Path planning of intelligent mobile robot based on Dijkstra algorithm

Xueyan Li$^1,*$

$^1$School of Artificial Intelligence, Wuchang University of Technology, Wuhan 430223, Hubei, China

*Corresponding author: xueyanli@wut.edu.cn

Abstract. Under the premise of grid environment modeling method, a relaxed Dijkstra algorithm is proposed to solve the problem of real-time path planning for mobile robot in large-scale and obstacle intensive working environment. Firstly, four neighborhood search is used to construct the Manhattan distance potential field from the source point to the global point in linear time, and then eight neighborhood search is performed from the target point to the source point to return a collision free and approximately optimal path. Matlab simulation results show that the algorithm is 10 times faster than dijkstra algorithm and A-star algorithm, and the error of path length is within a reasonable range compared with the shortest path.

Keywords: Grid method; Mobile robot; Path planning; Relaxed Dijkstra algorithm

1. Introduction

Mobile robot path planning [1] is to determine a collision free path for the robot from the starting point to the end point according to certain optimization criteria (usually using distance), such as distance, time and energy. Under constraints such as large working space and dense barriers, path planning problem has been proved to be NP problem [2].

Usually, path planning begins with modeling the working environment of mobile robots, namely map construction. Map construction [3] is divided into road marking method and grid method. The former is to construct a feasible path diagram of the robot composed of marks and connecting edges, such as viewable method and tangent method. The construction process is complex, the accuracy is low, and it is not suitable for the environment with large working space and dense obstacles. The latter is mainly to discretize the environment into regular basic units. Its discrete property is convenient for map construction, but it is difficult to complete real-time and high precision path planning on raster maps with large grid scale and dense obstacles.

Based on the grid environment modeling method, there are many methods for mobile robot path planning. The classical graph search algorithms include Dijkstra algorithm, a * algorithm and so on. This kind of algorithm is accurate and can get the optimal solution, but with the expansion of the grid size, the computational efficiency is low. Harabor et al. [6] proposed the jump point search method to speed up a * algorithm, but the speed increase degree will depend on the landform of the basic grid map. If these maps are characterized by large open areas, this method can make the speed increase of a * algorithm higher, if not, the speed increase is moderate [7]. The relaxed Dijkstra algorithm proposed by
Ammar et al. [8] Based on the structural characteristics of raster map has the possibility of many unnecessary broken lines and large error in the worst case. In addition, artificial intelligence path planning technology represented by evolutionary computing has also been developed rapidly. Evolutionary computing is a kind of self-adaptive optimization probability search algorithm which simulates the theory of biological evolution. It mainly includes genetic algorithm [9], ant colony algorithm [10], particle swarm optimization algorithm [11], artificial bee colony algorithm [12], etc. but the research of this kind of algorithm is mostly limited to small-scale working environment, and it is difficult to carry out real-time and effective path planning in large-scale working environment. Alajla et al. [13] proved the effectiveness of genetic algorithm for global path planning in large-scale grid environment when distance was used as the evaluation criterion, but the computational efficiency was not considered.

In order to solve the problem of real-time path planning for mobile robot in large-scale and obstacle intensive working environment, a new relaxed Dijkstra algorithm (rdj algorithm) is proposed in this paper. Firstly, the grid method is used to model the working environment, and then the rationality of rdj algorithm is elaborated and analyzed. Finally, the experimental comparison and analysis results are given to verify its effectiveness.

2. Grid modeling

Grid method was first proposed by W.E. Howden in 1968. Its core idea is to transform the working environment of mobile robot into a binary matrix, in which the basic element values of the matrix are only 0 and 1. 0 represents the area that the mobile robot can pass through, and 1 represents the obstacle area. In Matlab environment, you can see the working environment represented by grid method more intuitively, as shown in Figure 1, s represents the starting point, e represents the end point.

![Figure 1. Grid based mobile robot working environment](image)

The search of collision free path from s to e needs to follow certain search rules. The algorithm used in this paper needs to use four neighborhood search (Fig. 2 (a)), and the cost of moving from s point to all directions is 1. After traversing the whole region, eight neighborhood search (Fig. 2 (b)) is used to get the path. The horizontal and vertical moving cost is 1, and the diagonal moving cost is $\sqrt{2}$.

![Figure 2. Two kinds of neighborhood search](image)
Figure 3. Diagonal path selection

When using eight neighborhood search to obtain the path, in order to ensure the shortest path and the safety and reliability of the path, there are three cases in diagonal path selection, as shown in Fig. 3, in which the solid line is passable and the dotted line is not passable.

3. RDJ algorithm and its implementation

In graph theory, DJ algorithm is one of the classic algorithms to find the shortest path. For a connected weighted graph with n vertices, the time complexity of single source shortest path algorithm is $O(n^2)$, which can reduce the time complexity of DJ algorithm by using more efficient data structure. Barbehen et al. [14] used heap sorting algorithm to reduce the time complexity of DJ algorithm to $O(m + m\log(n))$, but it is still difficult to meet the requirements of real-time path planning. Bucket sorting [15] can sort uniform data in linear time, but if the data is not uniform, it will not only reduce the efficiency of the algorithm, but also cause a great waste of storage space.

3.1. Design of node expansion mechanism

DJ algorithm belongs to the algorithm of finding the shortest path from a single source point, and the final result is the shortest path from each node to the source point. In the process of node expansion, there are two important operations: one is to relax each node; the other is to find a node with the smallest distance from the source point from the OPEN list as a new expansion node in each iteration, and delete it from the OPEN list as an expanded node and insert it into the CLOSE list. In this process, it is the most time-consuming to extract the node with the smallest distance from the OPEN list to the source point. In order to reduce the time consumption of DJ algorithm, we can design a new node expansion mechanism to avoid the above two operations.

On the basis of environment modeling based on grid method, the algorithm adopts four neighborhood search (Fig. 2 (a)). Assuming that point s is the starting point, we can see that the distance from the four nodes extending from point s to s is 1, and then add the four nodes to the OPEN list respectively. Then, select a node from the OPEN list with the smallest distance to s point. At this point, it can be found that each node can be used as the next expansion node, so it will be redundant to select the node with the smallest distance to s point from the OPEN list. Therefore, four nodes can be regarded as the next expansion nodes in a certain order, that is, the breadth first search algorithm can be used to expand the nodes. The algorithm idea of the extension mechanism is described as follows:

Input: binary matrix (working environment); Starting point s
Output: L(x) (Manhattan distance from each node x to s)
1) $L(x) = 0$, the obstacle node set is O;
2) All nodes and $x \neq s$, $L(x) = \infty$;
3) Insert s point into the end of Q queue;
4) When Q is not empty, get the team head node v of Q;
5) For all four nodes $x$, $x \in Q$ and $x \notin O$ adjacent to v, find $L(x) = L(V) + 1$, insert x into the end of queue Q, and return 4).

According to the algorithm described above, the Manhattan distance potential field from each vertex x to s in Figure 1 can be obtained, as shown in Fig. 4. In addition, this algorithm is different from DJ algorithm in that it does not consider marking the parent node of the node to be extended when extending the node. When the traversal of all extensible nodes is completed, the shortest path from each point to
the starting point still cannot be obtained. It can be seen from Fig. 4 that if e point is taken as the target point and Manhattan distance is taken as the path evaluation standard, there are many possible shortest paths from e point to s point (line in Fig. 4), so this method can not meet the requirements of actual path planning.

Figure 4. Manhattan distance potential field from each vertex to s and all feasible paths from e to s

3.2. Design of reverse search algorithm

It can be seen from Fig. 4 that the path from point e to point s is only obtained by searching in the horizontal or vertical direction from point E, and the change rate along the direction of the path is 1. If the search in the diagonal direction is considered, the change rate in the direction of the path in Fig. 4 is 2. When the change rate of the path along the diagonal direction is 2, it means that the path in this direction is closer to the starting point, and in the actual moving process, if the robot moves along the diagonal direction, its moving cost is $2 - \sqrt{2}$ times lower than that along the horizontal or vertical direction. According to the Manhattan distance potential field, a more effective reverse eight neighborhood search algorithm is obtained. The algorithm is described as follows:

Input: Manhattan distance potential field from source point to each point
Output: approximate shortest path from point e to s

1) $v = e$;
2) When $v \neq s$, find all the eight adjacent nodes x of v;
3) Compare 8 nodes L(x), get the smallest node x and store x;
4) $v = x$, returns 2).

In essence, the algorithm uses the principle of greedy algorithm to construct the global optimal solution by selecting the local optimal solution. It can be seen from Figure 5 that a shortest path can be obtained from point e to point s, and its length is about $10 + 10\sqrt{2}$. Compared with the path in Fig. 4, its length is reduced by $20 - 10\sqrt{2}$. If the Manhattan distance from the source point to the target point is $p$, the path length returned by the algorithm is in the following range:

$$\frac{\sqrt{2}}{2} p \leq L \leq p$$  (1)
3.3. Path length error analysis

From point e to point s, the shortest path obtained by using DJ algorithm for eight neighborhood search is consistent with the length of the path obtained in Fig. 5, but this does not prove that the path obtained by using RDJ algorithm is the shortest path. In fact, RDJ algorithm is defective. In the process of eight neighborhood reverse search, the change rate of the algorithm will be equal in two directions, as shown in Fig. 6(a). At this time, the change rate of the direction from point e to the left or down is the same, but in fact, the distance from the left to point s is shorter than that from the down to point s, which is also the case of the largest error. In Fig. 6(a), the Manhattan distance of the target point is \( p = 10 \), and the maximum error between the path length obtained by RDJ algorithm and the shortest path length is as follows:

\[
\mu = p\left(1 - \frac{p}{2}\right) + 2\sqrt{2} - 4
\]  

(2)

In order to avoid the maximum error, it is necessary to improve the algorithm. When the change rates of the two directions are equal, the algorithm accepts either direction with equal probability, which avoids the situation of large error to a certain extent. However, when the case in Fig. 6(b) \( (P = 10) \) occurs, the error cannot be avoided. In this case, the maximum error is:

\[
\mu = p\left(1 - \frac{p}{2}\right) + 4\sqrt{2} - 8
\]  

(3)
Figure 6. Two kinds of defects (the solid line is the shortest path, and the dotted line is the path obtained by rdj algorithm)

The two cases in Fig. 6 are special cases when errors occur. Equations (2) and (3) are error analysis for the above two special cases, which are not general. In the actual working environment of mobile robot, the global environment as shown in Fig. 6 is rare, and the simulation experiment also proves that the errors caused by these two situations are in a reasonable range.

3.4. Analysis of algorithm time complexity
The RDJ algorithm in this paper is different from the DJ algorithm in two aspects. First of all, in the expansion mode of search node, the relaxation operation and optimization are abandoned, and the expansion is carried out according to the order of inserting the OPEN table, in which the OPEN table is implemented by circular queue. The basic operation of OPEN table is only insert and delete. In the circular queue, the time complexity of insert and delete is O(1). Therefore, RDJ algorithm only needs O(n) time to traverse the whole search space. Secondly, DJ returns a shortest path along the marked parent-child node, while RDJ uses the idea of greedy algorithm to search for the node with the shortest distance from the starting point in the range potential field. The time complexity of this process is less than O(n). In conclusion, the time complexity of RDJ algorithm is O(n).

4. Experimental comparison and analysis
In order to verify the effectiveness of RGJ algorithm, it is compared with the classic DJ algorithm and A* algorithm in terms of running time and path length. The simulation tools are: Visual Studio 2010, Matlab 2010b, the algorithm running platform is Windows 7 system, Intel Core I3 processor (2.53 GHz), memory is 2 GB. The map environment of the algorithm is random distribution of obstacles (hereinafter referred to as random, with 30% obstacle coverage), maze and warehouse, as shown in Fig. 7. The map scale is divided into 25 × 25, 50 × 50, 100 × 100 grid. Among them, in order to speed up the program running speed, in the data structure, the open table of classic DJ algorithm and A* algorithm is based on binary heap priority queue, and is implemented by mixed programming of C++ and MATLAB. In the process of the experiment, the three algorithms are run 100 times on each map, in which the starting point of each map is selected at the bottom left, and the ending point is selected at the top right.

Figure 7. Three test maps
Table 1. Running time comparison of RDJ algorithm, DJ and A* algorithm

| Map type | Grid scale | Minimum time/s | Maximum time consumption/s | Average time/s |
|----------|------------|----------------|---------------------------|----------------|
|          |            | RDJ            | DJ                        | A*            |
|          |            | Minimum       | Maximum                   | Minimum       |
| Random   | 25 x 25    | 0.0041        | 0.2528                    | 0.0629        | 0.0096        | 0.3165        | 0.2003        | 0.0061        | 0.2574        | 0.0656        |
| Maze     | 25 x 25    | 0.0041        | 0.1207                    | 0.0844        | 0.0078        | 0.1480        | 0.0939        | 0.0065        | 0.1296        | 0.0865        |
| Warehouse| 25 x 25    | 0.0043        | 0.1355                    | 0.0942        | 0.0100        | 0.1461        | 0.1052        | 0.0067        | 0.1369        | 0.0566        |
| Random   | 50 x 50    | 0.0146        | 1.1023                    | 0.4090        | 0.0272        | 1.2411        | 0.4587        | 0.0200        | 1.1498        | 0.4306        |
| Maze     | 50 x 50    | 0.0156        | 0.6759                    | 0.6924        | 0.0943        | 0.6921        | 0.7101        | 0.0218        | 0.6790        | 0.6961        |
| Warehouse| 50 x 50    | 0.0121        | 0.5849                    | 0.6745        | 0.0199        | 0.5977        | 0.7176        | 0.0181        | 0.5880        | 0.6795        |
| Random   | 100 x 100  | 0.0509        | 4.8763                    | 0.5808        | 0.1266        | 5.9406        | 0.6459        | 0.0566        | 5.2011        | 0.6048        |
| Maze     | 100 x 100  | 0.0487        | 2.5128                    | 1.7310        | 0.0654        | 2.5470        | 1.7479        | 0.0547        | 2.5222        | 1.7367        |
| Warehouse| 100 x 100  | 0.0405        | 2.4214                    | 2.7325        | 0.0506        | 2.7174        | 2.9317        | 0.0427        | 2.4365        | 2.7469        |

Table 2. Comparison of the length of the path obtained by RDJ algorithm and the shortest path

| Map type | Grid scale | RDJ minimum | RDJ maximum | RDJ average | Shortest path minimum | Shortest path maximum | Shortest path average |
|----------|------------|-------------|-------------|-------------|-----------------------|-----------------------|-----------------------|
| Random   | 25 x 25    | 35.7988     | 35.7988     | 35.7988     | 0                      | 0                     | 0                     |
| Maze     | 25 x 25    | 51.6982     | 51.6982     | 51.6982     | 0                      | 0                     | 0                     |
| Warehouse| 25 x 25    | 46.2406     | 46.8284     | 46.5472     | 0                      | 0                     | 1.27%                 |
| Random   | 50 x 50    | 75.2566     | 76.1124     | 75.8136     | 0                      | 0.78%                 | 0.38%                 |
| Maze     | 50 x 50    | 193.2366    | 193.2366    | 193.2366    | 0                      | 0                     | 0                     |
| Warehouse| 50 x 50    | 93.3136     | 93.8994     | 93.6124     | 0                      | 0.63%                 | 0.32%                 |
| Random   | 100 x 100  | 140.9940    | 153.8816    | 146.9282    | 0                      | 9.14%                 | 4.21%                 |
| Maze     | 100 x 100  | 327.8046    | 327.8046    | 327.8046    | 0                      | 0                     | 0                     |
| Warehouse| 100 x 100  | 189.2130    | 189.7988    | 189.4825    | 0                      | 0.31%                 | 0.14%                 |

At run time, on the comparison of table 1 illustrates the environment, no matter in what kind of map RDJ algorithm in minimum time, maximum time consuming, the average time on the value is the smallest, the average speed is higher than DJ algorithm and A* algorithm for at least 10 times, especially with the expansion of map RDJ computational efficiency, the more obvious advantages, This fully proves the rationality of the algorithm in computational efficiency.

As for the path length, the RDJ algorithm has its own defects in searching the shortest path, which leads to the instability of the path length obtained in the process of searching the path. Therefore, the minimum length, maximum length and average length of the experimental results are listed respectively and compared with the shortest path obtained by DJ and A* algorithm. It can be seen from Table 2 that on relatively small scale maps (grid size 25 x 25, 50 x 50), the error of RDJ algorithm in the minimum, maximum and average path length is small, and the maximum error is only 1.27%. On a relatively large scale map (grid size 100 x 100), the error of path length obtained by RDJ algorithm is small in labyrinth and warehouse maps, but large in random environment, with the maximum error of 9.14%. From the perspective of the minimum error of different maps, RDJ algorithm can find the shortest path in the experimental map environment. Overall, the RDJ algorithm performed best in warehouses and mazes.

To sum up, compared with DJ and A* algorithm, RDJ algorithm has a qualitative improvement in operation time. Although the RDJ algorithm has errors in the path length, it is also within the acceptable range.

5. Concluding remarks

Based on the raster modeling method and the structural characteristics of raster map, this paper proposes a RDJ algorithm. The simulation experiment proves that the algorithm can effectively realize the real-time path planning of mobile robot in the working environment with large scale and dense obstacles. On the basis of this method, there are two aspects that need to be done in the next step: 1) to further improve
the algorithm to maximize the error control of the shortest path; 2) At present, the research on this algorithm is only in the software simulation stage. Next, it is necessary to use Pioneer robot [16] to run this algorithm in the actual working environment (such as warehouse) to verify its effectiveness.

References

[1] Raja P, Pugazhenthi S. Optimal Path Planning of Mobile Robots: A Review [J]. International Journal of Physical Sciences, 2012, 7(9) : 1314-1320.
[2] Huang Biao, Kadali R. Dynamic Modeling, Predictive Control and Performance Monitoring: A Data-Driven Subspace Approach [M]. Springer, 2008.
[3] Zhu Daqi, Yan Ming zhong. A Review of Path Planning Techniques for Mobile Robots [J]. Control and Decision, 2010, 25(7) : 961-967.
[4] Chen Yi-Zhou, Chen Shi-Fei, Chen Tao, et al. Path Optimization Study for Vehicles Evacuation Based on Dijkstra Algorithm [J]. Procedia Engineering, 2014, 71:159-165.
[5] Cui S G, Wang H, Yang L. A simulation study of A-STAR algorithm for robot path planning [C] // 16th International Conference on Mechatronics Technology. 2012: 506-510.
[6] Harabor D D, Grastien A. Online Graph Pruning for Pathfinding on GridMaps [C] // Proceedings of the 25th AAAI Conference on Artificial Intelligence. 2011: 1114-1119.
[7] Qiu Lei. Accelerating A* Pathfinding Using Hop Search Algorithm [J]. Journal of Lanzhou University of Technology, 2015, 41(3) : 102-107.
[8] Ammar A, Bennaceur H, Chaari I, et al. Relaxed Dijkstra and Complexity of Robot Path Planning Problems in Large-scale Grid Environments [J]. Soft Computing, 2015, 20(10) : 4149-4171.
[9] WANG Pei-dong, Tang Gong-You, LI Yang, et al. Endpoint approximation for robot path planning [C] // 2012 The 31st Chinese Control Conference. 2012: 4960-4965.
[10] Xu Xiang, Liang Ruishi, Yang Huizhi. Simulation of Intelligent Agent Path Planning Based on Improved Genetic Algorithm [J]. Computer Simulation, 2014, 31(6) : 357-361.
[11] Shiltagh N A, Jalal L D. Optimal Path Planning for Intelligent Mobile Robot Navigation Using Modified Particle Swarm Optimization [J]. International Journal of Engineering and Advanced Technology, 2013, 2(4) : 260-267.
[12] Yin Xiaohong, Ni Jianjun, Wu Liaoying. A Real-time Path Planning Method for Robot Based on Improved Artificial Bee Colony Algorithm [J]. Computer and Modernization, 2015(3) : 1-4.
[13] Alajlan M, Koubaa A, Chaari I, et al. Using Genetic Algorithms [C] // 2013. Using Genetic Algorithms [C] // 2013 International Conference on Individual and Collective Behaviors in Robotics. 2013: 1-8.
[14] Barbehenn M. A study on the complexity of Dijkstra's algorithm for graphs with weighted vertices [J]. IEEE Transactions on Computers, 1998, 47(2) : 263.
[15] Yang Lei, Song Tao. Bucket Sort Algorithm Based on Array [J]. Computer Research and Development, 2007, 44(2) : 341-347.
[16] Wu Dengfeng, Mei Zhiqian, Yin Liwei, et al. A New Path Planning Algorithm for Indoor Mobile Robot in Unknown Environment [J]. Mechanical and Electrical Engineering, 2015, 32(3) : 389-392.