A Modified Dijkstra's Algorithm for Solving the Problem of Finding the Maximum Load Path

Kaicong Wei, Ying Gao, Wei Zhang, Sheng Lin
School of Computer Science and Educational Software
Guangzhou University
Guangzhou, China
e-mail: weikaicong@sina.cn

Abstract—**Path optimization is especially useful for improving freight efficiency, and the problem of finding the maximum load path is an important issue in path optimization. Although the problem of finding the shortest path is similar to the problem of finding the maximum load path, the Dijkstra's algorithm for solving the problem of finding the shortest path is not suitable for solving the problem of finding the maximum load path. In this paper, the Dijkstra's algorithm for solving the problem of finding the shortest path is introduced, and the Dijkstra's algorithm is modified to solve the problem of finding the maximum load path. Finally, the running process of the modified Dijkstra's algorithm is described by an example.**

Keywords-path optimization; maximum load path; Dijkstra's algorithm

I. INTRODUCTION

With the rapid development of e-commerce and road freight, the efficiency of road freight plays a crucial role in the profitability of trade between countries. Therefore, the requirement for improving the efficiency of road freight is imminent. Since the vehicle has strict load restrictions on the road, once it enters the restricted road section, it does not only break the rules and needs to pay a fine, but it may also pose a safety risk. Therefore, before the vehicle goes on the road, it is necessary to plan the route first and select a path that can satisfy the limitation of the road weight limit and maximize the load capacity of the vehicle, so that the travel is more intelligent and safe.

In the path planning, its requirement can be either a "bottleneck" constraint, such as the travel distance constraint or the path load limited constraint, or an "additive" constraint, such as cost and time constraints. But constructing a path subject to more than one additive constraint is NP-hard. Therefore, many researchers have turned their attention to designing either heuristic algorithms or approximation algorithms for the path planning problems, such as in terms of ant colony algorithm, Li et al. improve ant colony algorithm so that optimal vehicle route could be reselected according to actual situation [1], Yu et al. comprehensively consider the physical route distance and road condition, collect and draw the geographic information, optimize logistics terminal distribution path based on ant colony algorithm[2]. In terms of simulated annealing algorithm, Yiyi et al. consider many factors such as distance, cost and time in the transportation route, a multi-objective TSP optimization model is established, and the simulated annealing algorithm is used to solve the optimal solution [3], WU et al. design an improved simulated annealing algorithm to solve the vehicle routing problem with following three constraints: customer demand, maximum load and maximum distance of vehicles [4]. In terms of genetic algorithm, LUO et al. solve the path optimization problem under multi-constraint condition by improved genetic algorithm [5], Han et al. design an improved genetic algorithm to optimize distribution routing according to the actual operation and orders characteristics of e-commerce [6]. In terms of particle swarm optimization algorithm, Chen Qun apply Particle Swarm Optimization algorithm to path searching of dynamic route guidance [7], Zhang et al. construct a multiobjective model which considers many factors, such as the vehicle speed, load and road conditions, and use an improved particle swarm optimization algorithm to solve the model simulations in order to achieve a reasonable distribution route planning[8].

However, the heuristic algorithm or the approximation algorithm can only find the optimal solution or the approximate optimal solution within an acceptable time range, and cannot find the global exact solution. At the same time, the study has many sub-problems need to be solved, such as the problem of finding the maximum load path, its requirement is a "bottleneck" constraint.

In the path planning, when the requirement is a "bottleneck" constraint, the global search method can be used to accurately find the optimal solution. The problem of finding the shortest path is especially classic, and the Dijkstra's algorithm is a typical solution to the problem of finding the shortest path [9]. Given the starting and destination locations, the problem of finding a path that minimizes the travel time, or minimizes the travel distance, or minimizes the travel cost, can be equated with the problem of finding the shortest path, in other words, it can be solved by using the Dijkstra's algorithm. Although the problem of finding the maximum load path is similar to the problem of finding the shortest path, the Dijkstra's algorithm can not effectively solve the problem of finding the maximum load path. So the problem of finding the maximum load path is worth studying.

In this paper, a modified algorithm that based on the Dijkstra's algorithm is presented, and this modified algorithm is applied to the problem of finding the maximum load path.

978-1-7281-3323-2/19/$31.00 ©2019 IEEE

II. THE PROBLEM OF FINDING THE MAXIMUM LOAD PATH

Now there is a road map that indicates the road load limit between each pair of adjacent road intersections, how to find a path to maximize the load capacity of the vehicle according to the starting and destination locations? This problem can be modeled as a directed graph: let a road map be represented by a directed graph G = (V, E), where V is the set of nodes in G, representing the intersection of roads, and E is the set of links in G, representing every road. Let V = {1,2,3,...,n}, E = {e|e = [i,j],i,j∈V}. Each link e = [u,v] in E is associated with the limit load of the vehicle on the road corresponding to the edge [u,v] (referred to the limit load) w(e) = w([u,v]), and w(e) > 0. Let P = $\{v_1, v_2, ..., v_r\}$ be a path in G, where v_i, i = 1,2,….,r, are nodes in G and $[v_i, v_{i+1}]$, i = 1,2,….,r-1 are edges in G. The limit load of the path P is defined to be min{ w($[v_i, v_{i+1}]$) | 1 ⩽ i ⩽ r − 1}.

III. DIJKSTRA'S ALGORITHM AND ITS MODIFIED

Among lots of algorithms for solving the problem of finding the shortest path, Dijkstra's algorithm is the most popular algorithm, and many algorithms are modified based on the algorithm[10-14].

The pseudo-code of the Dijkstra's algorithm is shown in Fig. 1.

```
Algorithm. Dijkstra
Input: a graph G, a source vertex s and a destination
vertex t
Output: a path from s to t with the minimum weight
1. for each vertex v do
    { status[v]=0; wt[v]=-1; dad[v]=-1; }
2. status[s]=2; wt[s]=+∞;
3. for each edge [s, w] do
    { status[w]=1; wt[w]=weight(s,w); dad[w]=s; }
4. while there are fringes do
       v = the fringe with the min wt-value;
       status[v]=2;
       for each edge[v, w] do
          case 1. status[w]==0:
                { status[w]=1; wt[w]=wt[v]+weight(v, w);
dad[w]=v; }
          case 2. (status[w]==1) and
(wt[w]>(wt[v]+weight(v, w)):
                { wt[w]= wt[v]+weight(v, w);
dad[w]=v; }
```

Figure 1. The pseudo-code of the Dijkstra's algorithm.

During the running process of the algorithm, each node will experience three status, namely 0 (unseen), 1 (fringe), and 2 (in-tree). Unseen is the initial status, indicating that the algorithm has not yet accessed the node; fringe is the intermediate status, indicating that the weight of the node and the parent node of the node may be updated; in-tree is the final status, indicating the weight of the node and the parent node of the node have been determined.

The algorithm uses three arrays, status, wt, and dad. The array element status[v] records the state of the node v, indicating what operations the algorithm can perform on the node v; the array element wt[v] records the weight of the node v, indicating that it goes from the starting location to the node v, the weight of the path is summed; the array element dad[v] records the parent node of the node v, indicating from which node to go to the node v.

During the running process of the algorithm, the set of nodes whose state is fringe is stored with the smallest heap to ensure that the node with the lowest weight is gained and the operations such as insert and delete are completed in O(log n) time.

When the running of the algorithm is done, the parent node is searched according to the destination node, and back to the starting location, the shortest path from the starting location to the destination location can be output.

The first step of Dijkstra's algorithm takes O(n) time, the second step takes O(1) time, the third step takes O(n) time, and the fourth step takes O((n+m)logn) time, so Dijkstra's algorithm totally takes O((n+m)logn) time, where n is the number of nodes in graph G and m is the number of edges in graph G.

According to the starting location s and the destination location t, the shortest path can be found in O((n+m)logn) time by the Dijkstra's algorithm. But the problem of finding the maximum load path is not equivalent to the problem of finding the shortest path, so it is necessary to modify the Dijkstra's algorithm so that the problem of finding the maximum load path can be effectively solved.

In the problem of finding the shortest path, whether the shortest path can be found depends on whether a path with total weight is minimum can be found. Whereas in the problem of finding the maximum load path, the maximum load of a path subjects to the minimum load of the path, so whether a maximum load path can be found depends on whether the minimum weight of the path can be maximized.

In Dijkstra's algorithm, the weight represents distance, time, cost, etc. In the modified Dijkstra's algorithm, the weight represents only the load capacity of a vehicle limited by a road.

The pseudo-code of the modified Dijkstra's algorithm is shown in Fig. 2.

The difference from Dijkstra's algorithm is that in the modified Dijkstra's algorithm, the array element wt[v] records the weight of the node v which is the smallest of the weights from the starting location to the node v.

In Dijkstra's algorithm, the node in the status of fringe updated is conditional on the new total weight is smaller than the current total weight of the node. In the modified Dijkstra's algorithm, the node in the status of fringe updated is conditional on the weight is bigger than the existing weight of the node. So, once the node is updated, it means that the limit load of the path from starting location to the node is increased, in other words, the upper limit of load capacity that the path allowed is increased.

Algorithm. Modified Dijkstra
Input: a graph G, a source vertex s and a destination vertex t
Output: a path from s to t with the maximum load
1. **for** each vertex v **do**
{ status[v]=0; wt[v]=-1; dad[v]=-1; }
2. status[s]=2; wt[s]=+∞;
3. **for** each edge [s, w] **do**
{ status[w]=1; wt[w]=weight(s,w); dad[w]=s; }
4. **while** there are fringes **do**
v = the fringe with the max wt-value;
status[v]=2;
for each edge[v, w] **do**
case 1. status[w]==0:
{ status[w]=1; wt[w]=min{wt[v], weight(v, w)}; dad[w]=v; }
case 2. (status[w]==1) and (wt[w]<min{wt[v], weight(v, w)}):
{ wt[w]=min{wt[v], weight(v, w)}; dad[w]=v; }

Figure 2. " The pseudo-code of the modified Dijkstra's algorithm.

During the running process of the algorithm, the nodes whose state is fringe are stored in a max heap so that each time the node with largest weight is gained in O(log n) time.

When the running of the algorithm is done, the parent node is searched according to the destination node, and back to the starting location, the maximum load path from the starting location to the destination location can be output.

The first step of the modified Dijkstra's algorithm takes O(n) time, the second step takes O(1) time, the third step takes O(n) time, and the fourth step takes O((n+m)logn) time, so the modified Dijkstra's algorithm totally takes O((n+m)logn) time, where n is the number of nodes in graph G and m is the number of edges in graph G.

The running process of the modified Dijkstra's algorithm as follows:

1. Create three arrays and initialize the array elements. The array status represents the status of the node, initialized to 0; the array wt represents the weight of the node, initialized to -1; the array dad represents the parent node of the node, initialized to -1.

2. Let the status of node w to be 2, the weight to +∞.

3. Traverse each edge (s, w) connected to the starting location s, let the status of node w to be 1, the weight to be the weight of the edge (s, w), the parent node to be the starting location s, put the node w into the max heap H respectively.

4. While the max heap H is not empty, get the first element of H that is node v which has the largest weight, let the status of node v to be 2, traverse each edge (s, w) connected to the node v, different operation is performed according to the status of the node w.

4a. If the status is 0, let the status to be 1, the parent node to be node v, the weight to be the smaller of the weight of the node v and the weight of the edge (v, w), put the node w into the max heap H.

4b. If the status is 1 and the weight of the node w is smaller than the smaller of the weight of the node v and the weight of the edge (v, w), then let the weight of the node w to be the smaller.

5. If there has an element in H, then jump to step 4, otherwise, the algorithm is done, the maximum load path from the starting location to any destination location has been found.

IV." EXAMPLE

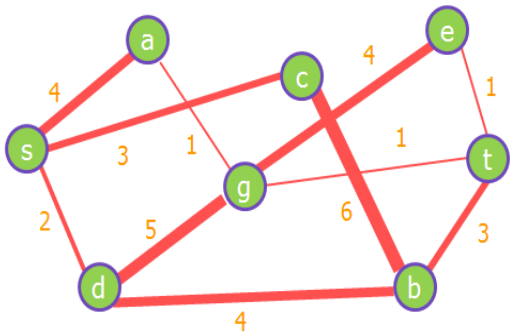

Figure 3. " The modified Dijkstra's algorithm for finding the maximum load path.

As shown in Fig. 3, to find the maximum load path from a to t:

1. Let the status of all nodes to be 0, and the weight on edge represents the load limit of the road.

2. Starting from s, let the status of s to be 2, let the status of a, c and d to be 1; let the weight of a to be the weight of edge (s, a), which is 4, let the weight of c to be the weight of edge (s, c), which is 3, let the weight of d to be the weight of edge (s, d), which is 2; let the parent node of a to be s, indicating that the algorithm goes from s to a, let the parent node of c to be s, indicating that the algorithm goes from s to c, let the parent node of d to be s, indicating that the algorithm goes from s to d.

3. Now the nodes in 1 status (a, c and d), the weight of a is the largest (a is 4, c is 3, d is 2), so let the status of a to be 2. And the status of g is 0, so let the status of g to be 1, compare the weight of node a (is 4) with the weight of edge (a, g) (is 1), the weight of edge (a, g) is smaller, so let the weight of g to be the weight of edge (a, g), let the parent node of the node g to be a, indicating that the algorithm goes from a to g.

4. Now the nodes in 1 status (c, d and g), the weight of c is the largest (c is 3, d is 2, g is 1), so let the status of c to be 2; And the status of b is 0, so let the status of b to be 1, compare the weight of node c (is 3) with the weight of edge (c, b) (is 6), the weight of node c is smaller, so let the weight of b to be the weight of node c, let the parent node of b to be c, indicating that the algorithm goes from c to b.

5. Now the nodes in 1 status (g, d and b), the weight of c is the largest (g is 1, d is 2, b is 3), so let the status of b to be 2; The status of d is 1, and the weight of node d (is 2) is smaller than the weight of edge (b, d) (is 4) and the weight of

node b (is 3), so let the weight of d to be the weight of node b, let the parent node of d to be b, indicating that the algorithm goes from b to d. The status of t is 0, so let the status of t to be 1, compare the weight of node b (is 3) with the weight of edge (b, t) (is 3), the two are equal, so let the weight of t to be the weight of node b, let the parent node of t to be b.

6. Now the nodes in 1 status (g, d and t), the weight of t is the largest (g is 1, d is 2, t is 3), so let the status of t to be 2; The status of g is 1, and the weight of node g (is 1) is not smaller than the weight of edge (t, g) (is 1) and the weight of node t (is 3), so it is maintained. The status of e is 0, so let the status of e to be 1, compare the weight of node t (is 3) with the weight of edge (t, e) (is 1), the weight of node t is smaller, so let the weight of e to be the weight of node t, let the parent node of e to be t, indicating that the algorithm goes from t to e.

7. Now the nodes in 1 status (g, d, and e), the weight of d is the largest (g is 1, d is 2, e is 1), so let the status of d to be 2; The status of g is 1, and the weight of node g (is 1) is smaller than the weight of edge (d, g) (is 5) and the weight of node d (is 2), so let the weight of g to be the weight of node d, let the parent node of g to be d, indicating that the algorithm goes from d to g.

8. Now the nodes in 1 status (g and e), the weight of g is the largest (g is 2, e is 1), so let the status of g to be 2; The status of e is 1, and the weight of node e (is 1) is smaller than the weight of edge (e, g) (is 4) and the weight of node g (is 2), so let the weight of e to be the weight of node g, let the parent node of e to be g, indicating that the algorithm goes from g to e.

9. Now the nodes in 1 status are only the node e, so let the status of e to be 2.

10. A maximum load path from a to t has been found, that is s-c-b-t. The Algorithm is done.

V. CONCLUSION

In the path optimization, the heuristic algorithm or the approximation algorithm can only find the optimal solution or the approximate optimal solution within an acceptable time range, and cannot find the global exact solution. But there are some similarities in the problem of finding the shortest path and the problem of finding the maximum load path, the requirements are all a "bottleneck" constraint, so the global search method can be used to accurately find the optimal solution.

Although the Dijkstra's algorithm for solving the problem of finding the shortest path is not suitable for solving the problem of finding the maximum load path, by analyzing the problem of finding the maximum load path, a modified Dijkstra's algorithm is proposed, and it is an excellent way to solving the problem of finding the maximum load path.

ACKNOWLEDGMENT

This work is supported by Science and Technology Projects of Guangdong Province, China, under Grant Nos. 2016B010127001; by Higher Education Teaching Research and Reform Projects of Guangdong Province, China; by Guangzhou Teaching Achievements Cultivation Project "Exploration and Practice of Collaborative Cultivation of IT Innovative Talents by School and Enterprise in the Integration of Subject Competition".

REFERENCES

[1] Li B , Gu H , Ji Y . Selection of optimal emergency rescue route based on improved ant colony algorithm[C]// International Conference on Geoinformatics. IEEE, 2010

[2] Yu M , Yue G , Lu Z , et al. Logistics Terminal Distribution Mode and Path Optimization Based on Ant Colony Algorithm[J]. Wireless Personal Communications, 2018

[3] Yiyi Y , Linpeng W , Keyang J , et al. Application of Optimal Logistic Distribution Problem Based on Simulated Annealing[J]. Journal of Ningbo University of Technology, 2018

[4] WU Yanqun,DONG Peng. Improved simulated annealing algorithm for vehicle routing problem in supply chain[J]. CEA, 2016, 52(12): 256-260

[5] LUO Yong,CHEN Zhi-ya.Path Optimization of Logistics Distribution Based on Improved Genetic Algorithm[J].Systems Engineering.2012

[6] Han L , Hou H , Yang J , et al. E-commerce distribution vehicle routing optimization research based on genetic algorithm[C]// 2016 International Conference on Logistics, Informatics and Service Sciences (LISS). IEEE, 2016

[7] Qun C . Dynamic Route Guidance Method Based on Particle Swarm Optimization Algorithm[C]// Second International Conference on Intelligent Computation Technology & Automation. IEEE Computer Society, 2009

[8] Zhang K , Qiu B , Mu D . Low-carbon logistics distribution route planning with improved particle swarm optimization algorithm[C]// International Conference on Logistics. IEEE, 2017

[9] Dijkstra E W . A note on two problems in connexion with graphs[M]. Springer-Verlag New York, Inc. 1959

[10] Bozyigit A , Alankus G , Nasiboglu E . Public transport route planning: Modified dijkstra's algorithm[C]// International Conference on Computer Science & Engineering. IEEE, 2017

[11] Xin Zhang,Yan Chen,Taoying Li.Optimization of logistics route based on Dijkstra[C]// IEEE International Conference on Software Engineering and Service Science. IEEE, 2015:313-316

[12] Kadry S , Abdallah A , Joumaa C . On The Optimization of Dijkstras Algorithm[M]// Informatics in Control, Automation and Robotics. Springer Berlin Heidelberg, 2012

[13] Shuxi W . The Improved Dijkstra's Shortest Path Algorithm and Its Application[C]// Seventh International Conference on Natural Computation. IEEE, 2011

[14] Shu-Xi W , An-Yu L I . Multi-adjacent-vertexes and Multi-shortest-paths Problem of Dijkstra Algorithm[J]. Computer Science, 2014