Application of Improved Dijkstra Algorithm in Two-dimensional Path Planning Problem

Zhang Ruifang
Department of Guilin University of Technology, Guilin City, the Guangxi Zhuang Autonomous Region
17307732908
631600961@qq.com

Ji Tianyi
Department of Guilin University of Technology, Guilin City, the Guangxi Zhuang Autonomous Region
18871720021
928284127@qq.com

Zheng Haitao
Department of Guilin University of Technology, Guilin City, the Guangxi Zhuang Autonomous Region
15179285973
1052729718@qq.com

ABSTRACT
The Dijkstra algorithm is a typical single-source shortest path algorithm for calculating the shortest path from one node to the other in a non-negative weight map, but its use of the roulette method greatly affects the node selection. Speed and efficiency. Therefore, on the basis of ensuring the search accuracy, this paper improves the initial Dijkstra algorithm to improve the efficiency of the algorithm and meet its needs in 2D or 3D path planning. In this paper, MATLAB2018a is used as the experimental platform to simulate the initial Dijkstra algorithm and the improved Dijkstra algorithm. The experimental results show that the improved algorithm greatly reduces the path planning vision and improves the operating efficiency.

CCS Concepts
• Theory of computation ~ Shortest paths

Keywords
Greedy Algorithm; Dijkstra algorithm; Shortest Path Planning

*Corresponding author

1. INTRODUCTION
Path planning is an important part of the current vehicle positioning navigation system. It refers to the path from the starting point to the target point through the algorithm when one or more irregular obstacles appear in the provided working environment, and must be collision-free. Bypassing all obstacles that appear in the work environment. So far, there are more than one path planning algorithm, and a wide range of applications include A* algorithm, simulated annealing algorithm, and Dijkstra algorithm.

ICIIP'2019, November 16–17, 2019, China, China
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6191-0/19/11…$15.00
https://doi.org/10.1145/3378065.3378106

The Dijkstra algorithm is widely used because of its good code maintainability and ease of execution. However, the speed and efficiency of the algorithm are not high because of the diversity and uncertainty of its node selection method[1] .

This paper improves the Dijkstra algorithm from this perspective, and improves the running speed and efficiency of the algorithm while ensuring the accuracy of the algorithm, so that it can be widely applied to various problems

2. GREEDY ALGORITHM
The greedy algorithm is also called the greedy algorithm. Its central idea is to solve its heuristics according to the current problem, and make optimal choices in each selection stage, without considering the global results.
The greedy strategy usually does not produce the optimal solution in many problems, but the greedy heuristic algorithm will generate the local optimal solution with great probability, and on the basis of it, the solution to the single problem approximates the global optimal solution. [2] Greedy algorithms are not intended to find the best solution, but they will terminate with reasonable steps; the best solution for solving complex problems often requires many unreasonable steps.
In general, there are five steps to implementing a greedy algorithm:
1. Create candidate sets and create solutions from them
2. Choose the best candidate to add to your solution
3. Choose a feasibility function to determine if it can be used to make the right choice for the solution
4. Determine an objective function that can provide a solution or partial solution assignment
5. Select the end condition to prompt when to find the complete result

3. DIJKSTRA ALGORITHM
The Dijkstra algorithm, proposed by E.W. Dijkstra in 1959, is called the Dijkstra algorithm and is a typical algorithm for solving problems using greedy patterns. It is also the best method recognized in solving the shortest path problem.
The algorithm solves the shortest path problem from a single source point to other vertices in a directed graph. The main feature of the Dijkstra algorithm is that each iteration is selected. The next vertex is the vertex closest to the source point outside the marker point. The specific steps of the Dijkstra algorithm are as follows:

1. Initializing the node set V of the undetermined shortest path and the determined shortest path node set S, and using the adjacency matrix arcs of the weighted graph to initialize the shortest path length D of the source point to other nodes, if the source point has a connecting arc to other nodes, corresponding The value of the value is the weight of the join arc, otherwise the corresponding value takes the maximum value.
2. The minimum value D[i] in D is selected, D[i] is the shortest path length from source point to point i, point i is taken out of set V and placed in set S.
3. The path length value corresponding to the node k in the update array D to the node k in the set V is modified according to the node i.
4. Repeat steps 2 and 3 until you find the shortest path from the source point to all nodes.[4]
The Dijkstra algorithm flow is shown in Figure 1.

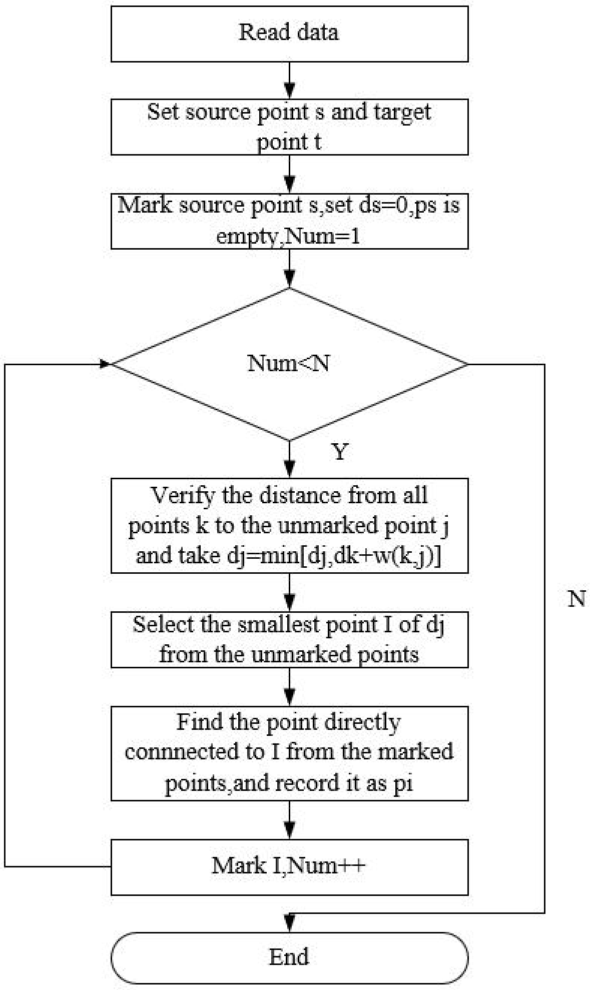
Read data
Set source point s and target point t
Mark source point s,set ds=0,ps is empty,Num=1
Num<N
Y
Verify the distance from all points k to the unmarked point j and take dj=min[dj,dk+w(k,j)]
Select the smallest point I of dj from the unmarked points
Find the point directly connnected to I from the marked points,and record it as pi
Mark I,Num++
N
End

Figure 1 Dijkstra algorithm flow chart

3.1 ALGORITH FLOW

The establishment of the spatial model uses MAKLINK graph theory to establish the two-dimensional space of path planning. The initial path planning of Dijkstra algorithm uses an initial path from the starting point to the ending point to initialize the algorithm parameters.[5] The pheromone update uses the path searched according to the ant colony algorithm. The length and length of the pheromone with the new node.

3.1.1 Representation of the solution

A Dijkstra algorithm is used to generate a sub-optimal path of the path nodes S, P1, P2....Pd, T passing through in the figure. Let Li be the corresponding free link line (i=1.2...d). $P_i^{(1)}, P_i^{(0)}$ is the two endpoints of Li, and other points on the link are as shown in Equation 1.

$$P_i(h_i) = P_i^{(0)} + (P_i^{(1)} - P_i^{(0)}) * h_i \qquad (1)$$

Where hi∈[0,1], i=1,2,...,d,hi is a proportional parameter; d is the number of nodes divided by the link.
It can be seen from the formula that the free link line is obtained by using the Dijkstra algorithm. As long as a set of parameters (h1, h2...hd) is given, a new path from the start point to the end point can be obtained, so the solution of the ant colony algorithm can be expressed as (H1, h2...hd).
The ant colony algorithm is required to discretize the workspace. Since the length of the free link lines for initial selection is different, the division of the link lines is determined by a fixed distance division method, and the division length is set to ζ, and the number of divisions of each free link line L is as follows. Equation 2 is shown.

$$\pi_i = \begin{cases} Int\left(\frac{L_i}{\zeta}\right), Int\left(\frac{L_i}{\zeta}\right) for \text{ odd number} \\ Int\left(\frac{L_i}{\zeta}\right)+1, Int\left(\frac{L_i}{\zeta}\right) for \text{ even number} \end{cases} \qquad (2)$$

When Int(Li/ζ) is an odd number, the midpoint of the path is also a bipartite point, and the number of divisions is πi plus 1. Since the link line Li is equally divided by πi, then from the link line L(i-1) to the other phase The neighboring link line Li has πi+1 roads to choose from.

3.1.2 Node Selection

The ant colony algorithm optimizes the search for the path parameter set to get the shortest path in the discretized space. It is assumed that there are a total of m ants starting from the starting point S and reaching the ending point T, the circulation path is S→T, and ndj is indicating that the path point is at the jth bisector of the dth link line. During the movement, when the ant is on the link line Li, the method of selecting the node j on the next link line L(i+1) is as shown in Equation 3.

$$j = \begin{cases} \arg max_{K\in I}(|\tau_{i,k}| \, |\eta_{i,\ k}^{B}|) ,q \geq q_0 \\ J, \qquad others \end{cases} \qquad (3)$$

Where i is the set of all points on the link line; q is the random number in the interval [0,1]; q0 is the [0,1] interval tunable parameter; η(i,j) is the heuristic value, τ(i,k) for pheromones.
j is calculated by first calculating the selection probability p(i, k) of the current link line node i to the down link line node j, and then using the roulette method to find the next node j according to the selection probability P(i+j) The calculation formula of P(i+j) is as shown in Equation 4.

$$p_{i,j} = \frac{\tau_{i,\ j}\eta_{i,j}^{\beta}}{\sum_{\omega\in I}\tau_{i,\omega}\eta_{i,\omega}^{\beta}} \qquad (4)$$

3.1.3 pheromone update

The pheromone update includes real-time pheromone update and path pheromone update, wherein the real-time pheromone update means that each ant must update the pheromone of the node after selecting a certain node, as shown in Equation 5.

$$\tau_{i,j} = (1-\rho)\tau_{i,j} + \rho\tau_0 \quad (5)$$

Where τ0 is the initial value of the pheromone; ρ is the tunable parameter of the interval [0,1].

When all the ants go from the initial point to the end point and complete the iterative search, select all the ants to pass the shortest one in the path, and update the pheromone of each point on the path as shown in Equation 6.

$$\tau_{i,j} = (1-\rho)\tau_{i,j} + \rho\Delta\tau_{i,j} \quad (6)$$

Where Δτ(i,j)=1/L*, L* is the shortest path length; ρ is the [0,1] interval tunable parameter.

3.2 Improved Dijkstra algorithm

When performing the two-dimensional path planning, the Dijkstra algorithm must pre-plan each node appearing in the search direction during the calculation process,which greatly increases the calculation amount and affects the planning efficiency.

Therefore, this paper proposes an improved Dijkstra algorithm based on the bidirectional programming method. Based on the original algorithm, the improved algorithm performs two-way search, that is, using the Dijkstra algorithm for the shortest path planning at the starting point and the end point. When the prediction nodes in the two directions are consistent, the algorithm is terminated and the result is input.[6]

The algorithm selects the optimal function of the node when planning the path as shown in Equation 7.

$$\mathrm{F(n)} = \mathrm{G(n)} + \mathrm{H(n)} \quad (7)$$

Where G(n) is the actual distance from the starting node to the hypothetical node n; H(n) is the actual distance from the ending node to the hypothetical node n; F(n) is the optimal function of the hypothetical node n. The estimated cost between two points using the Euclidean distance metric is shown in Equation 8.

$$\mathrm{d} = \sqrt{\left(X_g - X_s\right)^2 + \left(Y_g + Y_s\right)^2} \quad (8)$$

Where Xg, Yg represents the horizontal and vertical coordinates of point g, and Xs, Ys represents the horizontal and vertical coordinates of point s.

In the case of predictions, the improved Dijkstra algorithm will eventually meet and plan successfully no matter how it is planned.

However, after simulation, it is found that the improved algorithm will have the problem that the target node in the bidirectional planning of the starting point and end point has never met in the running process, which will cause the algorithm to fall into an infinite loop and cause the planning path to fail.[7]

In order to solve this problem, it is necessary to reproduce the termination condition of the improved Dijkstra algorithm, and establish a heuristic optimal function of the end point bidirectional search as shown in Equations 9,10.

$$F_1(n) = G_1(n) + H_1(n) \quad (9)$$

$$F_2(n) = G_2(n) + H_2(n) \quad (10)$$

The termination conditions for the improved two-way Dijkstra algorithm are:

In bidirectional path planning, if both parties select the same node in any time period, the bidirectional path planning will be marked as the shortest path node at the same time[8] [9] .

When the value of G1(n)+G2(n) is the minimum in all the result calculations, the two-way Dijkstra algorithm terminates; where G1 (n) is the minimum distance from the starting point to the node n, G2 (n)) is the minimum distance from the end point to node n.[10]

When the above two conditions are satisfied by the algorithm, the program terminates the operation and outputs the calculated value.

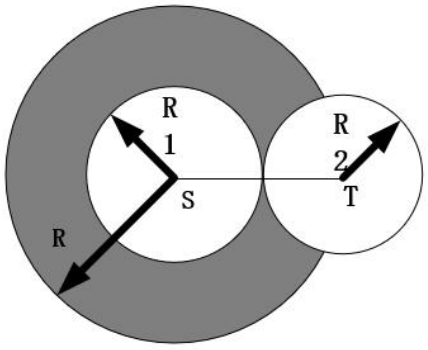

Figure 2 Improved Dijkstra algorithm schematic

In Figure 2, we set S and T as the start and end points of the search path respectively. The area searched by the forward Dijkstra algorithm starting from the starting point is the area with S as the center and R1 as the radius; The area searched by the reverse Dijkstra algorithm that starts the search is the area with the center of T and the radius of R2.

4 SIMULATION AND VERIFICATION

In this paper, under the experimental platform of MATLAB2018a, the Dijkstra algorithm and the improved two-way Dijkstra algorithm are simulated by the same starting point and end point and the two-way path planning simulation of obstacles to avoid accidentality. The computer configuration of this experiment is: Windows10 operating system, Inter i7-8650u processor, running memory 16GB, clocked at 2.11GHZ.

Find an optimal path from S to T in a two-dimensional space of size 200×200KM, and there are four obstacles of different sizes in the two-dimensional space.

The four vertices of the obstacle 1 are respectively (30 140 ; 60 160; 100 140; 70 120), the four vertices of the obstacle 2 are (50 40; 30 40; 60 70; 100 40), and the four vertices of the obstacle 3 are respectively (120 160; 140 100; 180 170; 165 180), the three vertices of obstacle 4 are (120 40; 170 40; 140 80), where point S is the starting point, the starting point coordinates are (20, 180); point T is the end point, and the end point coordinates are (160, 90).

Based on the phaseless network diagram, the Dijkstra algorithm is used for path planning. The results are shown in Figure 3.

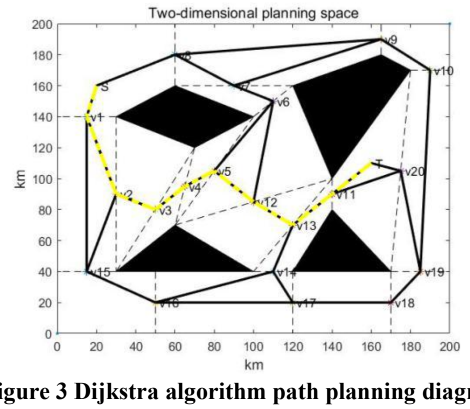

Figure 3 Dijkstra algorithm path planning diagram

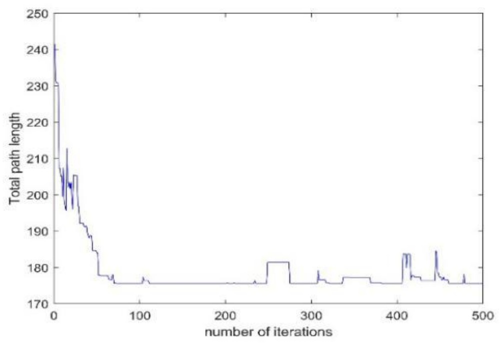

Figure 4 Digestra algorithm iteration diagram

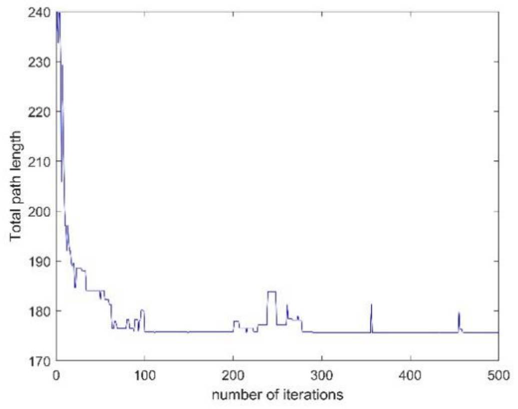

Figure 5 Two-way Dijkstra algorithm iterative graph

It can be seen intuitively from Figure 3 that in order to plan the path from the starting point S to the ending point T, many nodes are discarded when the original algorithm performs path selection, which greatly reduces the running speed and efficiency of the algorithm.

It can be seen from Fig.4 and Fig.5 that after 500 iterations of algorithm,the total path length of the two algorithms is significantly reduced,and the bidirectional Dijkstra algorithm has a planned path of 300 compared with the initial algorithm.There was no major fluction after the second.

Table 1 Comparison of the improved Dijkstra algorithm and the original algorithm search results

S	T	T1/ms	T2/ms	D1/km	D2/km	N1	N2
(20, 180)	(160, 90)	1206	241	245.861	245.861	20	13
(30, 170)	(150, 80)	1183	236	258.370	258.370	33	25
(50, 170)	(160, 60)	1253	250	341.253	341.253	26	19

In Table 1, S and T are the starting and ending points of the planning path respectively; T1 and T2 are the time required for the two algorithms to run; D1 and D2 are the lengths of the path planning; and N1 and N2 are the two algorithms for the planning path. The number of nodes.

It can be seen from Table 1 that the improved Dijkstra algorithm and the initial algorithm search for the shortest path are exactly the same when the start point and the end point are completely different, indicating that the two-way Dijkstra algorithm is identical in accuracy to the original algorithm.However, T1 is about 5 times that of T2, indicating that the improved algorithm is about 5 times faster than the initial algorithm; and N2 is also slightly smaller than N1, indicating that the two-way Dijkstra algorithm is slightly less than the edge planned by the initial algorithm.

5.CONCLUSION

This paper proposes a modified Dijkstra algorithm using bidirectional path planning. When the algorithm performs path planning, the planning of the starting point and the end point is synchronized, and the bidirectional planning exchange is used to avoid the problem that the two planning paths cannot meet each other. The theoretical and simulation results show that the two-way Dijkstra algorithm can improve the planning efficiency and reduce the running time on the basis of ensuring the accuracy of the algorithm planning path, and it has the use value under other different path planning problems. In the future research, the related algorithms related to path planning still have the prospect of continuous improvement, which can improve the planning efficiency in practical application problems.

6.ACKNOWLEDGMENT

This work was supported by the Guangxi Key Laboratory of New Energy and Energy Conservation Project(Gui ke Neng 15-J-21-1)

7.REFERENCES

[1] Wolfgang Fink,Victor R. Baker,Alexander J.W. Brooks,Michael Flammia,James M. Dohm,Mark A. Tarbell. Globally optimal rover traverse planning in 3D using Dijkstra's algorithm for multi-objective deployment scenarios[J]. Planetary and Space Science,2019

[2] Gao Xincheng, Liu Deju, Wang Lili, Ma Shuxuan.Design and Optimization of QOS Routing Model Based on Ant Colony Algorithm[J].Journal of Shaanxi University of Technology(Natural Science Edition),2019(02):67-72.

[3] Wu Xiaonian, Yan Yuang, Zhang Runlian.A hybrid path planning algorithm based on genetics and ant colony in DEM [J/OL]. Computer Application Research: 1-5[2019-09-21].

[4] Liu Guofeng.Research on Path Planning of Industrial Manipulator Based on Ant Colony Algorithm[J].Internal Combustion Engines & Parts,2019(16):119-121.

[5] Wang Haifeng,Chen Jingming,Wang Guangmin.Study on Optimization of Rural Garbage Recycling Route Based on Improved Distance and Ant Colony

Algorithm[J].Environmental Sanitation Engineering,2019,27(04):87-92

[6] Wang Zhaonan.Research and Implementation of Optimal Path Calculation Method for Massive Data Based on Dijkstra Algorithm[J].Bulletin of Surveying and Mapping,2012(09):32-34+37..

[7] Chen Honglu,Deng Yiran.Study on Large Building Escape Scheme Based on Game Theory and Dijkstra Algorithm[J/OL].Digital Technology and Application: 1-7[2019-09-21].

[8] Wang Shuxi,Li Anzhen.Multiple Adjacent Points and Multiple Shortest Path Problems in Dijkstra Algorithm[J].Computer Science,2014,41(06):217-224.

[9] Wang Zhihe,Ling Yun.Optimization and Implementation of Dijkstra Shortest Path Algorithm[J].Control & Automation,2007(33):275-277

[10] Le Yang,Gang Jianya.A High Efficiency Implementation of Dijkstra Shortest Path Algorithm[J].Journal of Wuhan Institute of Surveying and Mapping,1999(03):209-212.