An Improved Dijkstra Shortest Path Algorithm

Yizhen Huang, Qingming Yi, Min Shi
College of Information Science and Technology, Jinan University
Guangzhou, China
e-mail: t_asic_jnu@163.com

***Abstract*—An improved Dijkstra shortest path algorithm is presented in this paper. The improved algorithm introduces a constraint function with weighted value to solve the defects of the data structure storage, such as lots of redundancy of space and time. The number of search nodes is reduced by ignoring reversed nodes and the weighted value is flexibly changed to adapt to different network complexity. The simulation experiment results show that the number of nodes on search shortest path and computation time is significantly reduced. Thus, improved algorithm can be faster to search out the target nodes.**

Keywords-Dijkstra algorithm; Shortest path; Constraint function

I. INTRODUCTION

The shortest path problem is always encountered in our daily life. For example, with the overall popularization of e-commerce, the rapid rise of the logistics industry and the increasing of road traffic complexity and congestion, how to improve material distribution [1] efficiency and travel efficiency as well as road utilization in current transportation network can be reduced to the shortest path problem. In order to solve shortest path problem, the Intelligent Transportation Systems (ITS) emerge as the times require. Besides, path planning plays an important role in the Intelligent Transportation Systems. Therefore, it is crucial to increase the efficiency of path planning. In order to make sure that computer and path planning algorithm can identify the network and calculate the shortest path on network, one important step of path planning is to use appropriate data structure to store the network information. The most widely used shortest path optimization algorithm is Dijkstra algorithm [2, 3] and the A* algorithm [4, 5]. Dijkstra algorithm is easy to be implemented, but it has low search efficiency due to its large computations. Computational complexity of A* algorithm is relatively smaller than the previous one, but it is easy to fall into endless loop [6]. Greatly limited by hardware in the past, the research on above algorithms mainly focus on the improvements of the data structure storage, in which adjacency matrix and adjacency list are commonly used. However, in such storage modes, the number of elements in correlation matrix will expand as geometric series with the increasing of node numbers. It results in a serious waste of resource space and large amount of calculation [7] due to large number of elements of *0* and ∞ in correlation matrix.

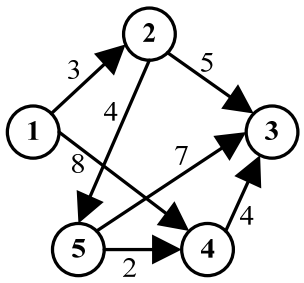
With the rapid development of hardware technology, storage structure is not the main problem to restrict the path planning technology any more [8]. The popular research direction turns from storage structure to search strategy. Therefore, we propose a new shortest path algorithm introducing a constraint function with weighted value. In the improved Dijkstra shortest path algorithm, the number of search nodes is reduced by ignoring reversed nodes. And the weighted value is flexibly changed to adapt to different network complexity. The simulation experiment results show that both of the number of nodes on search shortest path and computation time are significantly reduced.

II. ALGORITHM BASED ON STORAGE MODE OF DATA STRUCTURE

One important step in the algorithm based on data structure storage is to utilize appropriate data structure, generally the adjacency matrixes, to store the network information. Suppose that there is the digraph $G=(V,E)$ $V=\{v_1, v_2, ..., v_n\}$ including n nodes. For weighted directed graph, its adjacency matrix $A = (a_{ij})_{n\times n}$ is defined as follows:

$$a_{ij}=\begin{cases} w_{ij} & ,(v_i, v_j)\in E \\ \infty & ,(v_i, v_j)\notin E \end{cases} \quad ,i, j = 1, 2...n \qquad (1)$$

where w_{ij} denotes the weight of arc $<v_i, v_j>$, ∞ denotes that there is no edge between v_i and v_j [9]. The adjacency matrix is shown in Fig.1:



(a)

	1	2	3	4	5
1	0,	3,	∞,	8,	∞
2	3,	0,	5,	∞,	4
3	∞,	5,	0,	4,	7
4	8,	∞,	4,	0,	2
5	∞,	4,	7,	2,	0

(b)

Figure 1. Map representation (a). weighted digraph (b). adjacency matrix

General algorithm based on storage of data structure stores the network information with appropriate data structure, such as adjacency matrix, before carrying out the

shortest path search. *S* denotes the set of the end points that have found the shortest path from the original point. (*V*-*S*) denotes the set of nodes that have not yet calculated the shortest path. Then according to the theory of Dijkstra algorithm, which in order of path length increasing gradually search out the shortest path. The main steps of the algorithm are as follows [9]:

1. Use adjacency matrix *C* to store network information. C_{ij} denotes the weight of arc $< v_i, v_j >$. If there is no arc between v_i and v_j, then C_{ij} is set to ∞. d_i is defined as the weight from the source points to node v_i. Initialize starting point as $d_s = 0$ and $D_i = C_{si}$.

2. Select v_p ,then we have

$$d_p = \min\{d_p \mid v_p \in V - S\} \tag{2}$$

v_p is the end point in the shortest path starting from the source point v_s . Then,

$$S = S \cup \{v_i\} \tag{3}$$

Set the end point to v_t. If v_p is equal to v_t , which means d_p is the shortest path from starting point v_s to end point v_t , then algorithm stops. Otherwise, turn to step3).

3. Modify the length of the shortest path from v_s to any point v_p in set of (*V* – *S*), and sadisfy d_p as follow:

$$d_p = \min[d_i, d_k + l_{kp}], \quad v_p \in V - S, v_k \in S \tag{4}$$

where l_{kp} is the direct distance from point *k* to point *j*.

4. Repeat step 2 and step 3, until the shortest path is found from starting point v_s to end point v_t .

III. IMPROVED ALGORITHM BASED ON SEARCH STRATEGY

The algorithm based on the storage of data structure stores network information with appropriate data structure before carrying out the shortest path search. Under the limitation of the hardware condition, it is beneficial to identify network and calculate the shortest path by the computer and the path planning algorithm. But in Fig.1, using adjacency matrix to store network information results in lots of elements ∞ and 0. The number of elements in correlation matrix expands as geometric series with the increasing of node numbers, which brings a serious waste of resource space and lots of useless cycle. Thus search efficiency is seriously reduced. Therefore, a new algorithm aiming at improving search efficiency and find the shortest path quickly must be proposed.

Due to the limitation of the hardware condition in the past, conventional algorithms mainly depend on data structure storage improvement. Considering the defect of low search efficiency in conventional algorithm, one improved shortest path algorithm is presented in this paper, which introduces a constraint function with weighted value. This algorithm introduces constraint conditions for searching each position in the state space to guide the search forward to expected direction. Meanwhile, weighted value is flexibly changed to adapt to different network complexity. By introducing constraint function, it can omit lots of useless search path and accelerate the process of problem-solving. In this case, the proposed algorithm can find the optimal solution and greatly improve search efficiency.

A. Improvements of The Shortest Path Algorithm

The algorithms based on data structure storage will use appropriate data structure to store the information of the network before carrying out the shortest path search. Then according to the theory of Dijkstra algorithm, which in order of path length increasing gradually search out the shortest path. The core formula of the Dijkstra algorithm is shown in (4). Through this formula, it gradually compares the minimum weight between the current optimal node and each node in the set of nodes that have not yet calculated the shortest path. The node that has been found with minimum weight replaces the current optimal node until the search stop. The characteristic of this search method makes the starting point as the center of circle and the distance from the beginning to the end as the radius of circle, which has low search efficiency. Therefore, in order to reduce the search range, the proposed improved algorithm based on the search strategy introduces constraint functions r(n) for each searching node in state space. So the core formula of the improved shortest algorithm is defined as follows:

$$\begin{cases} D(n) = d(n) + r(n) \\ r(n) = \omega * \cos\theta_n \end{cases}, -\frac{\pi}{2} \le \theta_n \le \frac{\pi}{2} \tag{5}$$

where d(n) denotes the weight value of shortest path from the starting point to the current node *n*. r(n) is constraint function. ω is a weighted value denoting the impact factor . θ_n is the angle between the vector that consists of nodes from staring point to current node and the vector that consists of nodes from staring point to the end point. Because of the limitation of θ_n , the search range of is reduced and the search efficiency is improved. When d(n) is large, in order to make sure that the constraint functions r(n) is not much less than d(n) in searching the shortest path, the impact factor should be set to a large value. Because $\cos\theta_n$ belongs to [-1,1]. When r(n) is far more less than d(n) or r(n) is equal to *0*, there is no constraint condition, and then the improved algorithm becomes a conventional Dijkstra algorithm. Therefore, to avoid falling into endless loop by removing too many relevant nodes, r(n) should not be set to a too small value and the constraint value of current node should not be overestimated. How to choose the

specific form of r(n) depends on the path optimization solution.

B. Algorithm flowchart and steps

Improved algorithm model based on the search strategy is shown in Fig. 2:

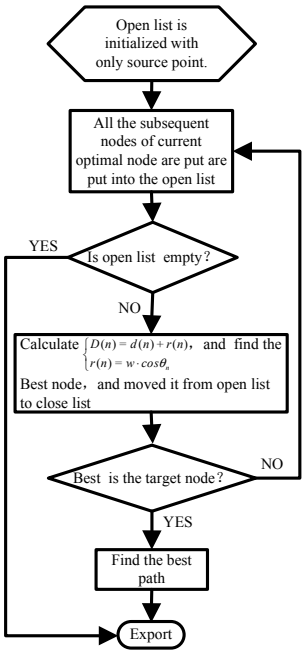


Figure 2. The improved algorithm based on the search strategy

The specific steps of the improved algorithm are as follows: Weighted directed graphs with *n* nodes are stored in the adjacency matrix C. C_{ij} denotes the weight of arc < v_i, v_j >. If there is no arc between v_i and v_j, then C_{ij} is set to ∞. Open list is the set of neighboring nodes that have found the optimal node. And it is initialized to contain signal source point v_S. Close list is a set of optimal nodes and is initialized by an empty set. d_i denotes the weight from v_S to v_i. The searching process of the shortest path from the starting point v_S to destination point v_i is as follows:

1. Initialize starting point d_S as *0,* open $=\{v_S\}$, close $=\{\phi\}$;

$$D_i = C_{si} \tag{6}$$

2. Put all the subsequent nodes of the current optimal node into the open list.

$$\text{open} = \{v_S, v_p\} \quad p = \{0,1,...n-1\} \tag{7}$$

3. Compare all subsequent nodes v_p in open list, and make them satisfy(8) :

$$\begin{cases} D(n) = d(n) + r(n) \\ r(n) = \omega * \cos\theta_n \end{cases} \quad , -\frac{\pi}{2} \le \theta_n \le \frac{\pi}{2} \tag{8}$$

Put the node v_p which satisfies (8) from open list into close list, then close={ v_p }; Here, v_p is the current optimal node ,and is passed by the best path through from source to destination node.

4. If current optimal node has subsequent nodes, repeat step 2) and step 3). If the current optimal node is the destination node, it indicates that the best path from the source to destination node has been searched.

IV. EXPERIMENTAL SIMULATION AND ANALYSIS

The simulation environment is based on Visual C++ in PC platform. The experiment is carried on the network with sizes of 50×50, 100×100, 200×200 and 250×250. The experiments number has four groups. The weighted value ω is changed to test the performance of the proposed improved algorithm for various complexity of network. The number of search nodes *N* and search time *T* calculated by conventional algorithm and proposed algorithm in terms of different weighted value ω are shown in Table I.

TABLE I. COMPARISON OF SEARCH NODES AND SEARCH TIME

No.	*Conventional algorithm*		*Improved algorithm*		*Weighted value*
	N/ind	*T/s*	*N/ind*	*T/s*	*ω*
1	18447	2.843	675	3.532	6500
	18447	2.843	1331	0.819	1500
2	61950	5.679	4923	0.196	6500
	61950	5.679	10603	0.339	1500
3	220626	26.954	35795	3.14	6500
	220626	26.954	68587	5.922	1500
4	492510	123.719	124136	8.936	6500
	492510	123.719	231471	37.563	1500

As can be seen from Table I, when there is large number of nodes in network, it means that the network complexity is high, the proposed algorithm based on the search strategy is obviously superior to the conventional algorithm based on the improvement of data structure storage. From the fourth group of experiment results in Table I, the number of search nodes of proposed algorithm is reduced about 75% than that of conventional algorithm, and the search time of proposed algorithm is about 92% less than that of conventional algorithm. It shows that the proposed algorithm is more superior to conventional algorithm both in number of search nodes and search time. Besides, it can be known from the last three groups of experiment results, when the network is more complex, by changing the weighted value ω, the constrain function of the proposed algorithm can make the search direction toward the target node. Thus the efficiency is improved about 42%~76%. However, the first group of experiment results shows that when the network is relatively simple, the higher the weighted value ω is, the lower efficiency of searching shortest path is. That is because when the number of network is small, the greater the weighted value is, the more advantageous nodes would be removed

while searching for the shortest path, thus the search efficiency reduces. Generally, the previous analysis indicates that the network is more complex, the influence by the weighted value w is more obvious.

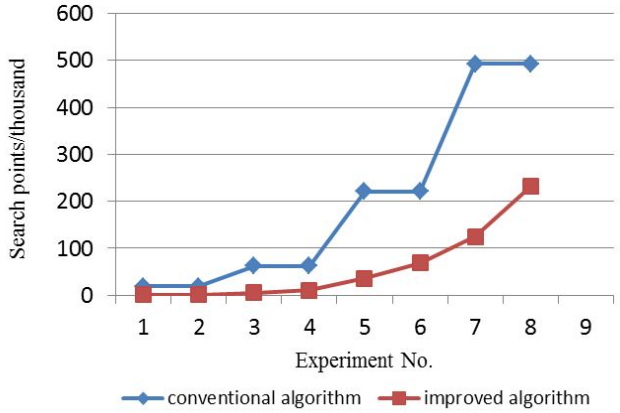


Figure 3. Search points comparison between proposed algorithm with conventional algorithm

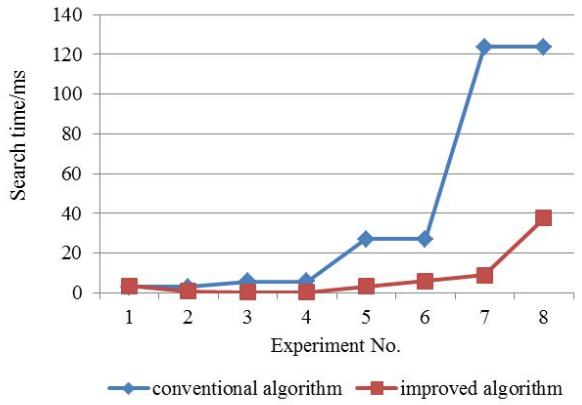


Figure 4. Search time comparison between proposed algorithm with conventional algorithm

Fig.3 and Fig.4 are comparisons on number of search nodes and search time between proposed algorithm and conventional algorithm. When the number of nodes in network is small, the gap between proposed algorithm and conventional algorithm is not obvious, both in search nodes and in search time. But when the network is more complex, proposed algorithm is significantly better than conventional algorithm in both search nodes and search time.

V. CONCLUDING REMARKS

An improved shortest path algorithm based on search strategy is proposed in this paper. In order to reduce the number of search nodes and improve the computation speed, the proposed algorithm introduces a constraint function with weighted value ω and ignores a large number of irrelevant nodes during searching the shortest path. Meanwhile, according to different complexity of map information, the weighted value ω can be flexibly changed to make the constraint function more reasonable to effectively improve the search efficiency. The experiment results show that compared with the conventional algorithm based on the improvement of data structure storage, the proposed algorithm based on search strategy is able to effectively increase the search efficiency, and reduce the number of search nodes during searching the shortest path.

ACKNOWLEDGMENT

This paper is supported by 2011 Education and Academy Project (cgzhzd1103), 2012 Guangdong Province Engineering Research Center project (2012gczxA003), 2011 Guangdong Province Ministry of Education Combination Project (2011B090400231) and 2011 Guangdong Province Ministry of Education Combination Project (2011A090200088).

REFERENCES

[1] Rui H. Optimization and realization of Dijkstra algorithm in logistics. Hang Zhou,Computer Era, vol. 2, 2012, pp.11-12.

[2] Dijkstra E W. A node on two problem in connexion with graphs. Numerische Mathemartik. vol. 1, 1959, pp.269-271.

[3] Paige R, Kruskal C. Parallel algorithms for shorest path problem. IEEE Transactions on Computer, vol.C34, 1985, pp.14-20.

[4] Ismail Chabini,Shan LAN. Adaptationd of the A* Alorithm for the Computation of Fastest Paths in Deterministic Discrete-Time Dynamic Network. IEEE Transations on Intelligent Transportion Systems, vol.4, 2002, pp.121-125.

[5] Chabini I, Lan S. Adaptations of the A*algorithm for the computation of fastest paths in deterministic discrete-time dynamic networks, IEEE Transations on intelligenttransportation systems, vol.3, 2002, pp.60-74.

[6] Jiaohong L. Design and Implementation Vehicle Navigation System on GPS/DR. Xian university of electronic science and technology, 2009, pp.31-33.

[7] Xiaojing Z. Digital map format of the car's navigation product .He nan, Gnss Word of China, 2004, pp.6-9.

[8] Hui S. Research on route planning algorithm in vehicle navigation syatem, 2010, pp.3.

[9] Liu Y. Research and implement of route layout algorithm in vehicle navigation systen. Beijing jiao tong university, 2008, pp.26