AN EFFECTIVE ALGORITHM OF SHORTEST PATH PLANNING IN A STATIC ENVIRONMENT

Lingyu Sun[1], Xuemei Liu[2], Ming Leng[3]
[1]*Department of Computer Science, Jinggangshan College, Ji'an, China. Email:lmsly@263.net*
[2]*School of Mechanical & Electronic Engineering, Shandong Agricultural University, China*
[3]*School of Computer Engineering and Science,Shanghai University, Shanghai, China*

Abstract: Path Planning is generating a collision-free path in an environment with obstacles and optimizing it with respect to some criterion. Because of its importance in robotics, the problem has attracted a considerable amount of research interest. In this paper, we present an effective algorithm of shortest path planning for planar mobile robot whose time complexity is $O(4 \times n)$, n is the geometric complexity of the static planar environment. The success of our algorithm relies on exploiting both a tabu restriction and the greedy strategy of the *Dijkstra* algorithm. Our experimental evaluations on four different test cases show that our algorithm produces the shortest path very quickly.

Key words: path planning, greedy algorithm, *Dijkstra* algorithm

1. INTRODUCTION

During the last century, automation has become an extremely fast growing phenomenon impacting almost all facets of everyday life. Recently, robots have become a major part of this trend. Therefore, autonomously navigating robots have become increasing important [1]. The *path planning* problem is one of important problems in intelligent control of an autonomous mobile robot. It uses a prior information about a given environment and knowledge about robot motion capabilities to provide a path to be executed between two points in the robot workspace. The *path planning* problem is then an optimization problem where we want to find the best path under a set of constraints. Classical approaches to solve the *path*

Please use the following format when citing this chapter:
Sun, Lingyu, Liu, Xuemei, Leng, Ming, 2006, in International Federation for Information Processing (IFIP), Volume 207, Knowledge Enterprise: Intelligent Strategies In Product Design, Manufacturing, and Management, eds. K. Wang, Kovacs G., Wozny M., Fang M., (Boston: Springer), pp. 257-262.

planning problems include *Dijkstra* algorithm [2], *A* search* algorithm [3] and *dynamic programming* technique [4, 5].

The *Dijkstra* algorithm is an optimization algorithm that is mainly used for determining the shortest path. The *Dijkstra* algorithm is an uninformed search algorithm for finding shortest paths that relies purely on local path cost and provides a shortest path from a start node to a goal node in a graph. The *A* search* algorithm was developed by Hart et al. [6]. The algorithm uses a heuristic function $h(n)$ to estimate the cost of the lowest cost path from a start node to a goal node plus the path cost g(n), and therefore the search cost $f(n)=g(n)+h(n)$. Using f, g, and h values, the *A* search* algorithm will be directed towards the goal and will find it in the shortest possible route. The *dynamic programming* technique resorts to evaluating the recurrence in a bottom-up manner, saving intermediate results that are used later on to compute the desired solution. This technique applies to many combinatorial optimization problems to derive efficient algorithms and is also used to improve the time complexity of the brute-force methods [7].

In this paper, we present a Modified *Dijkstra* (MD) algorithm that is an $O(4\times n)$-time algorithm of shortest path planning for planar mobile robot in a static environment. Our work is motivated by the greedy strategy of the *Dijkstra* algorithm which consists of an iterative procedure that tries to find a local optimal solution. Furthermore, we integrate our algorithm with tabu search [8]. We test our algorithm on four test cases and our experiments show that our algorithm produces the shortest path very quickly.

The rest of the paper is organized as follows. Section 2 provides the problem formulation, describes the notation that is used throughout the paper. Section 3 presents an effective algorithm of shortest path planning. The proposed algorithm complexity is analyzed in section 4. Section 5 experimentally evaluates the algorithm.

2. PROBLEM FORMULATION

Our goal is to implement a shortest path planning system that uses the MD algorithm for planar mobile robot in a static environment. First, we use the traditional grid point representation to present a planar workspace for analysis. We consider the workspace for the planar robot that is subdivided into cells on a two-dimensional grid. A *grid node* which is located at a cell's centre-point is allocated to each cell as shown in Fig. 1(a). This method of space mapping generates a set of discrete nodes that cover the entire construction workspace domain, as shown in Fig. 1(b). Let *MaxRow* denote the total row of the two-dimensional grid. Let *MaxCol* be denoted as the total column of the two-dimensional grid. Naturally, the geometric complexity of

the planar environment, denoted by n, is equal to *MaxCol*MaxRow*. Some dark areas indicate the presence of some obstacles that labeled as $O_j(x,y)$ where x and y are the coordinates of the obstacle j. The robot is considered as a punctual object and its position is given by $R(x, y)$. Next, a trajectory of the robot is composed of a set of the adjacent grid nodes which excludes the *grid node* occupied by some obstacles. The path cost evaluation function $F(p)$ is the number of the adjacent grid nodes residing between the start and the goal nodes where p is the entire path.

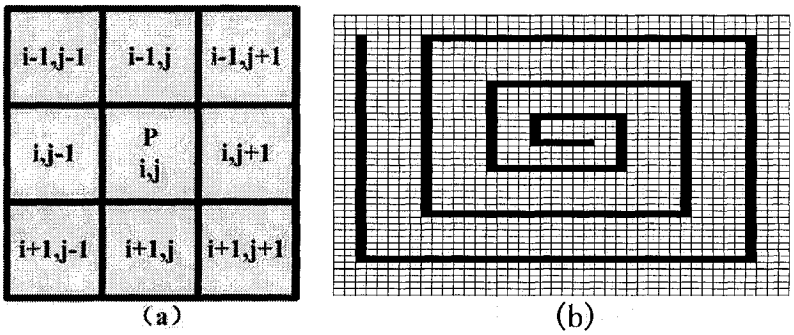

Figure 1. (a) Grid point representation, (b) A sample of a discretized site layout

3. AN EFFECTIVE ALGORITHM OF SHORTEST PATH PLANNING

The *Dijkstra* algorithm can find optimal solutions to problems by systematically generating path nodes and testing them against a goal. Because the time complexity of the *Dijkstra* algorithm is $O(n^2)$, it becomes inefficient when it applies to large-scale problems. The reason of high time complexity is that it is designed to find the shortest path on graphs in which each edge has a nonnegative length.

However, as we see in Fig. 1, each edge that straddles two adjacent grid nodes has the unit length. Our MD algorithm is motivated by the greedy strategy behind the *Dijkstra* algorithm which has been proved that it does indeed find the shortest path. In the greedy strategy, the *Dijkstra* algorithm tries to find a node whose distance from the start node is the length of a shortest path in each step of expanding leaf node. Because the cost of expanding leaf node is unit length in Fig. 1, MD algorithm just adopts the breadth-first strategy to traverse the two-dimensional grid that needn't

inspects every edge in each step. The time complexity of MD algorithms is small since only adjacent grid node is inspected exactly once by the algorithm. In the choice of adjacent *grid node*, the MD algorithm adopts simple tabu search strategy without aspiration criterion whose tabu restriction forbids expanding grid nodes which are designated as tabu status or traversed status. The pseudocode of the MD algorithm is given in Table. 1. The function *neighbors*(*v*) is used to iterate neighbors of grid node *v* in the two-dimensional grid.

Table 1. The pseudocode of the MD algorithm

```
INPUT:  1.MaxRow; 2.MaxCol;
        3.the list of obstacles O(x,y);
        4.a start node s; 5.a goal node g;
OUTPUT: 1.the shortest path p;
1  set non-traversed status in the MaxRow*MaxCol matrix for all grid nodes;
2  set tabu status in the matrix for grid nodes of O(x,y);
3.  initial empty queue and  insert s into queue;
4  while (the queue is non-empty) do {
5     v=pop(queue);
6     for (each u∈neighbors(v) ) do {
7          if (u's status==non-traversed status ) then
8               if (u==g) then {
9                    break;
10              } else {
11                   insert u into queue;
12                   set traversed status in the matrix for u;
13                   record v as the source of u;
14              }end if
15         }end if
16    }end for
17 }end while
18 if (u==g) then {
19       trace the source information to obtain the shortest path p
20 }end if
21 }
```

4. COMPLEXITY ANALYSIS OF MD ALGORITHM

The complexity of the algorithm is computed as follows. Step 2 costs $O(n)$ time. The while loop (Step 4-17) of MD is required to iterate *n-1* times in the worst case and the for loop (Step 6-16) of MD is required to iterate four times at most because the planar mobile robot moves in 4-neighbour mode. As result, the total time complexity of the algorithm is $O(4 \times n)$. In the whole process, the MD algorithm uses a queue with the first-in first-out scheme and a two-dimensional matrix to store the status of all grid nodes

whose size is n. In the worst case, the MD algorithm successively inserts $[n/2]$ grid nodes into the queue. Therefore, the total space complexity of the algorithm is $O(n*3/2)$.

5. EXPERIMENTAL RESULTS

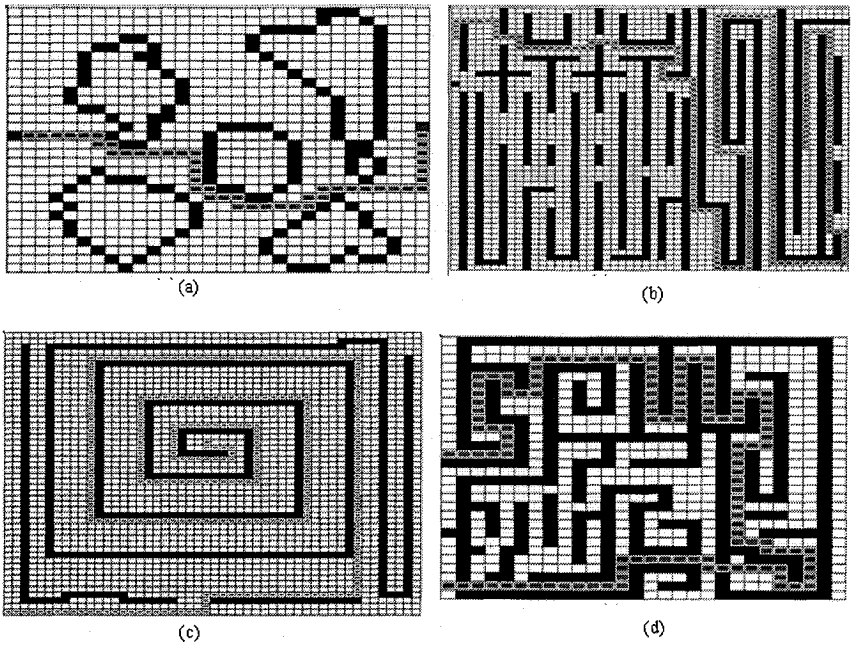

Figure 2. The result of our experimental evaluations on four different grids

The shortest path planning system is designed based on MFC programming technology. We use the four different grids in our experiments whose dimensional is 30 or 50 cells. The four evaluations are run on an 1800MHz AMD Athlon2200 with 128M memory. As expected, our experimental evaluations show that our algorithm produces the shortest path, as shown in Fig. 2, and its cost time is no more than 0.001 second.

6. CONCLUSIONS

In this paper, we have proposed an effective algorithm of shortest path planning whose time complexity is $O(4 \times n)$ and space complexity is $O(n*3/2)$. The success of our algorithm relies on exploiting both a tabu restriction and the greedy strategy of the *Dijkstra* algorithm. Although it has the ability to find the shortest path very quickly, there are several ways in which this algorithm can be enhanced. This brings up two questions about possible improvement. The first question is how to find the shortest path with multi-constraints. Our algorithm just uses to produce the shortest path in static environment for planar mobile robot. Therefore, the second question is how to produce a new shortest path in response to environmental changes.

7. ACKNOWLEDGMENTS

This work was supported by the Foundation of Ji'an Municipal Commission of Science and Technology, grant No. 2005-28-7.

8. REFERENCES

1. Sugihara, K., Smith, J.: Genetic Algorithms for Adaptive Motion Planning of an autonomous Mobile Robot. *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, Monterey, CA(1997) 138-146
2. Dijkstra, E.: A note on two problems in connection with graphs. *Numerical Mathematic,* Vol. 1 (1959) 71-269
3. Russell, S., Norvig, P.: *Artificial intelligence: a modern approach.* Prentice-Hall Publications, New Jersey (1995) 230-247
4. Sedighi, K.H., Ashenayi, K.., Manikas, T.W., Wainwright, R.L., Tai, H.M.: Autonomous Local Path Planning for a Mobile Robot Using a Genetic Algorithm. *IEEE International Conference on Systems, Man and Cybernetics* (2004) 1338-1345
5. Xuemei Liu, Jin Yuan, Kesheng Wang. "A Problem-specific Genetic Algorithm for Path Planning of Mobile Robot in Greenhouse". *Proceeding of PROLAMAT 2006.* (In press)
6. Hart, P., Nilsson N., Raphael B.: Correction to 'A formal basis for the heuristic determination of minimum cost paths'. *SIGART Newslett,* Vol. 37 (1972) 9-28
7. Alsuwaiyel, M.H.: *Algorithms Design Techniques and Analysis.* World Scientific Publications, Singapore (1999) 203-205
8. Glover, F., Manuel, L.: Tabu search: *Modern heuristic Techniques for Combinatorial Problems.* Blackwell Scientific Publications, Oxford (1993) 70-150