PROCEEDINGS OF SCIENCE

Application of A* Algorithm and Its optimization Method

Yunxiang Liu[1]
School of Computer Science and Information Engineering,Shanghai Institute of Technology ,Shanghai, 201400, China
E-mail: yxliu@sit.edu.cn

Jie Du[2]
School of Computer Science and Information Engineering, Shanghai Institute of Technology ,Shanghai, 201400, China
E-mail:stardujie@163.com

Hao Wang
School of Computer Science and Information Engineering,Shanghai Institute of Technology ,Shanghai, 201400, China
*E-mail:*mr.wanghao@163.com

Lu Jia
School of Computer Science and Information Engineering,Shanghai Institute of Technology ,Shanghai, 201400, China
*E-mail:*jialusit@163.com

Nowadays, the traffic congestion is becoming more and more serious. It is urgent to find a path planning algorithm with high efficiency. As a heuristic algorithm, A* algorithm can solve the problem of path planning with high efficiency, and the design of heuristic function is particularly important. By analyzing the problems of route planning, some improvements have been made in this paper: firstly, the evaluation function with two elements of distance and direction are considered; through the process of normalization, the problem of unified unit has been solved; secondly, use the smallest heap spatial to load the node information dynamically, and reduce the memory usage space.

CENet2015
12-13 September 2015
Shanghai, China

[1]Speaker
[2]Correspongding Author

PoS(CENet2015)062

1. Introduction

The problem of shortest path is an important issue of transportation network analysis. The so-called Shortest Path aims at finding the shortest path between two nodes in a figure (composed of nodes and the path). The shortest path search algorithm requires faster and less memory space, indicating the time complexity and space complexity of the algorithm. The classic shortest path search algorithm is Dijkstra algorithm, which belongs to the traversal search; however, when the network node number is large, the algorithm will search greater node number and the efficiency will be lower. As a result, some heuristic search algorithms are proposed including local optimal search method and A* algorithm, etc. The function of heuristic algorithm is to evaluate each search position and get the best location, then search in this location until searching the target in the state space.

At present, in the field of path optimization, A* algorithm [1] is a popular heuristic search algorithm as put forward by Hart, Nilson, Raphael. It uses heuristic search algorithm to estimate the distance of target point so as to reduce the search space and improve the search efficiency. Many literatures have studied the A* algorithm [1] and put forward the elements of distance and direction which are introduced into the valuation function. Because the units of distance and direction are not unified, some problems will arise when using it. Aiming at this problem, I have studied and improved the evaluation function. In addition, in order to further improve the operational efficiency of the algorithm, the paper uses the smallest heap structure to reduce the memory storage space.

2. Basic Principle of A* algorithm

The A* algorithm's whole frame is made of traversal search method; however, it uses the heuristic function to estimate the cost of any point on the map to the target point, so you can choose a good direction. A* algorithm [3] is introduced to the current node j inspired by the function f*(j) and the inspiration of the current node j function is defined as:

$$f^*(j) = g(j) + h^*(j) \tag{2.1}$$

G (j) is a measurement of the actual cost from the starting point to the current node j, h*(j) is an estimate of the minimum cost from the current node j to the destination. Note that if h*(j) = 0, there is no useful global information; then A* algorithm is turned into ordinary Dijkstra algorithm. As a result, the ordinary Dijkstra [2] can be seen as a special form of A* algorithm. H*(j) needs to satisfy the compatibility condition that it can't be higher than the actual minimum cost of node j to the end. If the evaluation function satisfies the compatibility condition and the original problem exists optimal solution, the A* algorithm can search the best path.

The benefit of using A* algorithm [2] is that it uses heuristic function to make the search path more intelligent towards the end, so its search depth is small with less number of nodes; therefore it occupies less storage space, as shown in Fig. 1.

The key of A* algorithm is to design a suitable heuristic function. Some literature analyzed its characteristic, believing that the enlightenment function [11] f*(j) was non-decreasing. As long as f*(j) can satisfy the compatibility condition, namely, the valuation function h*(j) can be smaller than the actual cost of node j to the target point, its generated path must be optimal. Some literature introduced two elements of distance and direction into the construction of the evaluation function [5], namely:

$$h^*(j) = w1 * L + w2 * \alpha \tag{2.2}$$

We can see in Picture two, where L represents the Euclidean distance of the current node to the end , α represents this starting point to this current point and this current point to this end of the line segment angle, that is ∠SJE, the literature also uses the angle between the line segments and related intermediate node associated with the node and the end of the ∠NJE. W1 and w2 are the weighted value of the distance and angle, and they range respectively from 0.55-0.6 and 0.35-0.45, but we can't ignore the problem that the units of distance and the angle aren't unified. It is likely that the distance value is far greater than the angle value ($L >> \alpha$), especially

in a large area path planning process, the problem is more obvious. When L >> α, the direction is no longer binding and makes the evaluation function pointless.

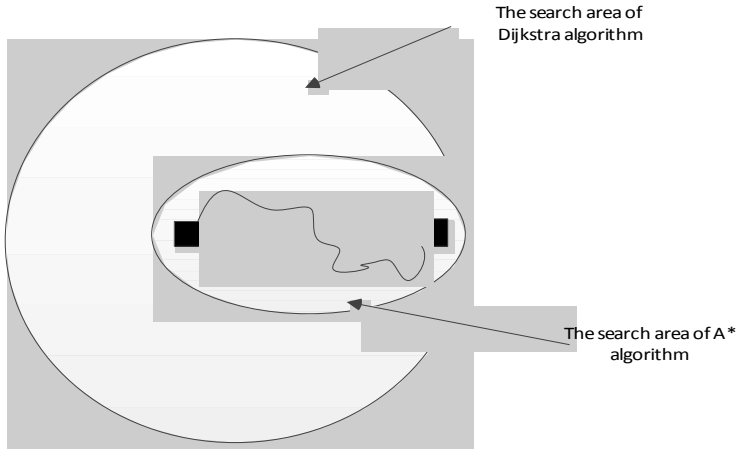

Figure 1 : Comparison between the Searching Area by Dijkstra algorithm and A* Algorithm

2.1 Operation Structure of A* Algorithm

Upon constructing an appropriate valuation function, we should consider its running condition. Currently, most method is to read the data into memory, and then search for the shortest path. Theoretically, A* algorithm [3] can search for the shortest path by searching fewer nodes, but when the algorithm is running, you must consider two questions: firstly, the speed. Even if the data can be quickly read into memory, we must consider the second question, the size of system memory. If the system memory is large enough, by running the algorithm, the repeated searching for the same node will also appear, thereby to reduce the operational efficiency. In view of the problem of data reading, some scholars put forward the A* algorithm based on restricted areas to reduce the amount of data; however, because A* algorithm itself is a costing algorithm, it is more likely that this approach can't search the shortest path, especially when the road attribute information and the traffic restrict information [12] are considered, as shown in Fig. 2.

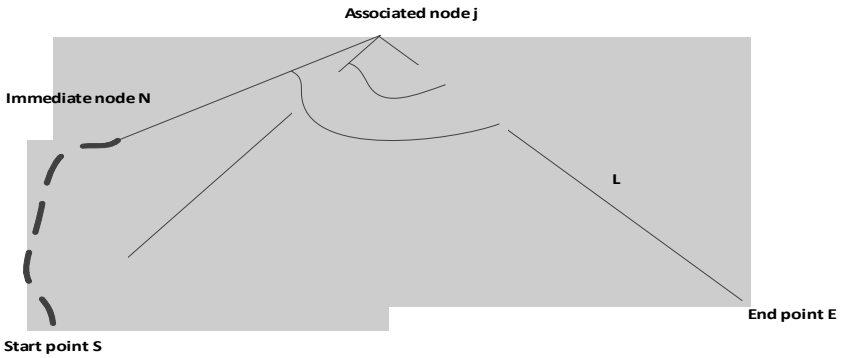

Figure 2: The structure of cost function

3. Improved A*algorithm

3.1 Improvement of A*algorithm

In view of the A* algorithm evaluation function [8], we will carry out the distance and angle normalized unit to compute its associated node corresponding value and find the average value of L and α:

$$L = \frac{1}{n}\sum_{i=1}^{n} Li \tag{3.1}$$

$$\bar{\alpha} = \frac{1}{n}\sum_{i=1}^{n} \partial i \tag{3.2}$$

Where n represents the associated nodes, associated with each node's distance and angle being normalized, the evaluation function becomes:

$$h^*(j) = w1 * \bar{L}' + w2 * \alpha' \tag{3.3}$$

$$L' = L / \bar{L} \tag{3.4}$$

$$\alpha' = \alpha / \bar{\alpha} \tag{3.5}$$

When we have completed the normalization treatment, we'll only consider the distance and the angle's contribution to the path, regardless of distance and its numerical size. Although the operation of the algorithm increases the amount of calculation, we can further reduce the searching time of the algorithm and enhance its operation structure to improve searching efficiency of the algorithm.

Aiming at the problems of algorithm efficiency as mentioned, the paper proposes to establish the smallest heap structure and the dynamic load of road network data in order to improve its efficiency. It is mainly used for multiple attribute data or multidimensional point data [4] which can coordinate rapidly the network data in the area.

In the course of implementation of the algorithm [5], it doesn't start to load all of the road network data, but according to the need of the algorithm, read relevant information of nodes or delete the node information while increasing the I/O operations in the process of operation, but this will avoid a lot of invalid data loading and takes a lot of memory space. For example, to determine whether the current node is within a certain range [8] or not. If not, load the data corresponding to the area; if within a certain range, read relevant information. On this basis, calculate the value of inspiration section and searching for the shortest path.

3.2Realizationof A*Algorithm

In the process of realizing the algorithm, we need to construct two linked lists which are used to store nodes expected to extend; and the extended nodes are called Open list and Close list [4].

The algorithm steps are shown as follows:

1) Initialization settings. Information of the starting point is loaded to the Open list. The Close list assignment is empty so that g (j) = 0.

2) Search the nearest node from the current node [4], namely, the smallest value of f (j); then delete the node j in the Open list and load it in the Close list. Determine whether it is the end. If so, the end, go to Step 4; otherwise, go to Step 3.

3) Judge whether its information of node j is in certain range. If so, the extension is node j; otherwise, load the node information of node j and extend. Go to Step 2.

4) Starting from node j. Output the optimal path and the shortest distance from the start point to the destination point by the method of using back, then the algorithm is terminated.

During the operation of algorithm, if we have avoided the repeated access to the same point, reduce the running time of the algorithm.

4 Experiment and Analysis

The slowest part of the A* algorithm is to find the node of minimum F value in OPEN table [9]. In this paper, a binary heap way is used to store OPEN table nodes and sort the nodes on the OPEN list from small to large; and the efficiency of inserting nodes and deleting nodes in a binary heap [7] is higher than that in an average sorting algorithm.

algorithm	The starting point	The terminal point	The number of search node	The length of the path	Search time /ms
A* algorithm	12,32	72,32	1045	69	391
	12,32	72,32	235	42	111
Improved A* algorithm	12,32	72,32	1045	69	200
	12,32	72,32	235	42	60

Table 1 : Comparison between the A* Algorithm and Its Improved Algorithm

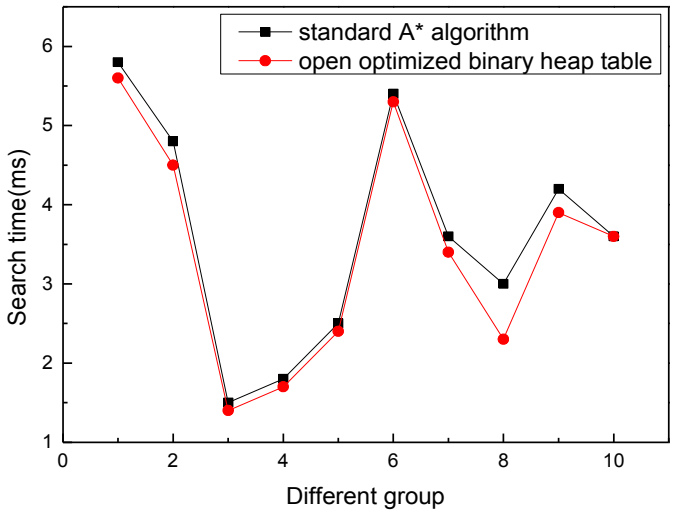

Figure 3 : Comparison Charts of the Improved Algorithm and A* Algorithm

By the above experimental data, it can be concluded that the number of nodes is bigger when the standard A* algorithm can be used to search the path [6], and the path search speed will also be affected. When the Open list is optimized by using binary heap, although the number of node has not changed, the speed to search the node of minimum F value has raised, so it improves the path search speed. On the basis of data analysis, the speed of finding diameter [10] is increased by 5%.

5. Conclusion

A* algorithm is widely used in the path planning as a heuristic search algorithm. We can reduce the search space and improve the search efficiency by using the heuristic function; therefore, it plays a key role in this algorithm. In this paper, the problem of the unified units of distance and angle in the A* algorithm has been studied, and the normalization process of the distance and angle as well as the use of the shortest heap are put forward. Upon experiment, the results show that the efficiency of the improved A* algorithm has been improved significantly.

References

[1] K. I. Trovato, L. *Dorst. Differential A** [J]. IEEE Transactions on Knowledge and Data Engineering, 14(6):1218(2002).

[2] D. X. Hu, H. H. Chen, W. K. Guo. *Neural network and genetic algorithm based global path planning in a static environment* [J]. Journal of Zhejiang University Science, 6(6):549-554(2005).

[3] Q. Y. Chao, Jay Lee, Mandy J. Munro Stasiuk. *Extension to least cost path algorithms for roadway planning*[J]. International Journal of Geigraphical Information Science(IJGIS), 17(4):361-376(2003).

[4] B. I. Cha, S. Lan. *Adaptations of the A* algorithm for the computation of fastest paths in deterministic discrete time dynamic networks*[J]. IEEE Transactions on intelligent transportation systems(2):60-74(2002).

[5] R J Szczerba, P Galkowski, IS Glickstein, et al. *Robust algorithm for real-time route planning*[J]. IEEE Transactions on Aerospace and Electronic Systems, 36(3):869-878(2000).

[6] Richard Kimmel, Allen Bruckstein. *Finding shortest paths on surfaces using level sets propagation*[J]. IEEE Transaction on pattern Analysis and Machine Intelligence, 1995, (6):45~52

[7] F. Wang, S. Z. You. *Application of Dijkstra and Dijkstra-based n-shortest paths algorithm to intelligent transportation systems*[J]. Application Research of Computer, 23(9):203-206(2006). (in Chinese)

[8] X. L. Geng, C. J. Li. *Application of parallel Reinforcement Learning Based on Artificial Neural Network to Adaptive Path Planning*[J]. Science Technology and Engineering,11(4):756(2011).(in Chinese)

[9] I. A. Getting. *The Global Positioning Syatem*[J]. IEEE Spectrum, 12:36-38, 43-47(1993).

[10] J. Y. Song, G. K. Teng, L. X. Hu. *Route Planning Algorithm and Use in Vehicle Location and Navigation System*[J]. Computer & Digital Engineering, 38(8):95. (2010)(in Chinese)

[11] F. B. Zhang, C. E. Noon. *A comparsion between label setting and label correcting algorithms for computing one-to-one shortest paths*[J]. Journal of Geographic Information and Decision Analysis, 4(2):1-13(2000).

[12] Montemerlo M, Becker J, Bhat S, et al. Junior: *The Standford entry in the urban challenge*[J]. Journal of Field Robotics, 25(9): 569-597(2008).