AGV Path Planning based on Improved Dijkstra Algorithm

Yinghui Sun 1, Ming Fang 2 and Yixin Su 2,*

1Department of Purchasing Management CNNP Nuclear Power Operations Management Co., Ltd, Jiaxing, China
2School of Automation, Wuhan University of Technology, Wuhan, China

*Corresponding author email: suyixin@whut.edu.cn

Abstract. Aiming at the path planning problem of Automated Guided Vehicle (AGV) in intelligent storage, an improved Dijkstra algorithm that combines eight-angle search method and Dijkstra algorithm for path optimization is proposed. The grid method is used to model the storage environment, and the improved Dijkstra algorithm is used to optimize the route of the AGV. The simulation test of the AGV path planning process with Matlab shows that the AGV can effectively avoid obstacles by using the traditional Dijkstra algorithm and the improved Dijkstra algorithm, and then search for a collision-free optimized path from the start point to the end point; and the traditional Dijkstra algorithm In comparison, the path length planned by the improved Dijkstra algorithm is shorter and the turning angle is less, indicating that the improved algorithm is correct, feasible and effective, and has a strong global search ability.

Keywords: AGV; intelligent warehousing; Dijkstra algorithm; route plan.

1. Introduction

The handling of goods in the warehouse is related to the shortest path and transportation efficiency of the automatic navigation vehicle, and is an important part of the construction of intelligent warehouse. AGV is equipped with electromagnetic or optical automatic navigation devices, which can ensure that the system can automatically drive along a predetermined route without manual piloting, and automatically transport goods or materials from the starting point to the destination. Path planning belongs to the traveling salesman problem [1], that is, the TSP problem (Traveling Salesman Problem) is also translated as the traveling salesman problem, which is one of the famous problems in the field of mathematics. That is to say, if a traveling merchant wants to visit multiple cities, he must choose the path he wants to take. The restriction of the path is that each city can only be visited once, and he must return to the original city in the end. The path selection goal is that the required path distance is the smallest value among all paths. Many scholars have used genetic algorithm [2], A* method [3], simulated annealing method [4], ant colony algorithm [5], etc. to solve the problem. Among them, genetic algorithm refers to finding the optimal path by simulating the selection and genetic mechanism in nature. It has a good global search ability, but when dealing with traveling salesman problems, there is a path crossing phenomenon, which leads to suboptimal search results; A* algorithm It is a widely used shortest path heuristic search algorithm. The search results of this algorithm are closely related to the heuristic function. Different heuristic functions will lead to different search paths; the simulated annealing algorithm can find the global shortest path in the sense of probability, but The performance of the algorithm is affected by the initial parameters, and the path search speed is relatively slow; the ant colony algorithm is a modern bionic evolutionary algorithm with strong robustness, but it is prone to the defects of slow algorithm convergence and easy to fall into local optimum.
Literature [6] applies Dijkstra's algorithm to topological maps, and verifies the feasibility of this method through Pyretic simulation software. In order to further improve the performance of Dijkstra algorithm, literature [7] uses Dijkstra algorithm and ant colony algorithm to mix and introduces a random node selection mechanism, literature [8] uses genetic algorithm to optimize the shortest path and delivery order of Dijkstra algorithm. These two methods can make it have better global search ability and shorter path length respectively, but at the same time they also increase the complexity of the algorithm and the complexity of the path.

This paper uses raster map to model the storage environment, optimizes Dijkstra's search path method, and verifies the effectiveness of the improved algorithm through MATLAB simulation.

2. Environment Modeling

Dijkstra algorithm is a search algorithm for solving the optimal path in a static grid. Before solving the shortest path between two points, it is necessary to model the path environment of the AGV. Generally, the grid method is used to divide the environment. The grid method decomposes the storage environment into a series of grid units with binary information, which are usually represented by a quadtree or an octree. The proportion of the grid has a direct impact on the storage of environmental information and the complexity of the model.

This article uses squares of the same size to represent the shelf information and the AGV model. In a square-distributed warehouse, there are 20 rows and columns, each of which is a 1m×1m square, with a total of 400 areas. First, use the lower left corner as the coordinate origin, the horizontal direction of the origin as the X axis, and the vertical direction of the origin as the Y axis to establish the XOY coordinate system, and then use the AGV unit motion step length L as the index to divide the X axis and the Y axis. In the grid area, finally number the grids according to the principle of left to right and top to bottom, and ensure that each grid number and its coordinates maintain a one-to-one correspondence [9]. The white grid represents the free grid, the black grid represents the obstacle grid, and START and GOAL are the start and end positions of the AGV, respectively. On the Matlab platform, the AGV operating environment model established by the grid method, Figure 1 is the grid map of the intelligent storage environment.

![Figure 1. Grid map of smart storage environment.](image)

3. Basic Principle of Dijkstra algorithm

Dijkstra algorithm is to find the shortest path point and finally get a shortest path tree. The algorithm uses a greedy strategy. The algorithm steps are as follows:

**Step 1:** Divide the weighted vertices of the directed graph \( \{V\} \) into two groups \( \{A=s\} \) and \( \{B=V-s\} \), with \( s \) as the source point.

**Step 2:** where \( \{A\} \) is the set of vertices for which the shortest path has been found (initially only the source point \( s \)) initializes the distance from \( A \) to the vertices in \( \{B\} \) as the weight of the edge.

**Step 3:** Select the vertex via with the smallest distance \( s \) from \( \{B\} \), and add \( via \) to \( A \).

**Step 4:** Using \( via \) as the newly considered intermediate point, modify the distance of each vertex in \( \{B\} \); if the distance from the source point \( s \) to the vertex \( x \) (through the vertex \( via \)) is shorter than the original distance, update it.

\[
dist(s, via) + dist(via, s) < dist(s, x)
\]  

(1)
Step 5: Repeat step 3, step 4 can get all vertices in \{A\}, that is, the search is completed.

Taking the node connection diagram shown in Figure 2 as an example, Table 1 shows the flow of the shortest path from point A as the starting point to points B and C respectively.

**Figure 2.** Node connection diagram.

**Table 1.** Dijkstra algorithm calculation steps.

| Algorithm Steps | S Collection | U Set |
|-----------------|-------------|-------|
| 1               | Select node A, S = < A > | U = B, C, D, E, F, G > |
|                 | A → A = 0 | A → B = 75, A → C = 61, A → D = 10, |
|                 | Take node A as the middle point, start searching from node A. | A → Other vertices in U = ∞ |
|                 | A → D = 10 | It is found that A → D = 10 has the shortest weight. |
|                 | Using node D as the middle point, start searching from the shortest path | |
|                 | A → D = 10 | |
|                 | A → D → B = 93 | |
|                 | A → D → C = 60 | |
|                 | A → D → E = 30, A → D → G = 22 | |
|                 | A → D → Other vertices in U = ∞ | |
|                 | It is found that A → D → G = 22 has the shortest weight | |
| 2               | Select node D, S = < A, D > | U = B, C, E, F, G > |
|                 | Using node D as the middle point, start searching from the shortest path | |
|                 | A → D = 10 | |
|                 | A → D → B = 93 | |
|                 | A → D → C = 60 | |
|                 | A → D → E = 30 | |
|                 | A → D → G = 22 | |
|                 | A → D → Other vertices in U = ∞ | |
|                 | It is found that A → D → G = 22 has the shortest weight | |
| 3               | Select the G node, S = < A, D, G > | U = B, C, E, F > |
|                 | A → D → G = 22 | |
|                 | A → D → G → E = 59 | |
|                 | A → D → G → F = 38 | |
|                 | Take G node as the middle point, start searching from the shortest path | |
|                 | A → D → G = 22 | |
|                 | A → D → G → E = 59 | |
|                 | A → D → G → F = 38 | |
|                 | A → D → Other vertices in U = ∞ | |
|                 | It is found that A → D → G → F = 38 has the shortest weight | |
| 4               | Select the E node, S = < A, D, E > | U = B > |
|                 | A → A = 0 | |
|                 | A → D = 10 | |
|                 | A → D → E = 30 | |
|                 | Using node E as the intermediate node, start searching from the shortest path | |
|                 | A → D → E = 30 | |
|                 | A → D → E → B = 72 | |
|                 | Found that A → D → E → B = 72 (less than the previously searched A → B = 75) | |
| 5               | Select node B, S = < A, D, E, B > | The U collection is empty and the search is complete. |
|                 | A → D → E = 30, A → D → E → B = 72 | |
| 6               | Select node F, S = < A, D, G, F > | U = C > |
|                 | A → D → E = 30 | |
|                 | A → D → G → F = 38 | |
|                 | With F as the middle point, start searching from the shortest path | |
|                 | A → D → G → F = 38 | |
|                 | A → D → G → F → E → B = 101 | |
|                 | A → D → G → F → C = 53 | |
|                 | A → D → G → F → C = 60 | |
| 7               | Select node C, S = < A, D, G, F, C > | The U collection is empty and the search is complete. |
|                 | A → D → E = 30 | |
|                 | A → D → G → F = 38 | |
|                 | A → D → G → F → C = 53 | |

4. Improved Dijkstra Algorithm based on Eight-angle Search

The traditional Dijkstra algorithm usually uses four-angle search for node expansion in a grid map, as shown in Figure 3. The four-angle search limits the steering angle of the AGV to a certain extent and increases the redundant path. As shown in Figure 4, taking a certain free grid as the starting point, you can search in the eight directions of the grid due to west, southwest, due north, northwest, due east,
northeast, due south, and southeast. Shorten the distance to the target point, which is more suitable for operation in a storage environment under complex conditions.

First, assume that the expression of the storage grid graph is: \( G=(V,E,W) \), where the vertex set \( V=\{v_1,v_2,...,v_p\} \), that is, the number of vertices \( |V|=p \). \( w_{ij} \) represents the weight of the edge \((v_i, v_j)\), and needs to satisfy the non-negative condition \( w_{ij} \geq 0 \). If \((v_i, v_j) \notin E\), let \( w_{ij}=\infty \). The shortest path problem is to find the distance from \( v_1 \) in \( G \) to the other vertices. Let \( d(v_j) \) denote the weight of the shortest path from \( v_1 \) to \( v_j \) that only passes through the selected vertices. The corresponding algorithm steps are as follows.

**Step 1**: Initialize, let \( d(v_1)=0 \), \( d(v_j)=w_{1j} \) \((j=2,3,...,n)\), \( S=\{v_1\} \), \( R=V\setminus S=\{v_2,v_3,...,v_p\} \).

**Step 2**: Find a vertex \( v_k \) in \( R \) such that:

\[
d(v_k) = \min_{v_j \in R} \{d(v_j)\}
\]  

(2)

Set \( S=S \cup \{v_k\} \), \( R=V \setminus S \). If \( R=\emptyset \), the algorithm terminates, otherwise, go to step 3.

**Step 3**: Correct \( d(v_j) \). For each \( v_j \) in \( R \), set: \( d(v_j)=\min\{d(v_j), d(v_k)+w_{kj}\} \) and go to step 2.

After this algorithm goes through \(|V|-1\) cycles, all vertices are selected, and the final value of \( d(v_j) \) \((j=1,2,...,p)\) gives the vertex \( v_1 \) to the remaining vertices \( v_j (j=2,3,...,p) \) the length of the shortest path, the shortest path can be obtained by backward tracking.

5. Simulation Research

In order to realize the feasibility of Dijkstra method in AGV path optimization under the storage environment, the simulation computer is configured with Intel i5 quad-core processor, 64-bit Win10 operating system, and Matlab2018 simulation software.

**Figure 5.** Unoptimized Dijkstra Algorithm.

**Figure 6.** The optimized Dijkstra Algorithm.

**Figure 7.** Ant Colony Algorithm.
As shown in Figure 5, Figure 6, Figure 7, Figure 8 and Figure 9, the red initial starting point and the green target end point of the AGV are artificially defined. The red line is the path area of the AGV trolley. The optimization results are shown in Table 2. The analysis shows:

a. Compared with the traditional Dijkstra algorithm, the path obtained by the improved Dijkstra algorithm reduces the shortest distance, total turning angle and running time by 39.04%, 75.00%, and 18.35% respectively. The improved Dijkstra algorithm replaces a straight path with a polyline path, making the path distance shorter. Although the number of turning points increases, the turning angle is relatively smooth and the turning degree decreases.

b. Compared with the ant colony algorithm, the improved Dijkstra algorithm reduces the shortest distance and total turning angle by 4.73% and 37.49% respectively. Although the time of the ant colony algorithm is less than that of the optimized Dijkstra algorithm, the program to run the ant colony algorithm is more complicated. In addition, the optimized Dijkstra algorithm reduces the total turning angle and the number of turning points compared with the ant colony algorithm, which is more in line with the actual path requirements of the AGV in the storage environment.

c. According to the above comparative analysis, in the AGV path planning problem in the storage environment, the optimized Dijkstra algorithm uses a polyline path instead of a straight path to make the distance to the target point shorter, less time, smoother steering, and more efficient. High, with better feasibility and practicality.

| Parameter                  | Dijkstra Algorithm | Improved Dijkstra Algorithm | Ant Colony Algorithm | Genetic Algorithm | A star algorithm |
|----------------------------|--------------------|------------------------------|----------------------|------------------|-----------------|
| Shortest distance (m)      | 38                 | 27.331                       | 28.624              | 30.624           | 38              |
| Number of turning points   | 7                  | 8                            | 11                  | 13               | 15              |
| Total turning angle (rad)  | 10.9956            | 6.2832                       | 8.6387              | 10.2102          | 11.781          |
| Running time (s)           | 13.516             | 11.420                       | 11.137              | 14.205           | 13.662          |

6. Conclusion
Aiming at the problem of AGV path planning in intelligent storage, the traditional Dijkstra algorithm is improved by introducing an eight-angle search method. In order to test the actual effect of the improved algorithm, MATLAB software was first used in the grid environment to simulate the traditional Dijkstra algorithm, ant colony algorithm, genetic algorithm and A star algorithm. The results can be concluded: there are obstacles Under the operating environment, AGV can effectively avoid obstacles, and then search for a path from the starting point to the key point; compared with testing other algorithms, the path length planned by the improved Dijkstra algorithm is the shortest, the total turning angle is the smallest, and the turning point The number is the least, and the complexity of the algorithm is low. The algorithm has more efficient search results on the AGV path optimization problem, and is more in line with the actual application of AGV in the storage environment.
References

[1] Q. Wang, "Improve TSP problem allowing multiple vehicles distribution," 2016 International Conference on Logistics, Informatics and Service Sciences (LISS), Sydney, NSW, 2016, pp. 1-3, doi: 10.1109/LISS.2016.7854489.

[2] TSAI C, HUANG H, CHAN C. Parallel elite genetic algorithm and its application to global path planning for autonomous robot navigation[J]. Transactions on Industrial Electronics, 2011, 58(10): 4813-4821.

[3] ANSARIA, SAYYEDMA, RATLAMWALAK, et al. An optimized hybrid approach for path finding [J]. International Journal in Foundations of Computer Science & Technology, 2015, 5(2): 47-58.

[4] P. Shi and S. Jia, "A Hybrid Artificial Bee Colony Algorithm Combined with Simulated Annealing Algorithm for Traveling Salesman Problem," 2013 International Conference on Information Science and Cloud Computing Companion, Guangzhou, 2013, pp. 740-744, doi: 10.1109/ISCC-C.2013.13.

[5] PAWAN KUMAR TIWAR, DEO PRAKASH Vidyarthi. Improved auto control ant colony optimization using lazy ant approach for grid scheduling problem [J]. Future Generation Computer Systems, 2016, 60(8): 78-89.

[6] M. Lee and K. Yu, "Dynamic Path Planning Based on an Improved Ant Colony Optimization with Genetic Algorithm," 2018 IEEE Asia-Pacific Conference on Antennas and Propagation (APCAP), Auckland, 2018, pp. 1-2, doi: 10.1109/APCAP.2018.8538211.

[7] Z. Nie and H. Zhao, "Research on Robot Path Planning Based on Dijkstra and Ant Colony Optimization," 2019 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS), Shanghai, China, 2019, pp. 222-226, doi: 10.1109/ICIIBMS46890.2019.8991502.

[8] H. Juzoji, I. Nakajima and T. Kitano, "A Development of Network Topology of Wireless Packet Communications for Disaster Situation with Genetic Algorithms or with Dijkstra's," 2011 IEEE International Conference on Communications (ICC), Kyoto, 2011, pp. 1-5, doi: 10.1109/icc.2011.5962439.

[9] Kang B, Wang X H, Liu F. Path planning of searching robot based on improved ant colony algorithm [J]. Journal of Jilin University (Engineering and Technology Edition), 2014, 44(4): 1062-1068.