ISSN 2616-7433 Vol. 6, Issue 4: 48-53, DOI: 10.25236/FSST.2024.060407

# Path Safety Inflation Planning Algorithm Based on Improved JPS

**Zirong Lin**

*Shanghai Starriver Bilingual School, Shanghai, 201108, China*

***Abstract:*** *This paper addresses the safety hazards present in the JPS path planning algorithm during actual operations, proposing an improved JPS planning algorithm based on path safety inflation. Firstly, the paper processes the map's obstacles with boundary inflation to reduce the probability of obstacle collision, thereby enhancing the feasibility and safety of the planned path. Secondly, considering the physical properties of drones and the feasible space of the environment in reality, this paper controls the degree of obstacle boundary inflation further by adjusting the inflation coefficient. Lastly, the paper validates the proposed algorithm through simulation experiments, with the results indicating a significant improvement in the feasibility of the drone reaching its destination amidst obstacle inflation.*

***Keywords:*** *Jump Point Search algorithm; Obstacle Inflation; Path Planning*

## 1. Introduction

In recent years, with the optimization and iteration of pathfinding algorithms for automated robots, the field of drone autonomous navigation has flourished, and autonomous path planning, as the foundation of drone autonomous navigation, has been widely developed[1]. Dutch computer scientist Edsger W. Dijkstra was the first to propose the graph-based Dijkstra algorithm[2], which finds the shortest path from the starting point to all other nodes by continuously updating the distances between nodes. However, it may face challenges of large search spaces and high computational complexity in practical applications. To address this issue, Peter Hart, Nils Nilsson, and Bertram Raphael proposed the heuristic pathfinding A* [3]algorithm based on the Dijkstra algorithm and greedy best-first search algorithm. This algorithm uses actual costs and heuristic evaluations to find the optimal path, and while guaranteeing the optimal solution, it reduces the search space and improves search efficiency through heuristic methods. However, the A* algorithm faces the issue of node redundancy in practical applications with large-scale maps, which decreases the efficiency of the algorithm. Based on this, Daniel Harabor and Alban Grastien introduced the Jump Point Search (JPS) algorithm[4], which optimizes path searching by eliminating unnecessary intermediate nodes and using jump points, significantly increasing search speed.

However, although the JPS algorithm performs excellently in virtual simulations, it overlooks key factors in practical applications, such as the size and speed of drones. This makes the algorithm capable of achieving faster path planning speeds and higher accuracy, but the paths planned carry a higher risk. To solve this problem, literature[5] combined the jump point concept of JPS with the A* algorithm, optimizing the path search of the A* algorithm and increasing the path planning speed while retaining the lower memory consumption of JPS. Meanwhile, by assessing the safety of jump points and replanning paths, the average search speed, average path length, and average collision risk were all improved[6].

Addressing the issues of safety and long search times of the JPS algorithm in global path planning for mobile robots, literature[7]proposed a bidirectional jump point search algorithm, effectively reducing search time and enhancing safety. Moreover, literature [8]introduced a path planning method that balances smoothness and search efficiency and optimized the path to reduce the trajectory length and the number of turns. Based on the jump point search algorithm, literature[9]applied its pruning rules to multiple nodes, reducing iterative calculations when searching for jump points. By eliminating intermediate turning points that only change direction through a "diagonal priority" approach, memory consumption was lowered, and computational real-time performance was improved to meet the requirements of rapid global path planning for mobile robots.

ISSN 2616-7433 Vol. 6, Issue 4: 48-53, DOI: 10.25236/FSST.2024.060407

Although the above methods improved the safety of the JPS algorithm by considering the distribution of environmental obstacles, they did not consider the physical characteristics of drones themselves. Based on this, this paper proposes an improved JPS planning algorithm based on path safety inflation, optimizing the obstacle inflation algorithm to further enhance the safety and usability of the path search algorithm in drone applications. Firstly, the paper processes the map's obstacles with boundary inflation to reduce the probability of obstacle collision, thereby enhancing the feasibility and safety of the planned path. Secondly, considering the physical properties of drones and the feasible space of the environment in reality, this paper controls the degree of obstacle boundary inflation further by adjusting the inflation coefficient. After employing the method proposed in this paper, the safety and reliability of drone flights have been greatly ensured, positively impacting the improvements to the JPS algorithm.

## 2. Jump Point Search Algorithm (JPS)

The Jump Point Search (JPS) algorithm is a heuristic search algorithm designed for efficient pathfinding, particularly suited for grid map-based path planning challenges. It optimizes the A* algorithm by incorporating preprocessing and jumping techniques, making the search process more efficient. JPS is widely used in practical applications such as game development, robotic path planning, and map navigation. The core idea of JPS is to reduce the search space through preprocessing and jumping techniques, thereby improving search efficiency. In the traditional A* algorithm, each node needs to be exhaustively expanded, leading to a large number of unnecessary node expansions. In contrast, JPS algorithm makes jumps on the map, skipping some intermediate nodes and reducing search complexity.

Specifically, during the expansion of each node, JPS performs jump operations, continuing in the current direction until encountering a blockage or a jump point. Jump points are nodes where a forced jump in the current direction occurs; they are key points that allow skipping unnecessary node expansions. For example, when encountering a straight-line obstacle in horizontal, vertical, or diagonal directions, it's possible to jump directly to the other side of the obstacle.

To find jump points, JPS uses a heuristic approach, utilizing the map's topology to determine which nodes are jump points, thus reducing unnecessary node expansions. This heuristic method allows JPS to significantly reduce search time while ensuring the search for the optimal path. The concepts of forced neighbors and jump points are important components of the JPS algorithm, which are introduced as follows:

**-Forced Neighbor**

A forced neighbor is defined during the pathfinding process as a neighbor node of a certain node X, which is on the path from a parent node P, and the cost to reach these neighbor nodes via node X is less than the cost directly from parent node P to these neighbors. This makes these neighbor nodes forced neighbors.

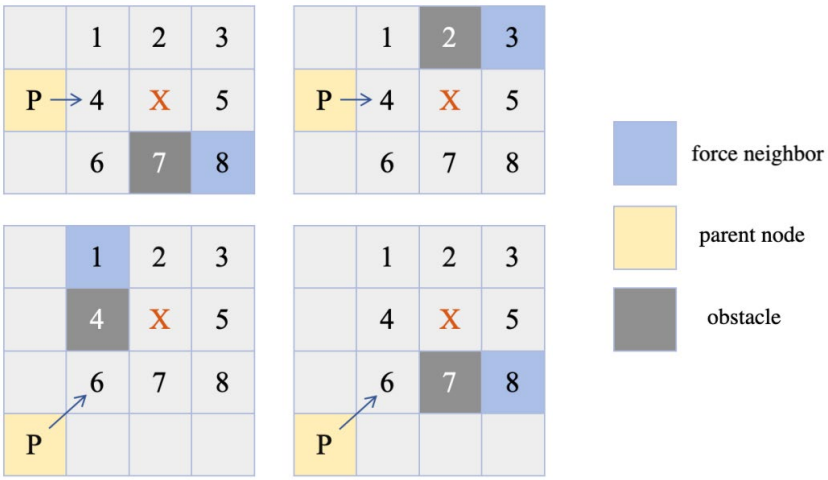


*Figure 1 Force Neighbor example diagram*

As shown in Figure 1, the blue grid represents the forced neighbor n, the yellow grid represents the

ISSN 2616-7433 Vol. 6, Issue 4: 48-53, DOI: 10.25236/FSST.2024.060407

parent node P, and the black grid represents an obstacle. Around the child node X, there are eight neighbor grids. If, starting from parent node P, the cost to reach one of the eight neighbor grids through X is less than the cost of any path not passing through X to that specific node, that grid node is termed as the forced neighbor n.

**-Jump Points**

Jump points can be categorized into four scenarios, defined as follows:

1) Start and End Points: The path's start and end points are defined as jump points. This means that in the initial stage of path planning, the start and end points are key nodes in the search path and need to be treated specially. The setting of the start and end points directly affects the quality and efficiency of the final path found, as illustrated in scenarios 1 and 4 of Figure 2.

2) Presence of Forced Neighbors: A node is defined as a jump point when it has at least one forced neighbor around it. The presence of forced neighbors indicates that the node is a necessary passage on the path, which can be used to optimize the choice of path. In this case, the existence of jump points allows the algorithm to find the optimal path more quickly, reducing unnecessary searches, as shown in scenario 2 of Figure 2.

3) Diagonal Jump Points: When a node is being searched in a diagonal direction, if there is a jump point in the horizontal or vertical direction of a diagonal node, that diagonal node is also defined as a jump point. In this case, the presence of jump points helps the algorithm skip some unnecessary intermediate nodes diagonally and move directly towards the target node, thereby accelerating the search process, as illustrated in scenario 3 of Figure 2.

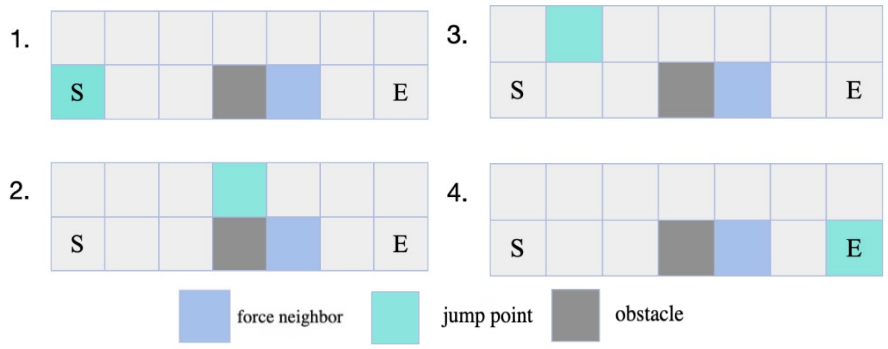


*Figure 2. Jump point example diagram*

**-JPS Algorithm Process**

The process of the Jump Point Search (JPS) algorithm can be defined in five steps, incorporating the concepts of an Open list and a Close list. The Open list consists of all nodes that are starting to be searched, while the Close list comprises all nodes that have finished being searched.

Step One: Select the node with the lowest cost in the Open list as the starting point S.

Step Two: Start the search from the starting point in horizontal, vertical, and diagonal directions. If a jump point or boundary point is encountered, end the search in that direction and add the jump point from that direction to the Open list.

Step Three: If no jump point is found, move one node diagonally forward and repeat Step Two.

Step Four: Once the search in all directions is concluded, the search for the current jump point is complete. Remove that jump point from the Open list and add it to the Close list.

Step Five: The algorithm ends when the Open list is empty or the end point is reached.

For illustration, consider Figure 3: The starting point S is added to the Open list, and the search begins as shown in part a. Jump points found during the search are added to the Open list, as illustrated in part b. In part c, S is removed from the Open list and added to the Close list because its search is completed, and the second jump point in the Open list starts being searched. Repeat Steps Two to Four until the end point E is reached and moved from the Open list to the Close list, as shown in part h, at which point the algorithm ends.

ISSN 2616-7433 Vol. 6, Issue 4: 48-53, DOI: 10.25236/FSST.2024.060407

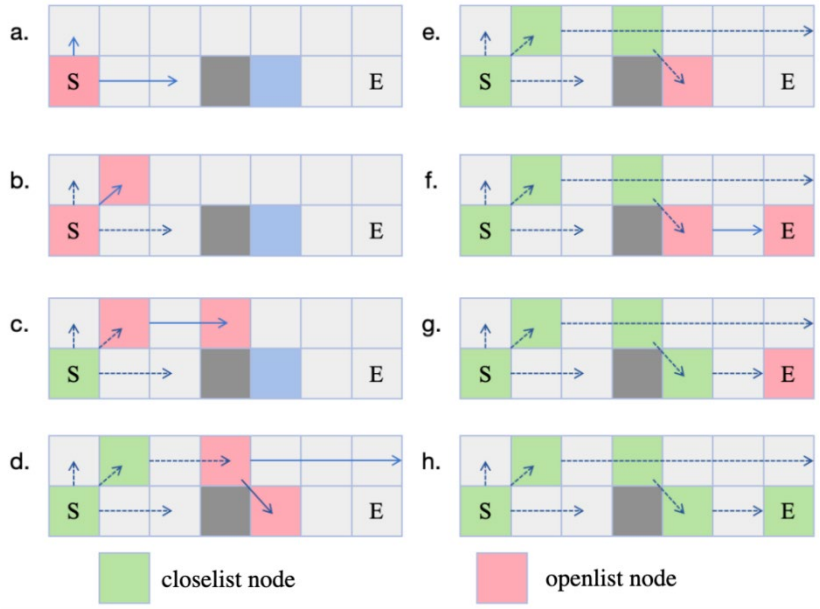


*Figure 3. Example diagram of JPS algorithm*

## 3. Obstacle expansion algorithm

In this paper, the obstacle expansion algorithm is improved based on JPS algorithm. The obstacle expansion algorithm is mainly divided into three steps. In the first step, the algorithm will determine the position and size of obstacles. As shown in Figure 4, the algorithm locates three 1*1 size obstacles at [0,2] [4,2] and [6,8] respectively. At the same time, the algorithm locates an irregularly shaped obstacle located [2,7]. In the second step, the algorithm compares the boundary of the obstacle to the boundary of the map to ensure that the swelling obstacle will not exceed the boundary. As shown in Figure 4, the expansion of the obstacle located at [0,2] will exceed the boundary at the top, and the algorithm will automatically eliminate this part of the expansion obstacle. Similarly, the obstacles located at [6,8] will expand beyond the boundary on the right and below, so the algorithm only expands the obstacles on the left and above. In the third step, the algorithm automatically calculates the impact between obstacles and adjusts the coefficient of expansion to reserve a channel for the drone to pass through. As shown in Figure 5, the obstacles at [2,5] in the figure are affected by the three obstacles at [0,2] [4,2] [4,8]. The expansion of the obstacles with an expansion coefficient of 1 will make it impossible for the UAV to pass. The algorithm automatically sets the expansion coefficient to 0.5 to reserve a channel with a width of 1 in the expanded obstacle for the UAV to pass through.

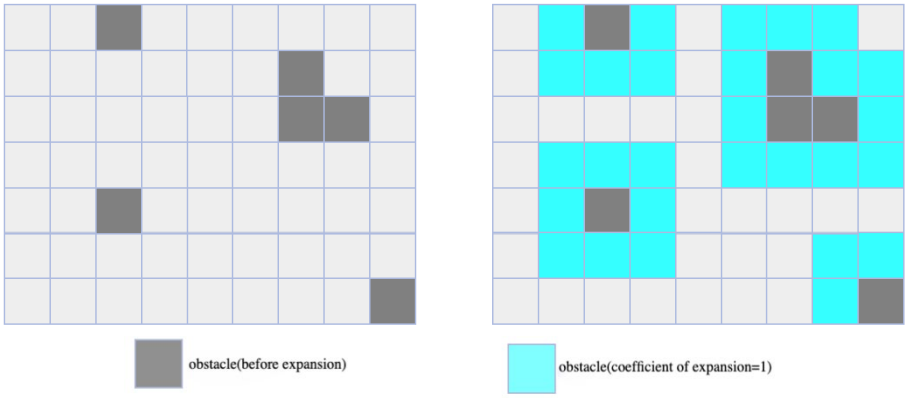


*Figure 4: The case when the expansion coefficient is 1*

ISSN 2616-7433 Vol. 6, Issue 4: 48-53, DOI: 10.25236/FSST.2024.060407

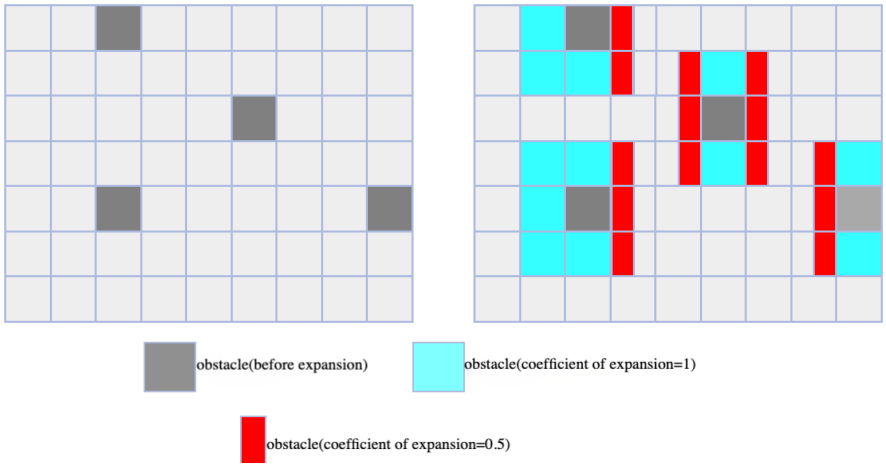


*Figure 5: The case when the expansion coefficient is 0.5*

## 4. Simulation Experiment

To verify the effectiveness of the obstacle inflation algorithm, we conducted a simulation experiment in PyCharm, with the results shown in Figures 6(a) and 6(b) as described below. As indicated by the red box in Figure 6(a), it can be objectively observed that before applying the obstacle inflation algorithm, it is very dangerous for the drone to cross grid obstacles diagonally, posing a high risk of collision with obstacles. However, the experimental results after using the obstacle inflation algorithm, as shown in the red box in Figure 6(b), effectively avoided collisions between the drone and real obstacles, significantly improving flight safety.

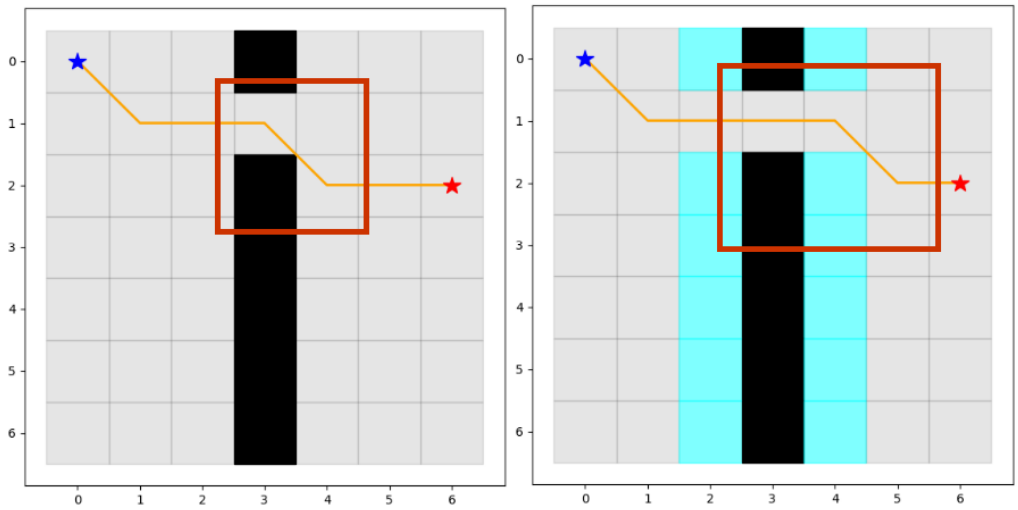

*Figure 6: (a) Experimental results without using the obstacle inflation algorithm (b) Experimental results with the obstacle inflation algorithm*

## 5. Conclusion

Addressing the potential safety risks associated with the JPS (Jump Point Search) path planning algorithm in practical operations, this paper presents an improved JPS planning algorithm based on a strategy of path safety inflation. Firstly, by inflating the boundaries of obstacles in the map, this paper successfully reduces the probability of obstacle collision, thereby enhancing the feasibility and safety of the planned path. Additionally, considering the physical properties of drones and the feasible space of the environment, the paper further optimizes the algorithm by adjusting the inflation coefficient to control the extent of obstacle boundary inflation, making the planned path closer to the actual operational requirements. Finally, the paper verifies the proposed algorithm through simulation experiments. The experimental results clearly demonstrate that the feasibility of the drone's path to the

ISSN 2616-7433 Vol. 6, Issue 4: 48-53, DOI: 10.25236/FSST.2024.060407

target endpoint is significantly improved under obstacle inflation. This conclusion underscores the potential value of the improved JPS algorithm in practical applications, providing an effective solution for the safety and reliability of drone path planning. Despite the significant achievements of the proposed improvement in the experiments, there remain some challenges and room for expansion. Future research could further explore the applicability of the algorithm in different environments and deeper strategies for path planning optimization to meet the ever-changing and complex real-world requirements. These efforts will help to better address the challenges of path planning in reality, promoting safer and more efficient applications of drone technology across various fields.

## References

*[1] Zhou Huike. Research on Low-Altitude UAV Path Planning Algorithms [D]. Xi'an University of Posts and Telecommunications, 2023.*

*[2] E. W. Dijkstra, "A note on two problems in connexion with graphs," Numerische Mathematik, vol. 1, no. 1, pp. 269–271, Dec. 1959.*

*[3] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100–107, Jul. 1968.*

*[4] D. D. Harabor et al., "Online graph pruning for pathfinding on grid maps," in Proc. 25th AAAI Conf. Artif. Intell., 2011.*

*[5] Yin Xiangzhao. Navigation Control of Mobile Robots Based on Improved A* Algorithm with Jump Point Search [D]. Nanchang University, 2021. DOI:10.27232/d.cnki.gnchu.2020.002800.*

*[6] Huang Zhibang, Hu Likun, Zhang Yu, et al. Research on Safe Path Based on Improved Jump Point Search Strategy [J]. Computer Engineering and Applications, 2021, 57(01): 56-61.*

*[7] Ma Xiaolu, Mei Hong. Research on Global Path Planning of Mobile Robots with Bidirectional Jump Point Search Algorithm [J]. Mechanical Science and Technology, 2020, 39(10): 1624-1631. DOI:10.13433/j.cnki.1003-8728.20190342.*

*[8] Huang Jianmeng, Wu Yuxiong, Lin Xiezhao. Smooth JPS Path Planning and Trajectory Optimization Method for Mobile Robots [J]. Transactions of the Chinese Society for Agricultural Machinery, 2021, 52(02): 21-29+121.*

*[9] Song Xiaoru, Ren Yiyue. Improved Jump Point Search Algorithm for Rapid Global Path Planning of Mobile Robots [J]. Science Technology and Engineering, 2020, 20(29): 11992-11999.*