larger storage space. In order to overcome the shortcomings of Dijkstra's algorithm, a considerable amount of researches have been done by many experts and scholars, and their research results have respective traits and merits, but the theoretical basis of related algorithm is Dijkstra's algorithm[2~7]. Combining actual traffic network's distribution characteristic, this paper has improved the classical Dijkstra's algorithm to reduce the complexity of time and space and decrease the searching scale of the algorithm, and improve this algorithm's running efficiency.

II. STORAGE STRUCTURE

A. Improvement of Storage Structure

Figures are adopted to explain an improvement principle of the storage structure. Shown in Figure 1, if we adopt the adjacency matrix to store data of classical Dijkstra's algorithm, and then Figure 2 is used to represent the adjacency matrix structure of the topological relations among each point. A 10×10 matrix is used to store the topological relations between points and points, the same number of rows and columns. The numeral values of the matrix $([x_i,x_j])$ are corresponding to the values between points i and j, with the value is zero when start point and end point are the same point, and the value is ∞ when there is no direct path between two points. Then this matrix contains a large number of zero and ∞, which increases the number of invalid cycles and takes the massive space in storage. Therefore it is unscientific from the perspective of space and time.

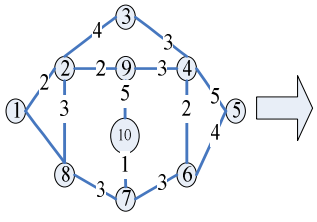


Figure 1. Diagram of spatial topological structure

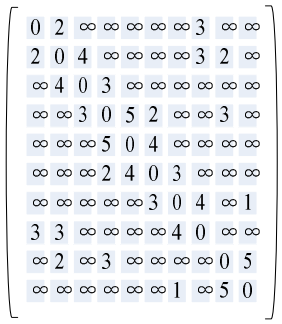


Figure 2. Diagram of adjacency matrix structure

About the storage modes mentioned above, the data storage space can be compressed by reducing the unnecessary zero and ∞. And the data can be stored by adjacency lists.

(1) Memory matrix G is expressed by list array MGragh. Each row of the adjacency matrix is expressed by a single linked list, and only the non-∞ elements are stored in the linked list. Each node has two elements, one is the column value, and the other is the weight. The storage format of Figure 1 is shown in Figure 3.

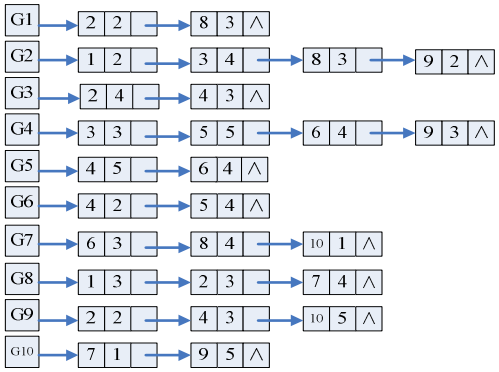


Figure 3. Storage map of adjacency list

(2) The shortest distance from every vertex to the source point is stored in a one-dimension array D .

(3) The precursor point is stored in a one-dimension array P.

(4) The node that is participating in the comparison is stored in an auxiliary two-way linked list.

Steps of algorithm are as follows:

Step 1. The $D[v_i]$ of the node which directly connects with the v_0 will be initialized to its weight, and the rest is maximum value which is allowed by the computer.

Step 2. The node which directly connects with the v_0 will be added to the path list.

Step 3. The node w that has the smallest weight is found, and deleted in path. If the remaining node is zero, then end.

Step 4. Modify the shortest path: compare $D[v_i]$ and $D[w]+s(w,v_i)$ among the weights of the rest nodes which directly connects with w in G . If the value of $D[v_i]$ is less than the value of $D[w]+s(w,v_i)$ and the value of $D[v_i]$ is ∞, then vi is added in Path. The pre- cursor point of $P[v_i]$ is set as w, and the shortest path is modified as $P[v_i]=D[w]+s(w,v_i)$. And continue from step 2.

B. Complexity Analysis

(1) Analysis of Space Complexity

Since list array MGragh is adopted, the space complexity is $O(T)$, where T is the edge number of directed graph. In the worst case , if $T=n^2$, then the space complexity is $O(n^2)$.

(2) Analysis of Time Complexity

After the storage structure is improved, the time complexity of the implementation algorithm's four steps is as follows:

The time complexity of step 1 is $O(n)$;

The time complexity of step 2 is $O(n)$;

For step 3, the first cycle number is the node number that directly connects with v_0 , that is n_1 .The second cycle number is obtained when n_1-1 adds the number of the nodes which directly links with vertexes the first time found and are at infinite distance from v_0 .To draw analogous conclusions, until the node number is zero, then end. In the worst case, v_0 connects with every node, the cycle number is respectively $(n-1),(n-2),\cdots.1$, then the time complexity is $(n-1)+(n-2)+\cdots+1$, that is $O(n^2)$.

The time complexity of step 4 is $O(T)$. If $T=n^2$, and then the time complexity is $O(n^2)$.

Therefore, the time complexity of Algorithm that improves storage structure is $\max\{O(n),O(T),O(n^2)\}$, that is $O(n^2)$. The time complexity is the same as classical Dijkstra's algorithm.

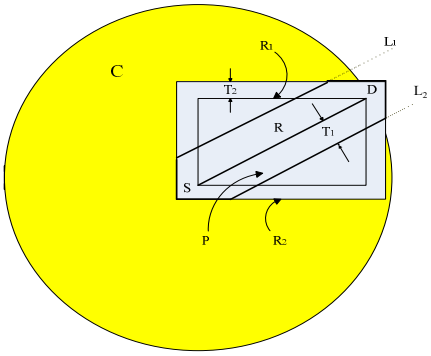
But the nodes of urban road network are large, besides the directed graph corresponding to road net is incomplete, then the degree of node is less than 5, so $T<<n^2$. In this case, improved storage structure is very practical and effectively eliminates storage redundancy and redundancy computation.

III. SEARCHING AREA

A. Restricted Searching Area

Because the classical Dijkstra's algorithm is a comprehensive search process, there must be redundant. According to characteristics of Dijkstra's algorithm and the spatial distribution feature of the real road network, the algorithm restricts the searching area reasonably. Because the straight line between two points is the shortest length, the direction from start point to destination point is generally strike of the shortest path when we plan the route of the real road network. Namely the ultimately actual shortest path between two points is generally on both sides of the connection line, and usually in the vicinity. If there is only one edge between two points, the edge itself is the

shortest path. However, sometimes maybe there exists reserve path of short distance in the two point's vicinity. Namely, in order to enter the right traveled lane ， the vehicle travels the route.



C——area of circle P——area surrounded by black thick lines

S——start point D destination point R—— Euclidean distance from S to D R_1 ——small rectangular R_2 —— big rectangular

T_1 ——threshold（the distance from L_1 , L_2 to SD）

T_2 ——threshold（the distance between two thresholds）

Figure 4. Comparative schematic diagram of the classical Dijkstra's algorithm and restricted searching area algorithm

Shown in Fig 4, in order to search the shortest path from S to D, the theoretical searching area of a classical Dijkstra's algorithm is a circle C that circle center is S and radius is R. the theoretical searching area of restricted searching area algorithm is threshold R_1 that diagonal line is the line from S to D (there is a line segment when the line from S to D is horizontal or vertical). Simultaneously, considering that there maybe exist reserve path, each side of R_1 can be extended outward a threshold T_2, to form a larger rectangle R_2. Then the restricted searching area is formed by L_1 and L_2 to cut rectangular R_2, L_1 and L_2 are two lines that parallel to line segment SD and the distance is T_1. Shown in Figure 4, the searching area P is enclosed by thick lines.

B. Searching Area Analysis

(1) The searching area of the classical Dijkstra's algorithm is $A_1 = \pi R^2$;

(2) The searching area of the restricted searching area algorithm is $A_2 < 2T_1[R + \sqrt{2}T_2]$;

(3) Because the searching node is proportional to the searching area, their time complexity is respectively $O(A_1^2)$ and $O(A_2^2)$, commonly two threshold of T_1 and T_2 are relatively small constant. Thus, the ratio of their time complexity can be approximately expressed as $O(A_1^2)/O(A_2^2) \approx A_1^2 / A_2^2 \approx R^4 / R^2 = R^2$.

By comparison, we can find that the searching area of the restricted searching area algorithm is smaller than that of the classical Dijkstra's algorithm. With the increase of the road network scale and distance of neighbor nodes, the difference of time complexity is more significant. Therefore, the improved algorithm restricting the searching area reasonably will reduce the scale of algorithm searching, minimize the complexity; and improve the efficiency of a system.

IV. Application Examples and Results Analysis

In order to compare the advantage and disadvantage of the two algorithms, stochastic road network with 3600 nodes is realized by VC++ (CPU: 2.0; main memory; 512M; OS: WINXP).

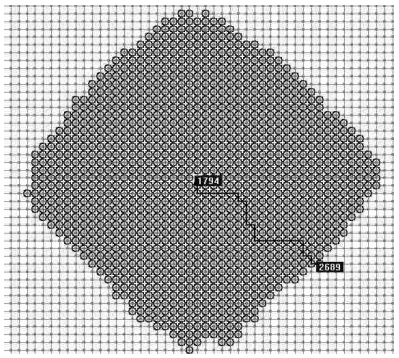


Figure 5.Searching area schematic diagram of the classical Dijkstra's algorithm

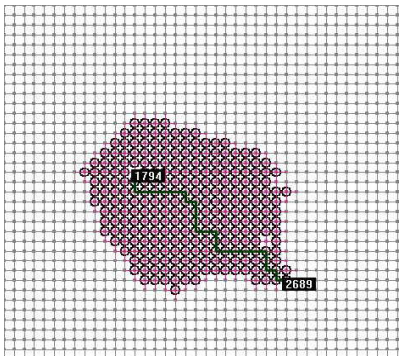


Figure 6. Searching area schematic diagram of the improved algorithm

Only one search result is given in this paper, from the start point 1794 to the destination point 2689. The search result of the classical Dijkstra's algorithm is

given in Fig 5, which the number of searched nodes is 627; the search result of the improved algorithm is given in Fig 6, which the number of searched nodes is 243.

The black lines are the shortest path of the relevant algorithm. Through comparing, we can find that the shortest path is the same, but the searching area of the classical Dijkstra's algorithm is very close to a circular, while the searching area of the improved algorithm is very close to a rectangular.

Eight groups of test data are given in table 1, while t1 is the searching time of the classical Dijkstra's algorithm; t2 is the searching time of the improved algorithm; n1 is the node number that the classical Dijkstra's algorithm has searched; n2 is the node number that the improved algorithm has searched.

TABLE 1. THE SEARCHING RESULTS OF TWO ALGORITHMS

start points	destination points	Dijkstra's algorithm		Improved algorithm		t2/t1(%)	n2/n1(%)
		t1(s)	n1	t2(s)	n2		
1794	2689	2.365	627	0.156	243	6.60%	38.60%
33	675	1.433	486	0.092	183	6.40%	37.70%
689	1980	2.895	974	0.214	352	7.10%	36.10%
1632	3520	4.198	1395	0.293	517	7.00%	37.10%
1938	3211	2.877	955	0.212	339	7.40%	35.50%
1222	3333	3.985	1541	0.277	578	7.00%	37.50%
769	2813	4.061	1480	0.337	562	8.30%	38.10%
1137	2928	3.846	1228	0.279	484	7.70%	39.40%

From the data of the Tab1, we can find that the node number searched by the improved algorithm is about 1/3 compared with the classical Dijkstra's algorithm, and the searching time of the improved algorithm is about 1/10 compared with the classical Dijkstra's algorithm. So, we draw a conclusion that the real-time performance of route planning is significantly increased.

Unfortunately, the improved algorithm itself restricts searching area; maybe the algorithm sometimes can't search the genuine shortest path, and can only be an approximate shortest path. It is a defective shortest path algorithm, and reduces the accuracy while optimizing. But in most conditions, the shortest path is in the rectangular inner. Overall speaking, the advantage of the optimization algorithm is evident and the improved algorithm obtains the purpose of optimization.

V. CONCLUSION

This paper，which bases on the municipal transportation cyberspace distribution characteristic and combines of the classical Dijkstra's algorithm to its own traits reduces the searching scale of algorithm and improves running efficiency. The results show that the improvement of algorithm is reasonable and effective.

REFERENCES

[1] Zhao Yilin. Vehicle Location and Navigation System.Beijing: Publishing House of Electronics Industry, 1999.

[2] Lu Feng. Shortest path algorithm: Taxonomy and Advance in Research. Acta Geodaetica et Car tographica Sinica, 2001, 30(3): 269~275.

[3] Fu Mengyin, Li Ji, Deng Zhihong. A Hierarchical Route Planning Algorithm with Restricted Search Area. Journal of Computer-Aided Design & Computer Graphics, 2005, 17(8): 1773~1777.

[4] Su Haibin, Zhang Jitao. A New Algorithm of Hierarchical Route Planning with Restricted Search Area. Journal of Henan University (Natural Science). 2008, 8(1):81~84.

[5] Yan Hanbing, Liu Yingchun. A New Algorithm for Finding Shortcut in a City s Road Net Based on GIS Technology . Chinses Journal of Computers, 2000, 23(2):210~215.

[6] Li Qingquan, Zheng Nianbo, Xu Jinghai. A Hierarchical Route Planning Algorithm Based on Multi-level Topological Structure of Road Network. Journal of Image and Graphics, 2007, 12 (7): 1280~1285.

[7] Wang Yawen, Wang Xili, Cao Han .Multi-scale optimal route planning algorithm within restricted searching area. Application Research of Computers, 2007, 24 (12):66~68.

[8] Yan Weimin, Wu Weimin. Data Structure.Beijing: Tsinghua University Press,2002.