Heuristic Pathfinding Algorithm Based on Dijkstra

Yan-Jiang SUN[1, a], Xiang-Qian DING[2, b], Lei-Na JIANG[3, c]
[1, 2, 3] College of Information Science and Engineering
Ocean University of China, Qingdao, Shandong, China
[a]pswqdx616237781@163.com, [b]Dingxq1995@vip.sina.com, [c]jiangleina520@163.com
*Leina Jiang

Keywords: Dijkstra algorithm, shortest path, small heap, passing point, heuristic, pathfinding

Abstract. The problem of shortest path solution belongs to the classical algorithm problem, the Dijkstra algorithm has wide research and application, but there are still some shortcomings in practical applications. First of all, in order to solve the problem of low efficiency of Dijkstra algorithm, this paper adopts the way of small heap, it improves the efficiency and makes the time complexity reduce to O (nlogn); secondly, since Dijkstra is a single source shortest path algorithm, and cannot be better to solve the problem that the path has passing points, so, this paper presents a heuristic path-finding strategy based on the improved Dijkstra algorithm. Under the restriction of passing points, with the distance between the source point and passing point as a elicitation, this paper solves the shortest path problem from the starting point to finishing point; in the end, this paper verify the effectiveness of this pathfinding strategy by experimental results.

Introduction

The problem of calculating route belongs to the basic algorithm problem, the study of such problem is very important. Optimizing and dealing with the shortest path that encounter in the work can greatly improve the efficiency of work. For example, in the sequencing of craftwork, connects the various of craftwork processes so that makes the assembly work well and quickly; in the traffic network diagram, it is very important to reasonable select the delivery path, it can make the material transfer work to spend less time but more in quantity; however, in the problem of network calculating route, it is also important to propose an optimized pathfinding strategy to achieve efficient utilization of network resources.

With the arrival of the wave of science and technology, the number of nodes in the network rise sharply. In the process of network analysis, the storage and calculation of massive network nodes are the key of network analysis [7], and the optimization of the shortest path algorithm will become an important part of system performance improvement. In the algorithm of solving the shortest path, the Dijkstra algorithm has good stability and effectiveness. In order to reduce the time complexity and space complexity of the Dijkstra algorithm and improve the efficiency of the algorithm, document[1] optimizes the Dijkstra algorithm's exit mechanism and improves the Dijkstra's label algorithm; document[2] optimizes the Dijkstra algorithm by adopting binary tree architecture; document[3] improves the problem of the multiple adjacent contact and multiple shortest path with Dijkstra algorithm; document[4] uses the adjacent tables to store nodes and adopts the heap sorting to sort the nodes by William's proposed.

Therefore, this paper uses the adjacent tables and small heap to optimize the time and space complexity of the Dijkstra algorithm, it achieves the search of shortest path in the network. Under the restriction of passing points, this paper proposes a heuristic pathfinding strategy. This strategy better solves the multiple passing points in the network, simplifies the solution of the shortest path, the time and space complexity are close to the improved Dijkstra algorithm.

Basic concept

The graphs have a wide application in life, for example: the common traffic diagram, circuit diagram, and various flow charts. On the basis of graph theory, vertices and edges in graphs are transformed into nodes and arcs in network data structures. Obviously, the nodes existing between arcs and arcs establish a network with topological characteristics. The topological structure of the network, uses a mathematical method to define the relationship of spatial structure. The spatial structure, on the one hand, it integrates the spatial data; on the other hand, it also has important scientific significance for spatial analysis and Application.

Defining directed graph S= (K, L), K is the vertex set, L is the directed edge set, there is a weight between the directed edges of vertices. Giving the two vertexes m, n, and the k's subset k', finding the non-ring directed path T from m to n in the graph, and makes the directed path pass k' all vertexes (For k' node's order, there is no special requirement).

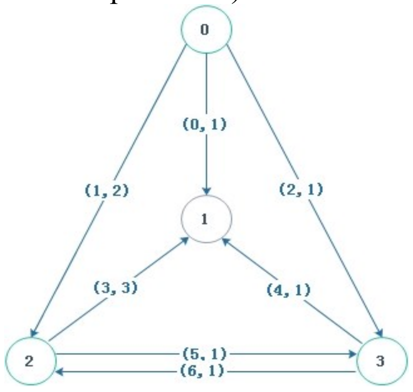

Fig. 1. Direct graph

Supposing a directed graph, like figure. 1, the 0,1,2,3 represents the nodes, (1,2) respectively presents the number and weight of arc, directed graphs are transformed into the network data structure, it will get the following directed digraph information: (0, 0, 1, 1), (1, 0, 2, 2), (2, 0, 3, 1), (3, 2, 1, 3), (4, 3, 1, 1), (5, 2, 3, 1), (6, 3, 2, 1), the four numbers respectively indicate the number, starting point, finishing point, weight. From the graph, finding the directed path that 0 as the starting point and 1 as the finishing point, the vertices that must pass 2 and 3, the path information are as followed:

0, 1, 2 | 3

For this test case, there are two possible paths as follows:

2|6|3 the path meaning: Start at the edge of number 2, and go to No. 3 by route 6.

1|5|4 the path meaning: Start at the edge of number 1 and go to No. 4 by route 5.

Owing to the first path's weight is 5, the second path's weight is 4, so the optimal solution should be 1|5|4.

Dijkstra algorithm

Algorithm definition. Dijkstra algorithm is well known as the single source shortest path algorithm, it is used to find the shortest path from one node to all the other nodes. The main idea is putting the nodes as the center starting point to expand, until all the nodes are traversed and the shortest path is found [6]. As the classic shortest path algorithm, the Dijkstra algorithm requires the graph not contain negative weights edges.

The steps of algorithm.

Dijkstra Algorithm:

```
1.  Init: v as source point, S = {v}, U = {other vertices};
2.  Calculate: the distance from v to the other vertices, dist[i] = map[v][i];
3.  Add: find the nearest point t and add t to the set S,
        if (!visit[t] && dist[t]<min)
            min = dist[t];
4.  Relax: modify the vertex distance,
        if (!visit[u] && dist[u]>dist[t]+tab[t][u] )
            dist[u] = dist[t]+tab[t][u];
5.  Repeat steps 3 and 4 until all vertices are included in S.
```

Pathfinding strategy based on the improved Dijkstra algorithm

The improvement of Dijkstra algorithm. The classic Dijkstra algorithm, combines the greedy strategy and the dynamic programming method to do the single source shortest path search work. When greedy algorithm is used to find the shortest node from the source node, it is necessary to traverse all the nodes in the graph, which will inevitably reduce the efficiency of the algorithm, and the time complexity will reach O (n2). Therefore, when the number of nodes in the network is very large, the operation time of the algorithm will be longer. In order to increase the efficient of Dijkstra, on the basis of the greedy strategy, this paper improves the Dijkstra algorithm.

Since the greedy strategy used in the Dijkstra algorithm looks for the shortest node from the current node and stores it in the S collection, the set contains all the nodes of the shortest distance. So this paper uses the data structure that is similar to priority queues to provide the shortest node. Practice has proved that using efficient heap data structure as priority queue is very effective for improving the efficiency of Dijkstra, and at the same time, heap sort algorithm possesses the advantages of insert sort and merge sort, it can make the sort to have a better time complexity O (nlogn).

- This paper uses the small heap to maintain the Dist array in the Dijkstra algorithm, the vertexes in the heap are divided into two parts. The vertices containing the passing points in the adjacency point have a high priority, and when the number of the passing points are more, the priority is much higher; the other vertexes belong to the low priority, first of all, sorting by priority, and then sorting by distance [5]. Owing to select the heap top element each time, so it increases the reasonable of heap sort and reduces the time complexity.
- Using adjacent tables instead of adjacent matrix, this reduces the storage space of Dijkstra algorithm and makes the space complexity reduce from O (n2) to O (n+e), it also improves the space utilization of the algorithm.

Pathfinding strategy. In order to improve the execution efficiency of pathfinding strategy, this paper marks the vertex that have passed during the pathfinding process. In the next search process about path, if it arises the marked vertex, we will select the adjacent nodes once again. This approach greatly improves the speed of pathfinding and avoids the path's search to fall into the dead loop. The improved Dijkstra algorithm adopts the following strategy to search the shortest path:

1) The starting point as the source point, the finishing point as the last passing point, at first, array T and stack S is empty;

2) According to the Dijkstra algorithm, this paper first traverses all the vertexes, then the source points and the passing points are sorted by the distance, in the end the neartest distance is in the front;

3) If the vertex has already appeared in the array T when source point arrives at the nearest passing point, we will select the second passing point as the nearest passing point, and repeat the step until there is no vertex in the T; If the passing point has not been found after all attempts have been completed, the source point is removed from the S stack, and if the element of the stack is empty, the loop will stop and unanswered, at the same time, the vertex from the last source point to this source point in the T will be deleted; we will select the second nearest passing point and repeat the step3;

4) The vertex stores in the array T when source point arrives at the nearest passing point, the nearest passing point is pushed into the stack S, we select the nearest passing point as the source point, repeat step 2 and step 3, if the passing points are included into the path, jump to the loop and find the optimum solution.

Obviously, if there is an optimum solution in the vertex set, the pathfinding strategy will find it first time. This not only saves the time, but also the time and space complexity are close to the improved Dijkstra algorithm.

Experimental results and analysis

This paper verifies the efficiency and accuracy by the experiment, the test cases and environment are followed: 1) the edge weight is the number from 1 to 20; 2) the vertexes of graph are no more than 600, the out-degree is no more man 8 in each vertex; 3) the number of passing points are no more than 50; 4) finding the directed path of no ring from the starting point to destination point, it must pass the passing points. If we can find the directed path, the optimal path will be output; if we cannot find, output without solution; 5) this paper uses the C++ and selects the Microsoft Visual Studio 2017 to test. The experiment results are followed:

Table1. Test Cases

Test Cases	Correct Cases	Correct Rate
750	750	100%

In Table1. Test Cases, this paper select 750 test cases, it includes from easy to complex, the biggest vertexes is 600, the biggest directed edges is 4367, the biggest passing points set 50. Finally, the accuracy of test results is up to 100%. This paper research and analyze the test cases and get the relation between passing points, vertexes and the time.

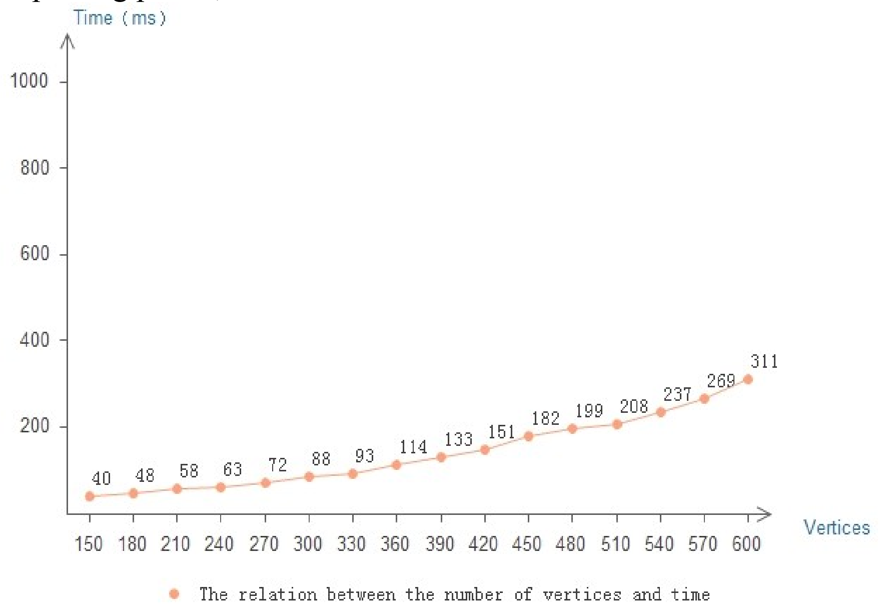

Fig. 2. Vertices and time

- Passing points set 20, this paper select the 10 group's test cases to test each vertex, and get the average time after the completion of the path search, then we get the relation graph about vertexes and time. From the above graph, it is found that the ascending trend of the line chart is slow. That is to say, the influence of the number of vertices on time is small.

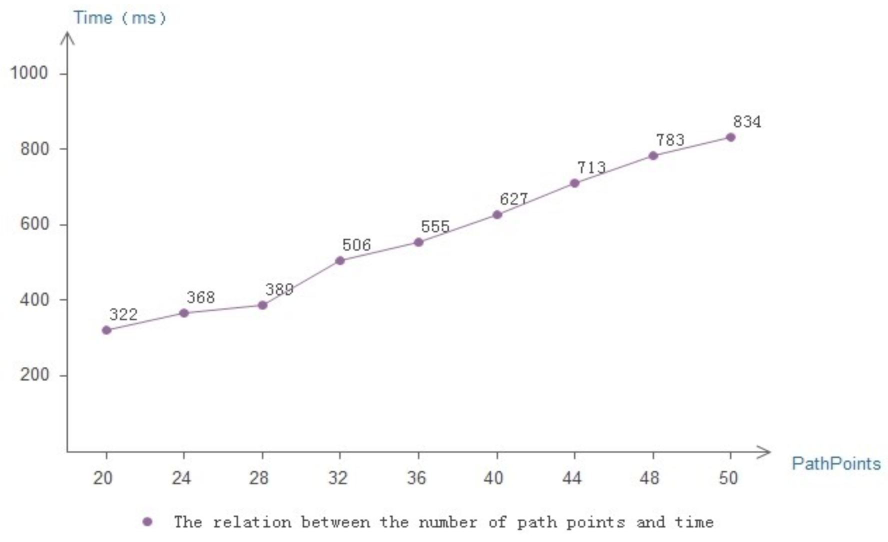

Fig. 3. Path point and time

- The vertexes set 600, this paper also select the 10 group's test cases to test each passing point, and get the average time after the completion of the path search, then we get the relation graph about passing point and time. From the above graph, comparing with the relation between the vertices and time, the number of passing points varies little, but the ascending trend of the line chart is fast and the time grows faster. That is to say, the influence of the number of passing points on time is big.

In general, the pathfinding strategy achieves better results for path searching. This paper add the restriction of passing points during the process, finally, we use the less time and 100% accuracy to complete the shortest path search and provide a feasible direction for the research of this kind of problem.

Conclusion

This paper analyzes and researches the network topology structure and proposes an improved Dijkstra heuristic pathfinding algorithm. Because the Dijkstra algorithm's core concept adopts the greed strategy and makes the efficiency lower, so the method of efficient heap optimization is proposed, and the path searching speed of Dijkstra algorithm is improved better; this paper selects the path by the way of heuristic, and researches the multiple intermediate points pathfinding algorithm, in the end , this thesis provides a simple and efficient way, it achieves the goal of finding the shortest path in a relatively short time and the accuracy of shortest path is up to 100%. At present, this pathfinding method has a great dependence on the number of intermediate points, and how to reduce the influence of the intermediate point on the algorithm is the future research direction of this paper.

References

[1]Wang S X. The Improved Dijkstra's Shortest Path Algorithm and Its Application [J]. Procedia Engineering, 2012, 29:1186-1190.

[2]Yuanchen L I, Liu W. Analysis of the Shortest Route in Network on Dijkstra Algorithm [J]. Microcomputer Applications, 2004.

[3]Wang S X, An-Yu L I. Multi-adjacent-vertexes and Multi-shortest-paths Problem of Dijkstra Algorithm [J]. Computer Science, 2014.

[4]Wang Z H. Analysis and Improvement of Dijkstra Algorithm [J]. Journal of Hubei University of Education, 2008.

[5]Ming B P. AN OPTIMIZATION ALGORITHM BASED ON DIJKSTRA'S ALGORITHM IN SEARCH OF SHORTCUT [J]. Journal of Computer Research & Development, 2001.

[6]Lian-Fa S I, Wang W J. Realization of Optimal Algorithm for Fast Dijkstra Latest Path [J]. Bulletin of Surveying & Mapping, 2005.

[7]Yue Y. An Efficient Implementation of Shortest Path Algorithm Based on Dijkstra Algorithm [J]. Journal of Wuhan Technical University of Surveying & Mapping, 1999.