Multi-Goal Path Planning Algorithm for Mobile Robots in Grid Space

Liu Hongyun, Jiang Xiao, Ju Hehua

Beijing University of Technology, Beijing 100124, China
E-mail: jiangxiaotwn@hotmail.com

Abstract: In accordance with the problem which ingle target path planning is difficult to simulate the complex real world, this paper presents a new multi-goal path planning method based on grid map for planetary exploration, which designs and realizes the corresponding branch-detected algorithm and heuristic algorithm to solve the local optimal path planning and TSP, eventually reach the target state. The theoretical proof and simulation results show that this algorithm can successfully solve path planning with multi-goal and with a good performance at the same time.

Key Words: Path Planning, Multi-Goals, Branch-Detected, TSP

1 INTRODUCTION

In last few years, space exploration has become an important subject internationally. The recent planetary exploration, Mars Exploration Rover (MER) [1] mission has become a huge success, and achieved a tremendous scientific ambition. In MER mission, MAPGEN [2, 3] was deployed as a key component of the ground operation system. From the referenced paper, we can see that MAPGEN is based on a mature combination of constraint reasoning technology, and planning technology. The system consists of APGEN and EUROPA. APGEN is a human machine interface that can handle the input of constrains. EUROPA is the core planner based on D* algorithm [4], which uses a Cartesian grid of eight-connected cells to represent the map. However, single target path optimization in MER is difficult to simulate the complex surface of the planet. In contrast, multi-goals path optimization is more close to reality, has more practical affection on engineering problems.

There are literatures full of elegant solutions to the multi-goals path planning problem. Alexander Stepano [5] uses integer programming to solve Multi-objective evacuation routing in transportation network. Line Blander Reinhard [6]adopt Dynamic planning method to figure out the optimal path with Multi-objective and multi-constrain. Jean-Francois ERUBE [7] solves the Traveling Salesman Problem by heuristic algorithms. This is the approach we will take in this paper. In Vehicle Routing Problem (VRP), P.Lacomme [8] constructs the heuristic algorithms to solve multi-goals VRP from the idea of Non-dominated sorted genetic algorithm. Ant Colony Optimization (ACO)is a new simulated evolutionary algorithm, first proposed by M.Dofigo [9]. Different from other meta-heuristic algorithms, ACO takes the process of distributed construction. Whenever each ant choose the next step, it needs to refer to the accumulate experience and heuristic information of the whole population. In accordance with bi-objective shortest path problem, Keivan Ghosei [10] puts forward an improved multi-goal ACO. FEO [11] presents Greedy Randomized Adaptive Search Procedure (GRASP) which contains two phases in iteration: construction phase and local planning phase. This provides very useful ideas for this paper.

Our working team has been doing research in the field of planetary rovers for several years. In our research, we use Tour Construction Procedures to construct a graph, followed by Tour Improvement Procedure to pursue the optimal path. Branch-detected algorithm is introduced to calculate the local optimal path, which means the optimal path between two nodes. Considering the practical requirement in planetary exploration, we use heuristic algorithm to solve Traveling Salesman Problem (TSP) in global planning. Through the proof of theorem, we solve combination problem of global planning and local planning. After the complexity analysis, we finish the simulation experiment.

In this paper, we present a branch-detected algorithm in chapter 2, as while as a heuristic algorithm for Traveling Salesman Problem and demonstrate the algorithm in grid map. In chapter 3, we design a multi-goal path planning algorithm processes, calculate the time complexity and prove the admissibility. Considering the successful experience above, we accomplish the emulation test in chapter 4.

2 PATH PLANNING

The purpose of robot path planning is to fix a path line that achieves a goal state while avoiding obstacles. Algorithms of path planning usually adopt the following three basic methods: cell decomposition, roadmap methods, and potential field techniques. Planetary rover is a kind of specific mobile robot, whose work environment is extremely complicated since it has to be on a planetary surface. We use texture pictures taken by panoramic camera to describe the world for planning, from which an obstacle map can be extracted. The map, including obstacles, is used for two-dimensional space. It is convenient to use grid map to express the extraction. So our search is based on the grid map.

2.1 branch-detected algorithm

In this paper, we use branch-detected algorithm to plan local optimal path. This algorithm is inspired by the path finding process of true animal in nature, and improved to

This work is supported by National Nature Science Foundation under Grant ××××××××

solve various barrier problems. Through contrast test, the efficiency is much higher than that of A* algorithm.

Definition:

1 Searching node: the node searching toward to the goal. The starting searching node is the start node.

2 Free searching node: when a searching node is searching for the goal, if its next grid is not blocked, it can go to next node freely.

3 Branched searching node: when a searching node comes to a blocked grid, it will branch and get around it.

The process of algorithm is as follows:

Step 1: From the start point, Searching node is a free searching node, go forward to the goal point.

Step 2: Judge whether the next grid is blocked when Free searching node goes.

Case 1: Next grid is not blocked, the node goes ahead one step, and it is still a free searching node.

Case 2: Next grid is blocked, the node branch to two nodes trying to get around the blocked grid, and these two nodes are branched searching node.

Step 3: After the Branched searching node get around the blocked grid becomes the free searching node, and then returns to step 2.

Step 4: Go into next grid and judge whether it is the goal point. If so, stop searching, else return to step 2.

Then we use the visible detecting method [12] to optimize the path, which is mentioned below:

When we calculate the distance, we scan whether there is an obstacle between two points. If there are no obstacles between two points, then we use the visible Euclidean distance to displace the grid distance. Then we will discard the nodes between these two points, which is also reduced the burden of the memory storage. We call this process Cut Cost. This greatly improves the overall efficiency of planning. Once the cost of a path from the initial state to the goal has been calculated, the path and its work mode are extracted. Because of our prediction technique, it is possible to compute the path cost of any point based on a grid cell, which is useful for both path planning and mission planning. Unlike paths produced using classic grid-based planners, paths produced using an improved D*lite are not restricted to a small set of headings. As a result, the improved D*lite provides smoother, lower-cost paths through both uniform and non-uniform cost environments.

All in all, in normal planning, we will find a path $A \to B \to C \to D \to E \to F$ as we can see in Figure 1. However the addition of the idea of recursion changes this situation. First, we deduce a path $A \to B$ as A is a start point, we get a path $A \to B \to C$ at this time, and we need judge the environment according the visuality. If A can 'see' C without any obstacle, we deduce the path $A \to C$ finally. If not, we use C as a new start point. We repeat this process until we get the goal.

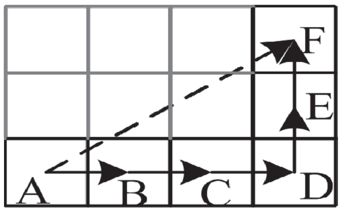

Figure 1. The grid distance and the Euclidean distance

The following shows a fragment of detecting function:

```
while (x0 != x1) {
    f = f + dy;
    if (f >= dx) {
        xt = x0 + (sx - 1 >> 1);
        yt = y0 + (sy - 1 >> 1);
        if (m[xt][yt]) return false;
        y0 = y0 + sy;
        f = f - dx;
    }
    xt = x0 + (sx - 1 >> 1);
    yt = y0 + (sy - 1 >> 1);
    if (f && m[xt][yt]) return false;
    if (!dy && m[xt][y0] && m[xt][y0-1])
        return false;
    x0 = x0 + sx;
}
```

In the process, we make the realization of all integer operations, and avoid the multiplication and division operations，which improves operation efficiency greatly.

2.2 Traveling Salesman Problem (TSP)

Suppose a salesman is required to travel between a set of cities linking with distances (or costs) from any city to any other one. This salesman is asked to accomplish a round-trip tour with the minimal distances (or costs). The round-trip tour requires that the salesman must visit every city only once and at last return to the starting city. The traveling salesman problem is therefore to establish a Hamiltonian tour with minimum cost. The TSP is one of the discrete optimization problems which are classified as NP-hard [13]. Generally, TSP can be denoted via graph notations as follows [14]:

• Let G = (V, E) be a graph.

• V is a set of n cities, $V = \{v_1, \ldots, v_n\}$.

• E is a set of arcs or edges.

$E = \{(r, s) : r, s \in V\}$.

E is normally associated with a distance (or cost) matrix, which is defined as $D = (d_{r,s})$.

If $d_{r,s} = d_{s,r}$, the problem is a symmetric TSP (STSP). Otherwise, it becomes an asymmetric TSP (ATSP). In TSP, our objective is to determine the permutation π of set V that minimizes Equation:

$$C(\pi) = \sum_{i=1}^{n-1} d_{\pi(i),\pi(i+1)} + d_{\pi(n),\pi(1)} \qquad (1)$$

There are many methods to solve TSP, such as genetic algorithm, Ant Colony Optimization, heuristic algorithms, simulation annealing algorithm, tabu search algorithm, etc.

To guarantee the admissibility of algorithm as project needed, we adopt A*'s heuristic theorem. Every element in OPEN list represents a cost of node array, not just a cost of node in common.

We define the H(n)=MIN*LeftNodeNum, where LeftNodeNum denotes to the number of node which have been reached, and h denotes the real cost between nodes.

The admissibility of this algorithm is apparent:

1. The nodes in goal set are limited, so the elements (node array) in searching space are limited.
2. The distance between any two nodes in the graph is not negative, so the cost of element in searching space is not negative.
3. $H(n)=MIN*LeftNodeNum \leq h$

So the admissibility of the algorithm is proved. In addition, as the iteration of matrix M goes on (we present in chapter 3), the value of MIN is changing as well, which promotes the sensitivity of the algorithm.

2.3 TSP in grid map

With the development of the research in these years, TSP can be divided into four categories:

1. Start from a certain node, return to the start;
2. Start from a certain node, not return to the start;
3. Start from a uncertain node, return to the start;
4. Start from a uncertain node, not return to the start;

The research work in this paper belongs to the second category; the robot sets out from the start point, visits all the goal points, and need not return to the start point. However, in multi-goal path planning based on grid map, the cost of any two nodes is unknown before robot moving, which is different from the traditional Traveling Salesman Problem.

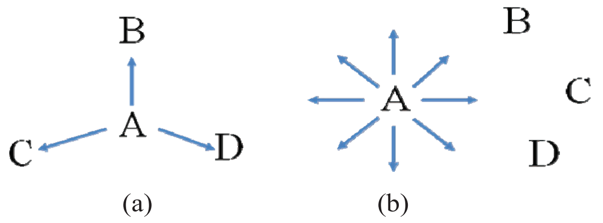

Figure 2. Difference between topological map and grid map

From Figure 2, we can see that in topological map, the number of path from start A to other goals is definite; the cost of the paths will be made clear. In contrast, the robot has a higher degree of freedom for every pace in the grid map, so determining the robots heuristic frontage is our first task. For example, if we have fixed the geometric center of the set of goal points, can we obtain the optimal path if we order the robot to move towards the geometric center, or just get close to the nearest goal point?

To solve this problem, we prove the following theorem:

Theorem:

Heuristic Target of searching node must be the element of goal set, which is the sufficient condition of finding the optimal path.

Definition:

Heuristic Target: the goal of searching node in 8-neighbours-expansion.

Move (a->b): Search an optimal path from node a to heuristic target b and arrive.

S: Start node.

g_i: The goal node with serial numbers i.

$G_i [g_1,g_2,...,g_i]$: The set of goals with i elements.

$G_i []/g_k$: G_i with the element g_k removed.

Prove:

1. When there is only one element in goal set, from the Linear axiom we can know that Move(S->$G_1 [g_1]$) is true
2. When there are two elements in goal set,

Move (S->$G_2 [g_1,g_2]$)=Move(Move(S->g_i)->$G_2[]/g_i$) is true

3. According to Mathematical induction, when Move(S->G_n[]) is true, we can deduce

Move(S->G_{n+1}[])=Move（Move(S->g_i)->$G_{n+1}[]/g_i$) is true.

So the proposition is permitted.

This theorem guarantees that the multi-goals planning in the grid map is on the framework of TSP. According to the theorem, we can design our program through global planning and local planning two parts.

3 MULTI-GOALS HEURISTIC SEARCH ALGORITHM

3.1 Basic flow

Through the discussion above, we design our program as follows:

Step 1: Load the grid map, set up start node and the set of goal nodes G_i [],

Step 2: According to the coordinates of nodes (start node and all the goal nodes), calculate the Euclidean geometry distances and establish the distance matrix M [].

Step 3: Calculate the optimal array S_0, S_1,..,S_n marked as OLD.

Step 4: Mark the minimum element in M [] as MIN, calculate the real cost between two nodes of optimal array old [S_0,S_1] [S_1,S_2],...[S_{N-1},S_N].

Step 5: Put the new cost element into the matrix M [] and change the former element.

Step 6: Calculate the optimal array again according to the new M [], and mark the new optimal array as NEW.

Step 7: Compare the array OLD and NEW. If matched, it is the optimal path and output it. If not, make OLD = NEW, and return to step 4.

The flowchart is presented in Figure 3.

3.2 Complexity analysis

For the reason that the cost of any two nodes is unknown at the initial stage, we reference the idea of Composite Procedure: get a path of approximate optimal solution first from the distance matrix and then gradually improve it by iterative method. We can calculate the time complexity of this algorithm as following:

Define the time complexity of computing TSP as T, time

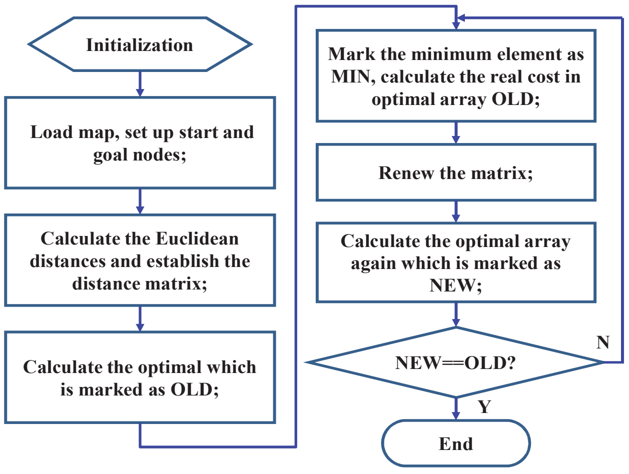
Initialization
Load map, set up start and goal nodes;
Calculate the Euclidean distances and establish the distance matrix;
Calculate the optimal which is marked as OLD;
Mark the minimum element as MIN, calculate the real cost in optimal array OLD;
Renew the matrix;
Calculate the optimal array again which is marked as NEW;
NEW==OLD?
N
Y
End

Figure 3. The flowchart of algorithm

complexity of branch-detected as A, so for a graph with N nodes(one start node and n-1 goal nodes), containing access(between two nodes)(N-1)N/2, node array (include all nodes)(N-1)!, the chance for finding the optimal array in every iteration is 1/(N-1)!.

To compare with the algorithm we design above, we calculate every access cost first, and find the optimal array, from which the time complexity is

$$T+\frac{A\times(N-1)N}{2} \tag{1}$$

The time complexity of improved algorithm is as following

1. Chance for finish in the first iteration

$$[T+(N-1)A]\times\frac{1}{(N-1)!} \tag{2}$$

2. Find the optimal array for at most [N/2] iteration, the chance is

$$[(\frac{N}{2}+1)T+\frac{N}{2}(N-1)A]\frac{1}{(N-1)!} \tag{3}$$

The Expectation of time complexity is

$$\sum_{i=1}^{i=[\frac{N}{2}]}[i\times T+i\times(N-1)A]\frac{1}{(N-1)!}=(\frac{T}{2}+\frac{N-1}{2}A)\frac{N(N+2)}{4(N-1)!} \tag{4}$$

Compare (1) and (4) we can deduce, when N>3, the improved algorithm has less time complexity.

At the worst situation, the improved algorithm complete searching at the last iteration, complexity is

$$(\frac{N}{2}+1)T+\frac{N}{2}(N-1)A \tag{5}$$

Compare with (1), the increment in computing reflects in the TSP. Considering the practical constraint such as energy and communication in planetary exploration, The number of points in goal set is less than 5 in most cases, the result is acceptable.

4 EXPERIMENT RESULTS

Based on the descriptions above, we program the simulate experiment in software platform of Visual C++ 6.0 and Open Inventor 5.0.We run it in hardware platform of AMD Dual Core 2.71GHz CPU and 4GB RAM. The figures we present below illustrate algorithm flow of branch-detected in detail.

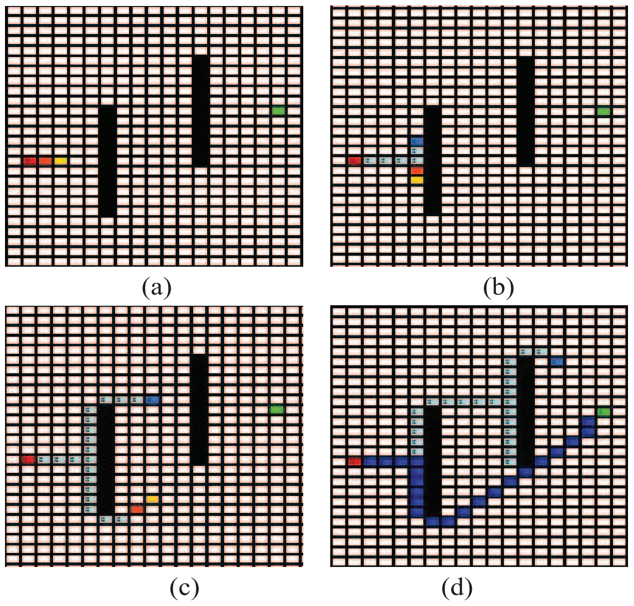

Figure 4. Algorithm flow of branch-detected

After the test, we summarize the performance comparison of branch-detected and A* in the table.

Table 1. The number of expanded grid in 5 seconds

	No block	Linear block	Ring block	Closed block
A*	5341	2351	672	468
branch-detected	19357	8247	4315	8641

In Figure 5, we present the simulation experiment results with different number of goals.

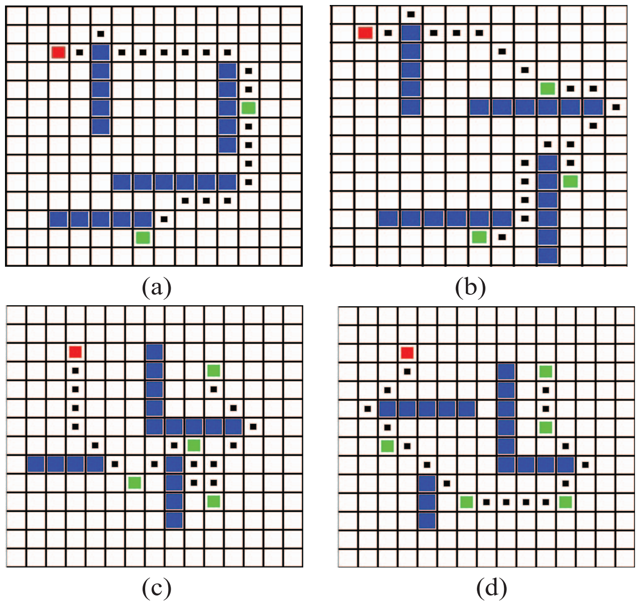

Figure 5. Simulation experiment results

For the above results of multi-planning, we summarize the execution time and path length in the table:

Table 2. Algorithm performance

Number of goals	Execution time(ms)	Path length
2	32	23.52
3	46	27.62
4	55	25.46
5	78	22.78

5 CONCLUSION AND FURTHER WORK

In this paper, we referenced other algorithms of TSP to structure and optimized our program process, which applied to the path planning in the grid map environment, in which we built a multi-goal, non-loop planning order of the guide lines, and according to the actual path data, we renewed the distance matrix as well as the optimal node sequence. Through theoretical proof and simulation experiment, the algorithm works with good stability, is able to search the optimal path efficiently. In further work, we will transform the 2D environment into three-dimensional environment. Based on the theory of the algorithm, transform path constraint to time constraint, achieve the multi-goals mission planning.

REFERENCES

[1] M. Ai-Chang, J. Bresina, L. Chares, MAPGEN: Mixed-Initiative planning and scheduling for the mars exploration rover mission, IEEE Trans. on Intelligent Systems, Vol.19, No.1, 8-12, 2004.

[2] D. P. Roland, Space mission operations DBMS (SMOD), Second IEEE International Conference on Space Mission Challenges for Information Technology, 214-221, 2006.

[3] J. L. Bresina, P. H. Morris, Mission operations panning: beyond MAPGEN, modeling interplanetary logistics, Second IEEE International Conference on Space Mission Challenges for Information Technology, 2006.

[4] Anthony Stentz, The focussed D* algorithm for real-time replanning, In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1995.

[5] S. ALEXANDER, M. JAMES, Multi-objective evacuation routing in transportation networks, European Journal of Operational Research, Vol.198, No.2, 435-446, 2009.

[6] B. R. LINE, P. DAVID. Multi-objective and multi-constrained non-additive shortest path problems, Computers&Operations Research, Vol.38, No.3, 605-616, 2011.

[7] J. F. ERUBE, G. MICHEL, P. JEAN-YVES, An exact e-constraint method for bi-objective combinatorial optimization problems, Application to the Traveling Salesman Problem with Profits. European Journal of Operational Research, Vol.194, No.1, 39-50, 2009.

[8] P. LACOMME, C. PRINS, M. SEVAUX, A genetic algorithm for a bi-objective capacitated arc routing problem, Computers&Operations Research, Vol.35, No.9, 3473-3493, 2008.

[9] M. DORIGO, V. MANIEZZO, A. COLORNI, The ant system: optimization by a colony of cooperating agents, IEEE Transactions oil Systems, Man, and Cybernetics Part B, Vol.26, No.1, 29-41, 1996.

[10] G. KEIVAN, N. BEHNAM, An ant colony optimization algorithm for the bi-objective shortest path problem, Applied Soft Computing, Vol.10, No.4, 1237-1246, 2010.

[11] T. A. FEO, M. G. C. RESENDE, A probabilistic heuristic for a computationally difficult set covering problem, Operations Research Letters, Vol.8, No.4, 67-71, 1989.

[12] S. Y. Dong, H. H Ju, H. X. Xu, An improvement of D* lite algorithm for planetary rover mission planning, International Conference on Mechatronics and Automation (ICMA), 1810 - 1815, 2011.

[13] G. Laporte, The traveling salesman problem: an overview of exact and approximate algorithms, European Journal of Operational Research, Vol. 59, No. 2, 231-247, 1992.

[14] P. W. Li, M. Y. H. Low, C. S. Chong, A bee colony optimization algorithm for traveling salesman problem, Second Asia International Conference on Modeling & Simulation, 818-823, 2008.