Proceedings of the 43rd Chinese Control Conference
July 28-31, 2024, Kunming, China

Method of Multi-agent Path Planning for Lunar Robots Swarm Based on Improved CBS Algorithm

Haolong Feng[1,2], Songtai Wu[1,2], Shengyang Liu[1,2], Ting Song[1,2,3], Fei Han[1,2]

1. Shanghai Aerospace Control Technology Institute, Shanghai 201109, China
E-mail: feng_hao_long@163.com

2. Shanghai Key Laboratory of Aerospace Intelligent Control Technology, Shanghai 201109, China
E-mail: feng_hao_long@163.com

3. Northwestern Polytechnical University, Xi'an 710072, China
E-mail: songtinghit@163.com

Abstract: As lunar missions become increasingly complex, the mission terrain will vary depending on different operating areas. In the future, there may be multiple lunar rovers working together, a more flexible and efficient algorithm to generate mission paths for lunar rover clusters is required. To address the path planning problem of lunar rover groups in large-scale and complex lunar terrain, this paper constructs a gridded map based on lunar image information and the actual lunar surface image is woven into a ring map. Path planning algorithms are modeled on this map, and the actual paths are projected onto the original lunar surface and into missions for implementing multi-lunar rover path planning tasks based on the conflict based search(CBS) algorithm. As the key to CBS algorithm is to resolve conflicts among agents, this paper combines the Deep Q Network with the CBS algorithm to propose a CBS algorithm based on a DQN conflict choice strategy to solve the multi-agent path planning problem for lunar rover clusters. The DQN can learn the policy of an agent, enabling it to better select paths. By integrating DQN with the CBS algorithm, path planning can become more intelligent and efficient. Finally, the simulation verifies that the CBS algorithm can effectively solve the multi-agent path planning problem and the improved CBS algorithm proposed has a certain degree of improvement in terms of solution efficiency.

Key Words: lunar robots swarm, multi-agent systems, path planning, CBS algorithm

1 Introduction

Lunar robots play an essential role in exploring the lunar environment and gathering data[1], making them currently the most direct and effective tools for close-range lunar exploration. The lunar terrain is complex, increasing the complexity of the tasks that lunar robots face. In the future, there may be multiple lunar robots working collaboratively on exploration tasks. Given the complex task environment for lunar robots, it's necessary to quickly plan safe, efficient, and reasonable exploration paths for multiple lunar robots to ensure the successful implementation of their exploration missions. However, the method most commonly used in engineering is to manually designate patrol and exploration routes which leads to low efficiency and a heavy workload for personnel. Algorithms for the automatic searching of paths also struggle with solving large-scale, complex map problems and are time-consuming. Concurrently, most path-planning algorithms for lunar robots can only cater to a single lunar robot, and are inadequate to simultaneously plan safe and effective paths for multiple lunar robots.

Currently, there are numerous path planning algorithms for individual lunar robots, including A* algorithm, Genetic algorithm, Particle Swarm algorithm, Ant Colony algorithm[2-3], etc. Liu et al.[4] improved the heuristic function of the Ant Colony algorithm based on the Artificial Potential Field method, introducing a method of spatial pheromone division to achieve shortest path planning for lunar robots. Zhao[5]'s research involves an improved Ant Colony algorithm for global path planning of lunar robots. This algorithm incorporates parameter self-adjustment and bidirectional search strategy in parallel, enhancing the success rate of ant path searching, whilst making corner processing of the path to provide a smoother and safer global route. Song et al.[6] designed a mixed-scale Ant Colony planning method. By gridizing the lunar images, this method models the path planning algorithms, effectively simulating the actual lunar surface with good generalization for different tasks. Yu et al.[7] proposed a large-scale autonomous fast and safe path planning algorithm based on the lunar digital elevation map, ensuring the generated path stays as far from danger zones as possible, thereby enhancing the safety of lunar patrol robot autonomous exploration processes. Wang et al.[8] employed an improved A* algorithm to generate auxiliary lines and proposed an algorithm optimized based on these lines. By introducing global map information to improve the original algorithm, a shorter pathway was designed.

The path planning algorithm for multi-agent systems is a classic search problem in the field of intelligent algorithms, and the conflict based search algorithm for path planning tasks of multiple lunar robots has received considerable attention. The CBS algorithm is an efficient path planning method. It decomposes the problem into independent subproblems, allowing each agent to make decisions independently without the need for a centralized planning process. It is also capable of adapting to hundreds or even thousands of agents, as each agent makes decisions independently without requiring a centralized planning process. Numerous studies and reviews on multi-agent path

*This work is supported by the Natural Science Foundation of China under Grant 62233005 and U20B2056, and the Natural Science Foundation of Shanghai under Grant 22ZR1427800.

planning search problems already exist[9-11]. Sharon et al.[12] utilized the Incremental Cost Tree (ICT) search method and proposed the concept of "two-layer search", where lower-level agents search for paths under the costs specified by higher-level agents. They subsequently based their research on this "two-layer search" concept, leading to the development of the Conflict-Based Search (CBS) algorithm[13]. Wang et al.[14] have summarized the research results of the CBS algorithm and classified it into four types: improvement of the division strategy, heuristic algorithm, handling of typical conflicts, and sub-optimal algorithms. Wang et al.[15] improved the CBS algorithm, adopting new conflict selection strategies based on conflict sub-nodes. They trained a ranking model using the RankNet algorithm and used the trained model to select conflicts for the CBS algorithm, effectively improving the efficiency of path planning. Wang[16] further improved the CBS algorithm and proposed a heuristic meta-agent path planning algorithm based on high-coupling multi-conflict as well as a fast bounded sub-optimal algorithm based on conflict search that guaranteed completeness, significantly enhancing search efficiency. In researching sub-optimal solution algorithms of CBS problems, the WA* algorithm was introduced into the heuristic with weighting, leading to sub-optimal solution paths[17]. Chan S.H.[18] and others proposed the Elastic ECBS algorithm, which dynamically adjusted the paths of intelligent agents by integrating sub-optimal solutions with the difference in path cost sum. Rahman M.[19] and others proposed the ASB-ECBS algorithm, which, depending on the number of conflicts per intelligent agent, assigned weights and specific dynamic sub-optimal solutions, thereby improving the efficiency of path planning in environments with narrow roads and more intelligent agents. The solution methods for multi-agent path planning can be divided into centralized and distributed methods. In the distributed method, each agent has independent computational capabilities and communicates with each other to share information. In the centralized method, there is a single central processor planning paths for all agents. Methods that have distributed computational capabilities but use a central solver to control all agents also fall into the centralized category. CBS is a hybrid algorithm for centralized methods, considering the global search space and focusing on single-agent searches at the search level.

Based on the path planning requirements of lunar rover groups, considering the constraints on path planning by these groups, a path planning task for multiple lunar rovers is implemented using the conflict-based search algorithm. The key to conflict-based search is how to resolve conflicts among agents. This paper combines the Deep Q Network with the CBS algorithm, proposing a CBS algorithm based on a DQN conflict choice strategy to solve the multi-agent path planning problem for lunar rover clusters. Through simulation verification, the CBS algorithm can effectively solve the multi-agent path planning problem, and the improved CBS algorithm proposed has a certain degree of improvement in solution efficiency.

2 Cluster-based Distributed Path Planning Modeling

The path planning problem for lunar robots involves planning the shortest safe path based on the task execution environment information, ensuring the path is as far from obstacles as possible and its length is minimized. Generally, path planning falls into two categories: global and local. Global path planning involves static path planning based on known global map information for environmental modeling. On the other hand, local path planning refers to dynamic path planning through real-time collection of local range information under circumstances where only local map information is accessible, typically used for local obstacle avoidance. The multi-lunar robot path planning problem studied in this article falls under global path planning, where the path planning is based on a global environmental model and implemented through a two-dimensional grid method.

The grid method is a simple and effective approach. It can handle irregular obstacles and is also applicable for three-dimensional environments. By employing the grid method, the lunar robot mission environment map was gridized. Selecting an appropriate grid size guarantees the accuracy of the environmental model while accelerating the speed of path planning. According to potential landing areas for the Chang'e 6 mission, the region was gridized and a grid map was generated based on gray-scale information from images. The process of gridizing the environment can be seen in the figure 1-4 below.

Figure1: Real image of the landing area

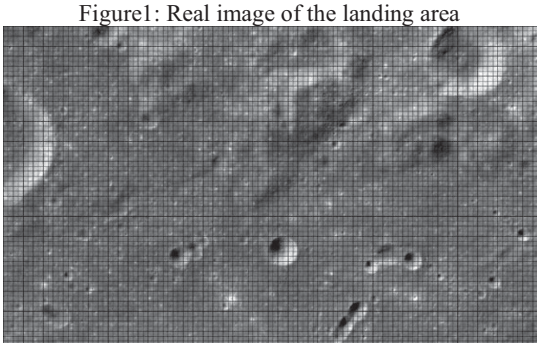

Figure2: Rasterization of real images in the landing area

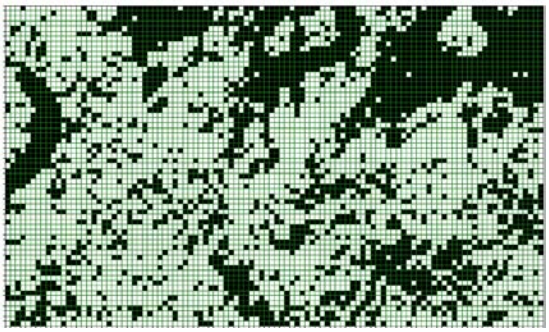

Figure3: Grayscale processing procedure

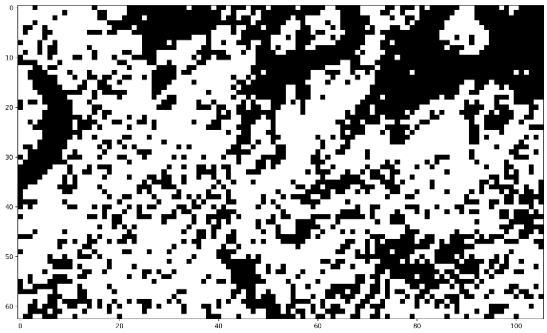

Figure4: The final resulting path planning environmental map

China's CE-6 mission has chosen to land in the southern part of the 490 km diameter Apollo peak-ring basin that formed inside the SPA basin. Figure 1 shows the data image of the potential landing area of Chang'e-6, and the area displayed is Area B referenced in [20]. Figure 2 is a schematic representation of the gridding process of the real image in Figure 1. Figure 3 is a map obtained after adjusting the grayscale information of the grid real image by choosing a threshold. Figure 4 is the processed map, which will be used in subsequent algorithm verification. The black areas represent moon pits or lunar surface protrusions, while the white areas represent zones that lunar robots can pass through.

3 Multi-agent Path Planning Algorithm

Multi-Agent Path Finding (MAPF) is a classic search problem in various fields, including robotics, gaming, and AI, and has many variations. Different MAPF research papers carry different assumptions and objective functions. Generally, a set of Agents and their respective start and goal positions are given, and the challenge is to identify the set of paths that guide each Agent to their goal with the lowest cost, while preventing collisions between the Agents and other obstacles. The definition of MAPF is as follows:

Consider an undirected graph $G = (V, E)$, and a set of Agents, denoted by $A = \{a1, a2 \dots ak\}$, numbering k. Each Agent ai has a starting vertex si and a goal vertex gi. Time is discretized into time steps, during which an Agent can move to an adjacent vertex or remain static at the same vertex. The objective of MAPF is to find a set of non-conflicting paths for the k Agents from their starting vertices to their goal vertices. Let pi represent the path of Agent ai. A feasible solution to MAPF is the set $P = \{p1, p2 \dots pk\}$ of paths for the k Agents. The duration taken for Agent ai to move from its starting position to its goal position is denoted as $t(i)$.

Definition 2.1 Conflict: A conflict arises when the paths of two Agents, ai and aj, occupy the same vertex v or the same edge e at the same time step t. A conflict that occurs on a vertex is denoted as (ai, aj, v,t). A conflict that occurs on an edge is denoted as (ai, aj,e,t).

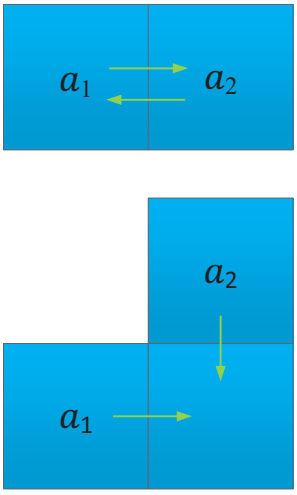

Figure 5: Example of Vertex Conflict and Edge Conflict

Figure 5 provides a diagram illustrating edge and vertex conflicts in a 4-neighbor grid. The paths of $a1$ and $a2$ are shown as solid lines, with arrows indicating the direction of Agent movement. In the top figure, the paths of $a1$ and $a2$ swap positions in the same time step in opposite directions. In the bottom figure, the paths of $a1$ and $a2$ pass through the same position in the same time step.

3.1 Principle of the A* Algorithm

The A* algorithm is a classic path search algorithm for single agents that is also applicable for multi-agent path planning. The A* algorithm uses a heuristic function to guide path finding, ensuring an optimal path is efficiently identified. The basic idea is to determine the search direction through an estimation function, expanding towards the target point. Then, the evaluation function is used to calculate the cost of each node, adding it to the Openlist, and selecting the neighboring node with the lowest cost as the next expansion node. This process repeats until the target point is added to the Openlist, and the final path is then generated in reverse via the parent-child relationships of the nodes.

The principle of the A* algorithm revolves around designing a cost estimation function:

$$f(n) = g(n) + h(n) \tag{1}$$

Where, the evaluation function $f(n)$ gives an estimate of the lowest cost path from the start node to the target node via node , where the function $g(n)$ represents the actual cost of the path from the start node to node n , and $h(n)$ represents the estimated cost of the potential best path from node n to the target node. The function $h(n)$ represents heuristic information used by the algorithm and can be adjusted to modify the search strategy of the A* algorithm. Based on the function $f(n)$, the cost of the current node can be calculated and the next accessible node can be evaluated. The search adopts the strategy of finding the node with the lowest cost each time as the new starting point, then continues the search and finds the optimal path.

However, when the A* algorithm is employed to solve the MAPF problem, it encounters issues with an excessive search space and a high branching factor, both of which increase at an exponential rate as the number of Agents increases. Even though A*-based algorithms can be expected to yield the optimal solution, the time required for

their execution and the amount of memory they occupy are often unacceptably large.

3.2 Conflict-Based Search Algorithm

A common algorithm to solve the MAPF problem is the Conflict-Based Search (CBS). The Conflict-Based Search is a two-level search algorithm, consisting of high-level and low-level searches. In the low-level search, we utilize the A* search to find the shortest pathway for each agent within a set of constraints. Note that the low-level search does not consider conflicts between agents, which is the responsibility of the high-level search. The high-level search adopts the paths returned from the low-level search, checks for conflicts between agents, and adds new constraints when the paths lead to conflicts. The algorithm alternates between low-level and high-level searches until a conflict-free solution is found.

The CBS algorithm is a two-tier algorithm where the high-level detects whether there are conflicts in all paths, while the low-level uses the A* algorithm to decouple and find the optimal path for agents under constraints. The high-level searches the constraint tree, which is a binary tree. Each node in the constraint tree is composed of the following information:

1) Constraint Set: The constraints are derived from conflicts, and each constraint is only related to one agent. Constraints can be categorized into vertex constraints and edge constraints. A vertex constraint is represented by the tuple <ai,v,t>, indicating that ai cannot appear at position at time ; an edge constraint is represented by the tuple <ai,u,v,t>, indicating ai is not allowed to move from position to position at time step .

2) Solution: The solution consists of k paths, which correspond to k agents. These solutions must satisfy the constraints of the current node, albeit they don't necessarily have to be feasible.

3) Conflict Set: All conflicts existing in the solutions.

4) Total Path Cost: The cumulative cost of all paths in the solution. Both maximum completion time and individual cost summation can be used in the CBS algorithm. Generally speaking, the cost here refers to the total individual cost, i.e., the sum of the costs of the k paths in the solution.

The CBS algorithm initially generates a root node. The constraint set of the root node is empty and the solution is comprised of the paths each agent plans through the low-level algorithm, after which the total path cost is calculated. Subsequently, the algorithm detects the path conflicts in the solution and forms a conflict set.In each step, the algorithm expands the node with the smallest total path cost in the constraint tree and verifies if the conflict set of the current node being expanded is empty. If the conflict set is empty, indicating that there are no conflicts, the CBS algorithm stops execution and returns the solution. Conversely, in the default scenario, CBS indiscriminately selects a conflict from the conflict set and splits its two constraints into new constraints for the left and right child nodes, inheriting the constraint set of the parent node to form the constraint sets of the two child nodes.Child nodes only replan the paths for the agents involved in the new constraints to satisfy the constraint set of the current child nodes, while the paths for other agents remain unchanged, thus constituting the solution of the child nodes. If the child nodes have a solution, they are added to the constraint tree for potential expansion.The CBS algorithm is complete and optimal. It ensures completeness by dealing with two cases for each conflict and guarantees optimality by utilizing best-first search both at the high-level and the low-level.

3.3 Heuristic Algorithm of CBS

The search space of CBS is a binary constraint tree, wherein the high-level search utilizes only the costs of constraint tree nodes for priority sorting. This value can be considered the g-value of the node. Many researchers have added a non-overestimating h-value to its priority to estimate the subsequent number of node splits, thereby deriving a CBS algorithm with heuristic information.To further improve CBS, heuristic methods can be added to the high-level search, which serve as approximations of the closeness between a given state and a conflict-free solution. These heuristic methods are beneficial for better determining which constraints to add, which are likely to result in conflict-free solutions more quickly. In this paper, the DG and WDG heuristic methods will be compared. The order of the heuristics from the most information to the least information is WDG, DG. The main drawback of more informative heuristics is that they require more time and resources to calculate.

Before delving into the details of these three heuristic methods, it is crucial to introduce the concept of the Multi-Value Decision Diagram (MDD). MDD serves as a graphical representation, aiming to encompass all valid paths that abide by agent constraints. It's worth noting that the layout of these paths ensures they are immune to cyclic effects. To effectively identify conflicts or dependencies between agents, we employ comparative analysis of MDDs. These graphics are generated using a modified version of the A* algorithm. The A* algorithm is a widely-used method for finding the shortest path from an agent's starting position to its target. In our enhanced version of A*, we surpass the quest for just a single shortest path; instead, we strive to discover all the shortest paths, utilizing them to construct an MDD tree that captures various shortest paths for the agent.

1) DG Heuristic

The DG heuristic, or Distance Graph heuristic, is a method that emphasizes understanding the spatial relations between agents in multi-agent pathfinding solutions. It aims to identify dependencies between agents, implying that the path chosen by each agent could potentially lead to conflicts between them. In other words, if two agents have the potential to collide on every optimal pair of paths they might adopt, they are deemed to be 'dependent'. Essentially, the DG heuristic offers valuable insights by ensuring that there will always be pivotal conflicts to settle when agents proceed along their optimal paths. This helps guide the planning process to address these conflicts early on, thereby aiding in the formulation of more efficient, safer multi-agent pathfinding strategies. In practice, an effective way to identify these dependencies is by combining the MDDs of various agents. By inspectively examining the MDDs simultaneously, it is possible to identify cases where the paths of two or more agents intersect or overlap, which could lead to conflicts or dependencies.

2) WDG Heuristic

The WDG heuristic, or Weighted Distance Graph heuristic, builds on the DG heuristic by introducing

weighted edges into the distance graph. These weights reflect the importance of the relationship between specific agents in terms of collision avoidance. By assigning different weights to edges, the WDG heuristic acknowledges that not all agent interactions bear the same level of importance. This allows for a more granular evaluation of conflicts and identification of the priority of certain paths or edges in pursuit of effective collision avoidance.

3.4 Advanced CBS Multi-Agent Pathfinding Algorithm

For the CBS algorithm, resolving conflicts swiftly facilitates quicker expansions to the target node, which effectively narrows the search space, thereby enhancing the algorithm's efficiency in pathfinding. Hence, a conflict selection strategy is proposed, in which the minimum total cost of the two corresponding child nodes that each conflict generates is used as the conflict's score. Conflicts with higher scores are chosen with priority. When conflicts have the same score, the one that brings its child nodes closer to the target node is given priority. That is, preference goes to the conflict corresponding to the child nodes with a larger g-value and fewer numbers of conflicts. To bring out the advantages of this improved conflict selection strategy and further reduce the algorithm's running time, the Depth-Q Network (DQN) algorithm-based conflict selection is adopted for the CBS algorithm.

DQN uses a parametrized neural network to approximate the Q function and find the optimal policy. The loss function for training is as follows:

$$L_i(\theta_i) = E_{s,a,r,s'}[(y_i^{DQN} - Q(s,a;\theta_i))^2] \quad (2)$$

Where, $y_i^{DQN} = r_i + \gamma_{a'}^{\max} Q(s',a';\theta_i)$, θ_i denotes the parameters corresponding to each node in the neural network. In the training process, DQN innovatively proposes the technique of experience replay, in which the state, action, and reward corresponding to each step of exploration are stored in the experience replay pool as training data, and random repeated data sampling is carried out. This not only breaks the correlation between samples but also improves the efficiency of data use.

4 Simulation Verification

Numerous simulation experiments were carried out to verify the effectiveness of the CBS algorithm. Comparative analysis was conducted between the improved CBS algorithm proposed in this paper and other heuristic CBS algorithms under the same map and the same number of intelligent agents. Implement the path planning algorithm proposed in this paper using Python language and PyTorch framework. Conduct simulation experiments to run on a computer with an Intel(R) Core(TM) i9-13900H CPU @ 2.60 GHz processor and 32 GB RAM. According to the potential landing areas of Chang'e-6 that were generated, a grid map was set up. Six intelligent agents were chosen to participate in path planning, and the departure and target point information of the six intelligent agents is presented in the following table. Under these conditions, the DG, WDG heuristic CBS algorithms and the improved CBS algorithm were used for multi-agent path planning. According to the results, the paths planned by the three algorithms were the same as shown in Figure 6, with the same total path costs but different algorithm consumption times. The consumption times for the DG-CBS, WDG-CBS, and the improved CBS algorithms were 0.42s, 3.63s, and 0.07s, respectively, as shown in Figure 7. The results show that the improved CBS algorithm proposed has a certain degree of improvement in terms of solution efficiency.

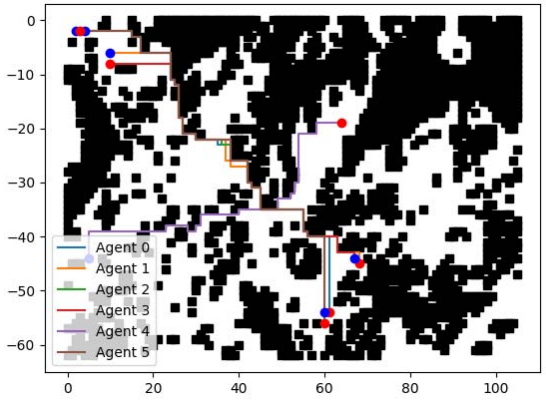

Figure6:Schematic diagram of algorithm path planning

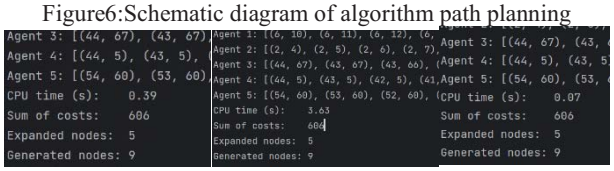

Figure7: DG-CBS, WDG-CBS, improved CBS time

5 Conclusion

This paper combines the deep Q network and CBS algorithm, proposing a CBS algorithm based on the DQN conflict choice strategy to solve the multi-agent path planning problem of lunar robot clusters. Simulation results show that under the same map, the proposed improved algorithm performs better in scenarios with different numbers of intelligent agents compared to previous improved algorithms, effectively enhancing the efficiency and success rate. The future work will explore more types of conflicts. The current CBS algorithm often considers pairwise conflicts between two agents when resolving conflicts, which are obviously insufficient in some intensive scenarios with multiple agents, where complex interactions among more than two agents may occur. In these scenarios, we should consider resolving multiple conflicts simultaneously in the future. Besides, the current algorithm adopts centralized solving, where each agent's path is planned by the central solver and all agents have the same priority. Centralized methods provide convenience in modeling, but recent research indicates that centralized control models may not be desirable in some fields. Future work will focus on distributed solutions, with multiple agents planning their paths and coordinating with each other, which show good robustness, especially in the event of interference.

References

[1] Gu Jiawei, Tang Wei, Zhang Le, et al. Collaborative Mapping and Path Planning of Indoor Multi Mobile Robots [J]. *Flight Control and Detection*, 2021,4 (04): 40-48

[2] Zhang Fuzhao, Wu Hailei, Cao Shuqing, et al. Research on Autonomous Navigation and Cross Platform Simulation System for Complex Lunar Environment [J]. *Flight Control and Detection*, 2023,6 (04): 36-44

[3] Liu Chang, Wang Yixuan, Wang Jingxin, et al. Path planning and virtual simulation of lunar robots based on improved ant colony algorithm [J]. *Flight Control and Detection*, 2021,4 (03): 23-33

[4] Liu Chang, Wang Yixuan, Wang Jingxin, et al. Lunar Rover Path Planning and Virtual Simulation Based on Improved Ant Colony Algorithm[J]. *Guidance & Detection*, 2021, 4(03):23-33.
[5] Zhao Di, Yu Liping, Hu Mengya, et al. Research on Lunar Rover Path Planning Optimization Algorithm[J]. *Mechanical Science and Technology*, 2021, 40(03): 364-370.
[6] SONG Ting， SUN Yuqi， YUAN Jianping， et al. Path planning for lunar surface robots based on improved ant colony algorithm［J］. *Transactions of Nanjing University of Aeronautics and Astronautics*，2022,39(6):672-683.
[7] Yu Xiaoqiang, Guo Jifeng, Zhao Yu, et al. Fast and Safe Path Planning for Lunar Rover[J]. *Acta Aeronautica et Astronautica Sinica*, 2021, 42(01): 277-283.
[8] Wang Kaiwen, Peng Song, Liu Shaochuang, et al. Research on Lunar Rover Path Planning Based on A* Algorithm Optimization [J]. *Spacecraft Engineering*, 2019, 28(01): 19-26.
[9] Stern R,Sturtevant N R,Felner A,et al.Multi-agent pathfinding: Definitions, variants, and benchmarks [C] //*Twelfth Annual Symposium on Combinatorial Search*,2019
[10] Liu Q Z, Wu F. Research progress of multi-agent path planning[J]. *Computer Engineering*, 2020,46(4):1-10.
[11] Che J, Tong X, Kong R. Intelligent trust path search[C]//*2020 13th International Cngress on Image and Signal Processing, BioMedical Engineering and Infomatics(CISP-BMEI).IEEE*,2020:1075-1080.
[12] Sharon G, Stern R, Goldenberg M, et al. The increasing cost tree search for optimal multi-agent pathfinding[J]. *Artificial Intelligence*, 2013, 195:470-495.
[13] Sharon G, Stern R, Felner A, et al. Conflict-based search for optimal multi-agent pathfinding[J]. *Artificial Intelligence*, 2015, 219:40-66.
[14] Wang Zihan, Tong Xiangrong. Research Progress on Multi-Agent Path Planning Based on Conflict Search [J]. *Computer Science*, 2023, 50(06): 358-368.
[15] Wang Zhuoran, Wen Jiayan, Xie Guangming, et al. Multi-agent path planning based on improved CBS algorithm[J]. *CAAI transactions on intelligent systems*, 2023, 18(6): 1336–1343.
[16] Wang Zihan. Research on Heuristic Multi-agent Path Planning Algorithm Based on Conflict Search [D]. *Yantai University*, 2023.
[17] POHL I. Heuristic search viewed as path finding in a graph[J]. *Artificial Intelligence*, 1970,1(3/4):193-204.
[18] CHAN S H, LI J, GANGE G, et al. ECBS with flex distribution for bounded-suboptimal multi-agent path finding[C]//*Proceedings of the International Symposium on Combinatorial Search*. 2021:159-161.
[19] RAHMAN M, ALAM M A, ISLAM M M, et al. An Adaptive Agent-Specific Sub-Optimal Bounding Approach for Multi-Agent Path Finding[J]. *IEEE Access*, 2022,10:22226-22237.
[20] Zeng X, Liu D, Chen Y, et al. Landing site of the Chang'e-6 lunar farside sample return mission from the Apollo basin[J]. *Nature Astronomy*, 2023, 7(10): 1188-1197.