Research of Improved Shortest Path Algorithm in Campus GIS

Xiaohui Zhang[1,*] Xiaoyan Guo[1], Gu Jing[2] and Houliang Xie[3]

[1]Information Engineering Department, Yellow River Conservancy Technical Institute, Kaifeng 4750004; [2]School of software, Central South University, Hunan 410004; [3]Information Engineering Department, Zhangjiajie Institute of Aeronautical Engineering, Zhangjiajie 427000

Abstract: In searching of campus geographic information system, the shortest path is always the key and its efficiency which determines the quality of the service of the digital campus directly. In the classic algorithm, the time complexity is proportional to the vertex number's square. With the increasing of the vertex number, the speed will fell down sharply. Based on the reality such as the large number of population, plenty of buildings, teaching resources, this article proposed an improved bi-directional A* algorithm which changing the target point into surface in search direction by using the middle list searching the target bi-directionally. The experimental results show that, compared with Dijkstra and A* algorithm, bi-directional A* algorithm is the fastest in the search, even when the vertex number large, it also can compute well.

Keywords: Campus GIS, search efficiency, the shortest path algorithm, time complexity.

1. INTRODUCTION

GIS is Geographic Information System [1, 2] (Geographic Information System, or Geo- Information System). It is a computerized data management system [3-5] used to capture, store, manage, retrieve, analyze, and display spatial information and it is also an important part of "digital earth network". In searching of campus geographic information system, the shortest path is always the key and its efficiency which determines the quality of the service of the digital campus directly. Based on the number of nodes is large, this paper studies how to improve the efficiency of node space, achieve the goal of finding the shortest path and make a practical application in digital campus.

2. RELATED THEORY FOR THE SHORTEST PATH ALGORITHM

2.1. Dijkstra Algorithm

Dijkstra algorithm [6, 7] mainly solves the problem of finding the shortest path from a point in a graph (the source) to a destination in the directed graph G = (V, E). This one runs the fastest among three classical Dijkstra, the Bellman – For [7], Floyd – Warshall [7]. For it set up a vertex set S, the final weight of the shortest path from the source points to the vertex in the S set have been determined. The algorithm repeatedly chooses the shortest path to estimate the peak of the u - S $\in$ V, and put u to join the rest of the S, for u all the edge one by one to visit (also known as relaxation technique called Relax [7]). At present, as a classical Dijkstra algorithm, have improved different [8], and the K shortest path algorithm is appeared [9].

2.2. A* Algorithm

The traditional A* algorithm [10] is an advanced Dijkstra version, namely, on the basis of Dijkstra it is added in the heuristic search which helps calculate the evaluation function F of all the extensible vertex currently when it extends to the next vertex, however it will choose the superb rather than the adjacent vertex whose path weight is the minimum to extend.

A* algorithm includes two lists: open list and close list. The former store those vertex will be accessed; the latter store those vertex had been accessed already and no longer accessed again. In A*algorithm, F=G+H is the evaluation function of extensional, G means the distance from the former vertex to the current one, also can be considered as the path weight; H means the estimation of distance that is from the current vertex to the destination one.

H is called A* algorithm's heuristic function. At present, there are many kinds of calculations methods about H function, for based on GIS, we using the Manhattan calculating, which calculate the difference between two points of abscissa and ordinate, then add together. Because the shortest distance between two points is always a straight line. Thus we speculate that, the H function will compute a minimum value, so the linear distance will be priority, which conform to the practical demand of GIS search.

3. THE IMPROVED SHORTEST PATH ALGORITHM OF GIS

3.1. Bi-Directional A* Algorithm

3.1.1. Main Idea of bi-Directional A Algorithm*

Considering the map entity data is more and more accurate in the GIS, that is in improving the accuracy of the data at the same time will increase the storage of data. For improve the efficiency of the algorithm further and provide search service better, bi-directional A* algorithm is proposed in this paper.

Bi-directional A* algorithm is based on A* but it has two distinctiveness: (1) A* algorithm search direction is single, from the starting point to the target point. Bi-directional A* algorithm can range from starting point to the target point or from the target point to the starting point; (2) the direction of the goal of A* algorithm is only target, the single point of one direction, and the direction of the bi-directional A* can include goals around the point, that is, A plane direction.

3.1.2. Design of Bi-Directional A Algorithm*

A* algorithm contains two pairs of lists that respectively are open (positive) and close (positive) by forward direction search and open (negative) and close (negative) by backwards direction search. In addition, A* can't simply alternates from start point to the target point and the target point to the starting point of the search. To solve this problem, the paper puts forward this, if the vertex of forward searching extension are closer to the goal one, positive search is executed, otherwise, the reverse search is executed.

Below are the Bi-directional A * algorithm:

1) Clear the two list that is open list and close list. Add the starting point to the open positive list, add target point negative list.
2) Detection whether close positive and close negative two list have any intersection, if " yes " then turn to (3); or go to (4).
3) In the intersection of close positive and the close negative, choose the direction of the positive and negative two functions of F value added the smallest vertices, which as the intersection of the forward and reverse search path of the output with the corresponding weights (*i.e.*, the value of the G function), return success.
4) Check open positive and open negative, if one is empty, then end of the algorithm, and then return inaccessible information of the starting point to target point;or go to (5).
5) Comparison of forward and reverse search, specific is as follows: to compare the function F value of the open positive and open negative for leftmost node in the middle-order, if the value of H positive is greater then H negative, then take positive search, otherwise take the reverse search.
6) Search forward (search reverse). Delete the leftmost node Vz of open positive list by middle-order, and put it to close positive list (close negative list). Expand Vz adjacent vertices, calculate the corresponding function values F, one at a time to join the open positive list(open negative list) (among them, the calculating process of F function is given in the subsequent), turn to (2).

The process of H function is given in the following:

1) To calculate the G function value of vertex Vp which need to insert in the open positive(negative) list, among them, G is the path weight of Vp to its father vertex Vz.
2) To calculate the G function value of vertex Vp which need to insert in the open positive(negative) list: list all the vertices of close negative list(close positive list), calculated the Manhattan distance of Vp to Vq, that is H = | Xp - Xq | | + Yp - Yq |.
3) F = G + H.

Above is the specific process of bi-directional A * algorithm. In contrast to traditional Dijkstra and A * algorithm, the process of bi-directional A * algorithm is relatively complex, but the bi-directional A * can effectively reduce the search area, especially in some cases need to bypass path, bi-directional A * more advantage.

*3.1.3. The pseudo-code of bi-Directional A**

The following is the pseudo-code of the bi-directional A * algorithm(G is the graph input, s initial vertex, t represents the target vertices, F evaluation function):

```
ShortestPath_DoubleAStar(G,s,t,F)
(1) Initialize_Source(G,s)
(2) close positive←∅; close negative←∅
(3) Insert(open positive,s); Insert(open negative,t)
(4) while close positive∩close negative = ∅
(5) do if open positive =∅ or open negative =∅
(6)         return FALSE
(7)     else
(8)         u1=Extract_Min(open positive,F); u2=Extract_Min(open negative,F)
(9)         if H(u1) > H(u2) then    // forward search
(10)                do Insert(close positive,u1)
(11)                    for each vertex v∈Adj[u1]
(12)                        do Relax(u1,v,F)
(13)            else    //reverse search
(14)                do Insert(close negative,u2)
(15)                    for each vertex v∈Adj[u2]
(16)                        do Relax(u2,v,F)
(17) u←Extract_Min(close positive∩close negative,F+F')// The best intersection point
(18) GetPathValue(u)
(19) ReturnPath(u,G+G')
(20) return TRUE
```

4. EXPERIMENTAL EVALUATION

4.1. Experiment Setting

Considering the bi-directional A* algorithm could be used in A variety of computer equipment, this article will choose two configuration of computer hardware for experimental evaluation, specific configurations are as follows:

Table 1. Three algorithms comparison for total run time.

Algorithm	Configuration 1(ms)	Configuration 2(ms)
Dijkstra	-	2690033
A*	809755	801688
Bi-directional A*	756819	738896

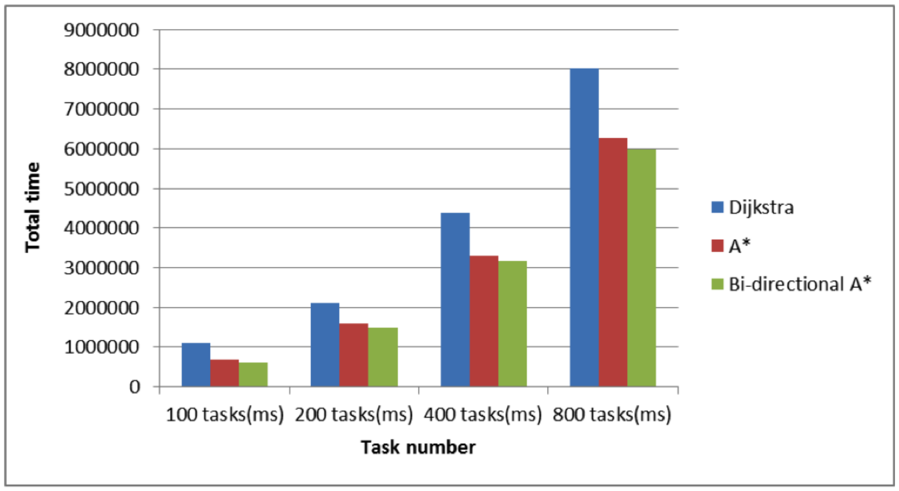

Fig. (2). Total time comparison of multitasking.

Configuration 1:

CPU: Intel Core i3-3110M 2.4GHz

Memory: 4G DDR3 1333MHz

Disk: WDC SATA 500G

Configuration 2:

CPU: Intel Xeon E5-2600 ×2 2.0GHz 8cores/CPU 32Mbcache/CPU

Memory: 8G×8 DDR3 1600MHz

Disk: WDC SATA 1T

Using Windows 7 development environment for Visual Studio 2012 [11], ArcGIS10, development language is c++ [12]. In this paper, experimental data are provided by the campus GIS, in order to facilitate programming, GIS data structure using the grid structure. In addition, this experiment will evaluate traditional Dijkstra and A*, bi-directional A* three algorithms. To simulate the multiplayer online request application environment at the same time, the experiment is divided into single thread and multithreading concurrent.

4.2. Experimental Results

4.2.1. Single Thread Results of the Three Algorithms

In order to highlight three algorithms running time gap, grid size for practical ground is 1 cm x 1 cm, each algorithm performs 10 groups of test data, and accumulate the results after the time, the results are shown as in Table **1**.

Experimental results show that as for the total run time of the 10 groups grid data, bi-directional A* algorithm is the best, traditional A* is the second, the worst one is Dijkstra algorithm. The reason why Dijkstra algorithm runs slowest is its searching with no direction, and too much intermediate points are produced. In configuration 1, due to Dijkstra algorithm need a lot of memory for running the 10 groups grid data, lead to the collapse of the computer system. Therefore, the sysbol "-" is used to express. Traditional A * algorithm runs faster because it has directivity. As for bi-directional A* algorithm, the search is not only bi-directional but also make the goal direction to be a surface, so it is the best of them.

4.2.2. Multithreading Concurrent Results of the Three Algorithms

Because the memory leak in Dijkstra algorithm, in the multitasking test, the grid size switch to the actual ground 50 cm by 50 cm. To evaluate the three algorithms, the each "setting" machine executes multitasking tasks four times, that is: 100, 200, 400, 800; each task is also 10 groups. Multitasking research requests made by another terminal machine with the local laboratory network, time interval of sending between each task is 100 ms, terminal machine will end it and service machine make the undisposed requests line up, when receiving the final commanding, it will record the total running time, and returns the result. The results of configuration 1 are shown in Fig. (**2**).

(一) The running results of configuration 1:

The total time consumed for multitasking by all the three algorithm increase with the increase of the number of tasks. In the time consumption of multitasking, bi-directional A* algorithm is best; traditional A* comes the second; Dijkstra algorithm is the worst. Traditional A* algorithm is better than the Dijkstra algorithm for the reason is that the middle

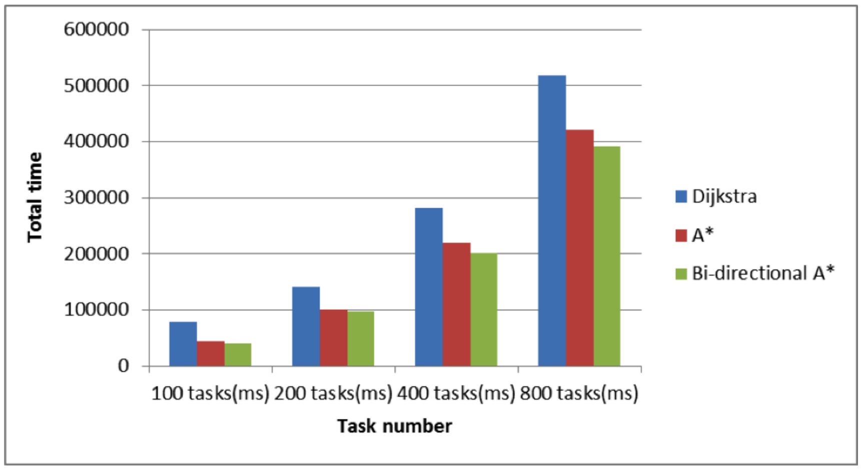

Fig. (3). Total time comparison of multitasking.

of the directional search and produce less points. Bi-directional A* algorithm is the best. The reason is about two-way characteristic and shape the direction of the search.

(二) The running results of configuration 2:

The total time consumed for multitasking by all the three algorithm increase with the increase of the number of tasks. In the time consumption of multitasking, bi-directional A* algorithm is best; traditional A* comes the second; Dijkstra algorithm is the worst. Traditional A* algorithm is better than the Dijkstra algorithm for the reason this is that the middle of the directional search and produce less points. Bi-directional A* algorithm is the best. The reason is about two-way characteristic and shape the direction of the search.

5. SUMMARY

This experiment used two different configuration of the machine to run the test about the three algorithms of single task and multi tasks. The results show that bi-directional A* algorithm is superior to traditional Dijkstra and A * algorithm.

Search path is the most important and the most commonly used function of GIS. With the popularity of electronic map, there will be more people using GIS path to search capabilities in the future. And bi-directional A * algorithm is an improvement of traditional A* algorithm. The algorithm can effectively avoid the search process into the "dead end" leading to too much time consumption.

CONFLICT OF INTEREST

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

ACKNOWLEDGEMENTS

Declared none.

REFERENCES

[1] Feizizadeh B, Jankowski P, Blaschke T. A GIS based spatially-explicit sensitivity and uncertainty analysis approach for multi-criteria decision analysis[J]. Computers & Geosciences, 2014, 64(1):81–95.

[2] Ghosh D, Ghose T, Mohanta D K. Reliability analysis of geographic information system (GIS) aided optimal phasor measurement unit location for smart grid operation[J]. Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability, 2013, 23(5): 06-13.

[3] Pradhan B. A comparative study on the predictive ability of the decision tree, support vector machine and neuro-fuzzy models in landslide susceptibility mapping using GIS[J]. Computers & Geosciences, 2013, 51(10): 350-365.

[4] Jakubiec J A, Reinhart C F. A method for predicting city-wide electricity gains from photovoltaic panels based on LiDAR and GIS data combined with hourly Daysim simulations[J]. Solar Energy, 2013, 93(8): 127-143.

[5] Devkota K C, Regmi A D, Pourghasemi H R, et al. Landslide susceptibility mapping using certainty factor, index of entropy and logistic regression models in GIS and their comparison at Mugling–Narayanghat road section in Nepal Himalaya[J]. Natural Hazards, 2013, 65(1): 135-165.

[6] Wagner R B D D P S D S D. Combining hierarchical and goal-directed speed-up techniques for Dijkstra's algorithm[J]. Proceedings of Workshop on Experimental Algorithms Wea', 2010,43(6):303-318.

[7] Sedgewick R, Flajolet P. An introduction to the analysis of algorithms[M]. Addison-Wesley, 2013.

[8] Dong Jun and Huang Chuan-he. Research on Shortest Path Search of Improved Dijkstra Algorithm in GIS Navigation Application [J]. Computer Science,2012,10(2):245-257.

[9] Mao Shao-wu, Zhang Huan-guo and Huang Chong-chao. A new fault-tolerance mechanism in communications based on K shortest path algorithm[J]. Wuhan Univ, 2013,16(06):534-538.

[10] Bell M G H. Hyperstar: A multi-path Astar algorithm for risk averse vehicle navigation[J]. Transportation Research Part B: Methodological, 2009, 43(1):97–107.

[11] Johnson B. Professional Visual Studio 2012[M]. John Wiley & Sons, 2012.

[12] Meyers S. Effective C++: specific ways to improve your programs and designs[M]. Pearson Education, 2005.

Received: June 10, 2015 Revised: July 29, 2015 Accepted: August 15, 2015