Research and Optimization of D-Start Lite Algorithm in Track Planning

KAILI XIE, JIE QIANG, AND HAITAO YANG
College of Aerospace Information, Space Engineering University, Beijing 101416, China
Corresponding author: Haitao Yang (yang htt@126.com)

ABSTRACT With the widespread application of the aircraft, how to achieve the safety and efficiency of the aircraft during flight has attracted widespread attention. The track planned by the traditional D-Start Lite algorithm is very close to the obstacle. Due to the large size of the aircraft itself, it is very likely that the aircraft will collide with the obstacle when operating in a real environment. However, the existing D-Start Lite algorithm is only a single solution to the safety problem of trajectory planning or to improve the efficiency of the algorithm’s trajectory search. Therefore, this article optimizes the existing D-Start Lite algorithm. First, this article adopts the strategy of selecting priority level by child nodes. The child nodes in the vertical direction are selected as the first-level nodes. The state of the first-level node determines whether the second-level node on the diagonal needs to be expanded, so that a safe distance is generated between the planned path and the obstacle, thus improving the safety of the path. However, a certain space is reserved between the track and the obstacle, which results in the increase of the number of nodes in the path and the low efficiency of search. This article improves and optimizes the valuation function and heuristic function in the algorithm. Therefore, the number of expansion nodes and the number of inflection points are reduced, and the search efficiency of the algorithm is improved. Therefore, the improved algorithm improves the efficiency of trajectory search while ensuring the safety of trajectory planning. And the effectiveness and feasibility of the algorithm are proved by the MATLAB platform.

INDEX TERMS D-Start lite algorithm, heuristic function, selection of child nodes, trajectory planning, valuation function.

I. INTRODUCTION
In recent years, many achievements have been made in track planning [1]. Currently, the trajectory planning is mainly divided into two parts: environment modeling and algorithm optimization search. Canny proved that trajectory optimization search is an NP problem [2]. Among them, the intelligent track search algorithm is divided into two categories: One is a deterministic search algorithm, the other is a random search algorithm. As the core of the deterministic algorithm, the A-Star (A*) algorithm [3] of the graph search algorithm solves the problem of trajectory planning in a static environment. And Introduce Dynamic A*(D*) algorithm to solve the problem of low efficiency in dynamic environment [4]. In order to solve the problem that the D* algorithm is more complicated and difficult to understand, in 2004, Sves Koening and others introduced environmental incremental information and developed the Lifelong Planning A* (LPA*) algorithm on this basis [5], which greatly improved search efficiency by multiplexing environmental information. However, when people use the LPA* algorithm, they need to re-plan the route from the starting point to the target point, so in 2005, Sves Koenig et al. proposed the D-Start Lite(D*Lite) algorithm [6], [7]. The D*Lite algorithm only needs to consider the path information from the current node to the target point during path re-planning, so the search efficiency is higher. Reference [7], a path planning algorithm based on fast D* Lite is proposed. The algorithm can completely or partially avoid re-planning and calculation of some nodes, which reduce search time and improve search efficiency. In reference [8], in order to solve the collision problem of the path planned by D* Lite, a method combining lazy line of sight algorithm and distance transformation is proposed, which makes the paths under different environments very smooth and safe. Reference [9] uses an improved D*Lite search algorithm to design a three-dimensional fast track planning method. This method uses an improved cost evaluation function to combine the constraints of trajectory
planning with an improved search algorithm, which can better solve the problem of trajectory planning when the target moves in an uncertain environment. Reference [10] in this article, it introduces a new multiobjective incremental algorithm, multiobjective D* Lite (MOD*Lite) built upon a well-known path planning algorithm, D* Lite. The improved algorithm solves the problem of obtaining better execution time without sacrificing the optimality of much path length. Reference [11] presents an interpolation-based planning and re-planning algorithm for generating smooth paths through non-uniform cost grids. It is particularly well suited to planning smooth, least-cost trajectories for mobile robots.

From the above documents, it can be found that the improvement of the original D* Lite algorithm is to use fast re-planning, reduce the search nodes of the algorithm to improve search efficiency, or combine with other algorithms to achieve path security. With the complexity of the application environment and the development of intelligent technology, how to improve the search efficiency of the algorithm in the process of trajectory planning while ensuring the safety of the planned path is the main improvement direction of the D* lite algorithm.

In this article, through the analysis of D* Lite algorithm, it is found that when operating in a real environment, due to the relatively large size of the aircraft itself, there will be a collision situation with obstacles. Therefore, this article proposes to improve the selection of child nodes, and adopt the node priority strategy to improve the D* lite algorithm, which makes the planned path the optimal path while being safe and feasible. At the same time, the evaluation function in the D* lite algorithm is optimized and optimized to reduce the search time of the algorithm and improve the search efficiency of the algorithm. Under the two-dimensional map environment, the improved algorithm is simulated and tested by MATLAB platform to verify its effectiveness and feasibility.

II. RELATED WORK
A. ESTABLISH THE ENVIRONMENT MODEL
In the space environment modeling of aircraft trajectory planning, the amount of calculation and engineering applicability need to be considered. And it is required to use simple rule diagrams for modeling as much as possible. Therefore, this article chooses the grid method to model among the grid network, the configuration space method and the topology map method. The grid method is currently the most widely studied method of planning space, which divides the space for low-flying aircraft into multiple simple areas. These simple areas are called grids. Use grid cost based description to describe obstacles or threats in the planned space. This description method is based on the description of the entire planning space, using grids to divide the environment, using a two-dimensional grid to represent the environment, and increasing the value of a specific grid to describe threats or obstacles. Usually, the projection of the threat zone on the aircraft flight path in the vertical plane is composed of one or more polygon rings. Simulate the threat area of aircraft flight, and approximate it as a combination of multiple squares. Set the threat area as follows:
- If the threat area is not full of a grid, it is counted as a grid;
- If there is a hollow part in the threat area, set this part as the overall threat area;
- If the physical distance between adjacent threat areas is small, the middle area of the two threat areas is also regarded as a threat area, otherwise, it is regarded as two different threat areas. As shown in Figure 1, the generation value of each gray grid is set to 1, which indicates that there is a threat; The value of non-gray area is set to 0, which means that there is no threat, and it is a feasible area for the aircraft. The task of planning is to avoid the gray threat area and search for a path from the starting grid to the target grid.

As shown in Figure 2, the raster map used in this article. Set the threat zone as an obstacle and indicate it with a red circle, and the substitute value of the obstacle is set to −1. The target generation value is set to 0, represented by a cyan circle. The moving body is represented by a purple circle, its generation value is set to 1. And the free space cost value is 2 indicating that there is no threat, which is a feasible area for aircraft.

B. SELECTION OF D-START LITE ALGORITHM
Compared with other heuristic search algorithms, D* lite algorithm can deal with the situation of unknown environment very well. When a new unknown obstacle appears in the environment, you can quickly update the information of the nodes...
around the obstacle, put the nodes that are not continuous due to the appearance of new obstacles into the open list again, and then quickly re-planning.

**TABLE 1. Comparison of A* and LPA* algorithms.**

| Algorithm     | Application of Environment | The Method of Search | Valuation function |
|---------------|----------------------------|----------------------|--------------------|
| A* algorithm  | High search efficiency in static environment, and not suitable for dynamic environment | Heuristic search | \( F(s) = g(s) + h(s) \) |
| LPA* algorithm | Both static and dynamic environments are applicable | Incremental search for rapid re-planning in a dynamic environment | Introduce the rhs\((s)\) variable as the minimum cost estimate |

It can be seen from Table 1 that the A* algorithm has a high path search efficiency in a static environment, but the LPA* algorithm has a significantly higher search efficiency than the A* algorithm in a dynamic environment. The cost function of the A* algorithm is \( f = g + h \), and the priority of the extended node is determined by the size of the f value. LPA* algorithm is an incremental version of A* algorithm. LPA* algorithm maintain two estimates \( g(s) \) and \( rhs(s) \) of the starting distance \( g^*(s) \) of each vertex \( s \), by determine the node status by comparing the values of \( g(s) \) and \( rhs(s) \). The LPA* algorithm uses a two-dimensional key, and the order of expanding nodes is determined based on the size of the key value.

D*Lite algorithm is deepened on the basis of LPA* algorithm idea. The version presented in LPA* algorithm searches from the starting vertex to the target vertex, and the starting point is fixed. However, the search direction of the D*Lite algorithm is opposite to the LPA* algorithm. It searches from the target vertex to the starting vertex, and the starting node can be changed. In addition, the definitions of variables such as \( h, g, \) and \( rhs \) are also the exact opposite of LPA*. As shown in Table 2, D*Lite algorithm has advantages over LPA* algorithm.

Based on the comparison between the above algorithms, the D*Lite algorithm will become the research direction of many scholars, and optimize and improve the algorithm from the efficiency and accuracy of path search.

**C. PRINCIPLE OF D-START LITE ALGORITHM**

The D*lite algorithm is extended based on the idea of LPA* algorithm, they are all based on A* algorithm. D*lite algorithm and LPA* algorithm add rhs variable on the basis of A* algorithm, and the core of the D*lite algorithm lies in the incremental search path based on the unknown area as a free space. The D*lite algorithm selects the minimum rhs value as the best expansion direction each time, and iterative searches and calculates the cost estimates of the 8 surrounding lattices until it finds the target position [12]. In the D* Lite algorithm, \( g(s) \) and \( rhs(s) \) are the minimum cost estimates of the starting point of the node \( s \) distance. Since the search direction of the D*lite algorithm is opposite to the A* algorithm and the LPA* algorithm, the \( g(s) \) value will be recalculated when expanding 8 adjacent grids around the grid. Update the \( g(s) \) value and choose the \( g(s) \) with the smallest generation value. Rhs\((s)\) is calculated from the \( g \) value of its forward point, and its calculation formula is as follows:

\[
rhs(s) = \begin{cases} 
0, & s = s_{goal} \\
\min_{s' \in \text{Pred}(s)} (g(s') + c(s', s)), & \text{others}
\end{cases}
\]

where \( c(s', s) \) represents the edge cost of nodes \( s' \) to \( s \), which is generally recorded as 1. \( S \in \text{pred}(s) \) denotes the set of predecessors of vertex \( s \in S \). When \( g(s) = rhs(s) \), it means that nodes are consistent, otherwise it is non-uniform. Among them, \( g(s) > rhs(s) \) is called local over-consistency, and \( g(s) < rhs(s) \) is called local under-conformity. When evaluating the estimated value of grid points, D* Lite also introduces key\((s)\) value for comparison, where key\((s)\) contains two values key\((s) = [\text{key1}(s); \text{key2}(s)]\), which satisfy the following formulas:

\[
\text{key1}(s) = \min(g(s), rhs(s)) + h(s) \quad (2)
\]
\[
\text{key2}(s) = \min(g(s), rhs(s)) \quad (3)
\]

\( k1 \) and \( k2 \) as the priority queue arrangement parameters. The size of the key\((s)\) value is used as the expansion priority, which determines the expansion order of the nodes in the queue. First compare the size of the K1 value, select the smallest k1 value grid as the next grid, and when the k1 values are equal, consider the size of the k2 value. The \( h(s) \) heuristic function is the same as the heuristic function in the LPA* algorithm, which represents the estimated value between the current node and the starting point, causing the object to always move toward the target point. The value of \( h(s) \) is very important, it determines the speed and accuracy of the algorithm search. The search heuristic function of node \( s \) includes two parts: the cost \( c(s, s') \) from node \( s \) to node \( s' \) and the value of the heuristic function from node \( s' \) to the target

**TABLE 2. Comparison of LPA* and D* lite algorithms.**

| Algorithm     | Application of Environment | The Method of Search | Re-planning operations |
|---------------|----------------------------|----------------------|-----------------------|
| LPA* algorithm | Both static and dynamic environments are applicable | Forward search from the starting node to the target node | Update the path information of the node from the starting point |
| D*Lite algorithm | Environment is more flexible, and the starting point can change with time | Reverse search from target node to starting node | Update the path information of the current node |
point \( s_{\text{goal}} \). The calculation formula of \( h(s) \) is:

\[
h(s, s_{\text{start}}) = \begin{cases} 
0, & s = s_{\text{goal}} \\
c'(s', s) + h(s', s_{\text{goal}}), & \text{others} 
\end{cases}
\]

(4)

The description of D*Lite path planning algorithm is shown in Table 3 [13]

### III. OPTIMIZATION OF D-START LITE ALGORITHM

The D*Lite algorithm is based on the LPA* algorithm, which is improved to make it possible to re-plan quickly when facing obstacles [7], thus improving the planning efficiency. There have been many studies on the application of D* lite algorithm in trajectory planning. How to improve the efficiency of the algorithm and the security of algorithm application is the main direction of algorithm improvement. However, in practical applications, planning algorithms need to consider the safety of planned trajectories while improving the efficiency of the algorithm.

The traditional D*lite algorithm effectively expands the 8 nodes around the node with the parent node as the center. Put the expanded node into the priority queue (including obstacle nodes), and select the node with higher priority in the priority queue as the current node. Traverse the surrounding nodes again until reaching the target node. The final planned path is easy to be close to obstacles. However, the D*lite algorithm itself still has deficiencies. The path planned by D*lite is close to obstacles, when operating in a real environment, due to the large size of the aircraft itself, if the aircraft is very close to the obstacle during the re-planning operation, then the possibility of collision between the aircraft and the obstacle is very high, which making D*lite plan Path is not safe. In order to solve this problem, this article improves the expansion mode of sub nodes, so that the planned path is safe and feasible. In addition, in order to improve the efficiency of D*lite algorithm in trajectory planning, the weighted valuation function in the algorithm is improved.

#### A. VALUATION FUNCTION

The D*Lite algorithm uses the priority queue \( U \) to store the non-uniform nodes to be expanded, and the expansion order of the node \( S \) is determined by the valuation function. In order to reduce search time and improve planning efficiency, heuristic function weighting is a commonly used method to improve the valuation function. In the valuation function of D*Lite, \( \min(g, rhs) \) is the width-first component, while the depth-first component is represented by \( h \). You can use \( w \) to adjust the relative weights of the two components \( \min(g, rhs) + w \cdot h(1 \leq w \leq \infty) \), thus, on the basis of maintaining the optimization of the algorithm as much as possible, the search efficiency of the algorithm is improved.

Where the evaluation function \( \text{key}(s) = [\text{key}(S1); \text{key}(S2)] \), key1 \( (s) \) is equivalent to the evaluation function \( f(s) \) in the A* algorithm, and key2 \( (s) \) is equivalent to the heuristic function \( g(s) \) in the A* algorithm. Through reading the literature, it is found that the number of nodes expanded by the A* algorithm to select key2 \( (s) \approx h(s) \) is significantly less than key2 \( (s) \approx h(s) \). Therefore, this article uses an improved weighted valuation function [13]:

\[
\text{key}(s) = \begin{cases} 
g(s) + h(s), & g(s) < rhs(s) \\
[rhs(s) + w \cdot h(s), h(s)], & \text{otherwise} 
\end{cases}
\]

(5)
At the same time, this article replaces the Euclidean distance in the original algorithm by Chebyshev distance. The aircraft can move along the direction of the corner line, solving the problem of relatively high turning frequency within a small range of the planned path. In this way, the distance between the current node and the target point can be determined by the following relationship:

$$D_{\text{chess}}(\text{Goal}, n) = h_{\text{diagonal}} + h_{\text{straight}}$$  \hspace{1cm} (6)

Among them, $h_{\text{diagonal}}$ and $h_{\text{straight}}$ are:

$$h_{\text{diagonal}} = \min((x_{\text{goal}} - x_1), (y_{\text{goal}} - y_1))$$  \hspace{1cm} (7)

$$h_{\text{straight}} = \text{abs}(\text{abs}(y_{\text{goal}} - y_1) - \text{abs}(x_{\text{goal}} - x_1))$$  \hspace{1cm} (8)

where $(x_{\text{goal}}, y_{\text{goal}})$ is the position coordinate of the target node, and $(x_1, y_1)$ is the position coordinate of the current extended node. In this article, the cost in the vertical and horizontal directions is 1, the cost in the diagonal direction is 1.4, and the cost in the two directions is substituted into:

$$h(s) = \min((x_{\text{goal}} - x_1), (y_{\text{goal}} - y_1)) + 1.3\text{abs}(\text{abs}(y_{\text{goal}} - y_1) - \text{abs}(x_{\text{goal}} - x_1))$$  \hspace{1cm} (9)

### B. SELECTION OF CHILD NODES

The D* lite algorithm is similar to the A* algorithm, and it is based on the current node. It uses an iterative search method to expand to the 8 neighboring nodes and push the resulting non-uniform nodes into the open queue. But the path planned in this way will be close to obstacles. Due to the large size of the aircraft, it is very likely to collide with the obstacle in the application of the actual environment, and there are hidden safety risks. In order to solve this problem, this article improves the selection of child nodes of the D* lite algorithm.

As shown in the figure, the priority selection strategy is adopted for node selection [14]. The child node in the vertical and horizontal directions is selected as the first-level priority node, and the child node in the opposite diagonal direction is the second-level priority node. Select the second priority node by the status of the first priority node: If node of 5 is an obstacle, child nodes 3 and 8 in the diagonal direction cannot be selected, and paths 0-3 and 0-8 are excluded. The same analogy, if there are obstacles in nodes(2, 4, 7), the corresponding diagonal sub-nodes cannot be selected.

### C. THE PROCESS OF IMPROVING THE D-START LITE ALGORITHM

In the actual living environment, the aircraft itself has a large size and is prone to appear close to obstacles. This article proposes the strategy of adopting node priority in generating nodes. Under the function of node priority, the nodes with obstacles around are not put into the scope of node search calculation, and a certain safety distance is reserved to meet the application in the actual environment. Although the phenomenon of being close to obstacles is avoided, the number of path nodes and turning points have increased. In this article, Chebyshev distance is used to replace the Euclidean distance in the original algorithm, and the diagonal function is used to improve the heuristic function, so that the turning point of the planned path is reduced. At the same time, heuristic function weighting is used to optimize the valuation function to reduce the number of expansion of path nodes. The search time is reduced, and the search efficiency is improved.

Figure 4 is a flowchart of the improved D* Lite path planning algorithm:

The specific steps are:

1. **Initialization**: The g(s) and rhs(s) values of all grids are infinite. First put the local inconsistent node sgoal into the priority queue.

2. **Calculate the shortest path**: Starting from the target point, adopt the priority grading strategy generated by the node. Put the currently needed expansion node into the priority queue, and calculate the size of g(s), rhs(s) and k.
value according to the above improved valuation function and heuristic function. Select the node with the smallest $k$ value as the next expansion node. Loop this step until the target point is equal to the starting point, then the search is terminated.

(3) Generate the shortest path: Moving from the current node to the grid with the smallest estimated value, generating a path to the target node.

(4) The environmental information changes: When it is found that the next node of the path is an obstacle, the environmental information will change accordingly, such as the value of edge cost. At this time, update the $k$ value of the node and the state of the surrounding critical point affected by the node, then go back to step 3 and find a path according to step 3.

IV. EXPERIMENTS

In order to verify the feasibility of the improved D*lite algorithm, this article conducts a simulation experiment on the improved D*lite algorithm in the MATLAB platform. The environment used in the simulation experiment is that the PC memory is 8G, the operating system is Windows 10 64-bit operating system, and MATLAB R2016b is used as the simulation software. The experimental results are compared and analyzed as follows: (1) Under the simple map, set obstacles to be randomly distributed in a grid of $30 \times 30$ units. (2) Set the grid map of $100 \times 100$ units, which the distribution of obstacles is more complex. The feasibility of the improved algorithm in this article is verified through comparative experiments.

Set obstacles randomly distributed in a $30 \times 30$ unit raster. As shown in Figure 5(a), it is the path simulation diagram of the original algorithm, where the red circle represents an obstacle, and when traversing to the threat area, the red circle becomes solid; the blue circle represents the trajectory of the aircraft. The yellow position is the starting point, and the cyan grid is the position of the target point. It can be seen from the figure that the path planned by the original algorithm has redundant nodes and trajectory segments that are close to obstacles. In the actual environment, considering the size of the aircraft itself, collisions will inevitably occur in the planned path. Therefore, the node priority strategy is adopted in the node generation link to improve the D*lite algorithm. The improved path is shown in Figure 5(b).

From the above two pictures, when the improved D*lite algorithm is close to the obstacle area, according to the node priority strategy, the node on the side with the obstacle is not expanded. It avoids the aforementioned phenomenon of clinging to obstacles and can be better applied to the actual environment.

Since a certain space is reserved between the trajectory and obstacles, the number of nodes in the path increases, making the search efficiency low. Therefore, this article uses Chebyshev distance to replace Euclidean distance in the original algorithm, and the diagonal function is used to improve the heuristic function. So that the turning point of the planned path is reduced. At the same time, heuristic function weighting is used to optimize the valuation function, which can reduce the number of expansion of path nodes and search time, and improve the search efficiency.

In the evaluation function of the improved D*Lite algorithm, use the variable $w$ to adjust the relative weights of width priority component $\min (g, rhs)$ and depth priority component $h$ in 

$$\text{key1} = \min (g, rhs) + w \cdot h (1 \leq w \leq \infty).$$

This article takes different values of $w$ for path planning experiments, and the corresponding results are shown in the following table:

| $w$ | Path Length | Expanded Nodes |
|-----|-------------|----------------|
| 1   | 20 units    | 300            |
| 2   | 30 units    | 200            |

It can be seen from the table that as $w$ increases, the path length increases but does not exceed $w$ times the shortest path. The number of expanded nodes of the path is significantly reduced, so the search speed of the improved D*lite algorithm
is faster. The search time is significantly reduced, and the search efficiency of the algorithm is improved. In order to make the effect of the planned path relatively optimal, the algorithm in this article takes \( w = 1.2 \). The simulation result is shown in Figure 6:

In order to verify the feasibility of the improved algorithm, this article conducts 8 experiments to compare the path length, the number of expanded nodes and the planning time of the planned path. As shown in Figure 7.

Table 5 lists the specific experimental data of the three algorithm trajectory planning in the simple map. According to the data in the table, compared with the above two algorithms, the improved algorithm in this article has significant advantages in planning time and expanding the number of nodes:

Perform planning experiments on three sets of algorithms in the same environment. The planning time of the improved D*lite algorithm is significantly reduced compared to the original D*lite algorithm. However, due to the change in the selection method of the child nodes, there are redundant nodes in the path of the planning site. The path length planned by the improved D*lite algorithm has only slightly increased. And the number of expansion nodes and turning points of the path have been significantly reduced, which the resulting path is safe and effective. Therefore, the improved D*lite algorithm is effective.

Compare the three performances of the original D*lite algorithm, the D*lite algorithm with improved child node selection method, and the weighted D*lite algorithm with improved child node selection method mentioned above, in three different sizes of unknown maps. Figure 8 compares the target-oriented navigation of the three algorithms in unknown terrain. In 3 areas of different sizes, randomly generated terrain of each size whose obstacle density varies from 10 to 40 percent. This graph shows the operating conditions of the three performances in different algorithms. The improved D*Lite in this article performs better than the other two algorithms with respect to all three measures. And the figure also shows the corresponding 95 percent confidence intervals to demonstrate that our conclusions are statistically significant.

At the same time, in order to further verify the scalability of the improved D*lite algorithm, a more complex environment is selected for simulation. This article uses the same experimental environment settings as the previous simulation.

### TABLE 4. The length of the shortest path, the number of extended nodes and the elapsed time in the D*lite algorithm under different weights.

| \( w \) | The length of the shortest path | The number of extended nodes | The elapsed time |
|---|---|---|---|
| 1.0 | 52.21 | 1580 | 16.013 |
| 1.1 | 52.796 | 1391 | 13.556 |
| 1.2 | 53.382 | 1231 | 11.907 |
| 1.3 | 53.968 | 1103 | 10.750 |
| 1.4 | 54.796 | 983 | 9.251 |

### TABLE 5. Comparison of algorithm experiment results.

| Simple map | The length of the shortest path | The number of the extended nodes | The elapsed time |
|---|---|---|---|
| Original D*lite algorithm | 18.782 | 153.875 | 3.145 |
| D*lite algorithm improved with child node selection method | 22.819 | 141 | 3.034 |
| D*lite algorithm with improved child node selection | 20.9155 | 128 | 2.022 |
experiments with three different algorithms in the unknown terrain with a more complex environment. The verification result is shown in Figure 9:

As shown in Figure 9, there are fewer redundant points and inflection points in the path, so the trajectory is smoother than the path planned by the original algorithm. Therefore, in the actual environment, it is possible to avoid collision between the aircraft and obstacles during navigation and steering. In order to further verify the effectiveness of the algorithm, in the same complex environment map above, select different starting points and target points, and conduct 10 different experimental simulations. The data is averaged after the experiment, and the comparison results are shown in Table 6.

From the statistical data in the table, it can be seen that the path length of the improved algorithm is increased by 2%. However, compared with the d* Lite algorithm, which only improves the selection of sub nodes, it reduces by 4.3%. At the expense of a certain path length, the security of the path is enhanced. Due to the improved heuristic function and optimization of the valuation function, compared with the original D*Lite algorithm, the improved D*Lite algorithm reduces the number of expanded nodes by 14.5% and the time consumed by the algorithm by 12.7%. Therefore, the improved algorithm used in this article not only improves the search efficiency of the algorithm, but also solves the safety of the planned flight path. And meets the feasibility and scalability of the D*Lite algorithm in practical engineering.

V. CONCLUSION

The existing D*Lite algorithm can only single solution the safety problem of trajectory planning alone or improve the search efficiency of the algorithm. Therefore, in order to solve the problem of ensuring the safety of the trajectory planning path while improving the efficiency of the algorithm. This article proposes an improved D*Lite algorithm based on the node priority grading strategy, which makes the planned trajectory away from obstacles. In the actual application environment, the possibility of collision between aircraft and obstacles is avoided. Through the improvement and optimization of the heuristic function and the evaluation function and the planning time is reduced, and the planning search efficiency is improved. This article compares the improved D*Lite algorithm with the original algorithm to verify the effectiveness of the algorithm and its adaptability to different environments. Although the D*Lite algorithm still has insufficient planning for the local environment, which is also a problem to be solved in the next step.

| Complex map          | The length of the shortest path | The number of extended nodes | The elapsed time |
|----------------------|---------------------------------|------------------------------|------------------|
| Original algorithm   | D*Lite                          | 52.796                       | 1580             | 16.013           |
| the algorithm with   |                                 |                              |                  |                  |
| improved child node  |                                 |                              |                  |                  |
| selection method     |                                 |                              |                  |                  |
| the weighted D*Lite  |                                 |                              |                  |                  |
| with improved node   |                                 |                              |                  |                  |
| selection            |                                 |                              |                  |                  |

| Original algorithm   | D*Lite                          | 56.981                       | 1432             | 15.725           |
| the algorithm with   |                                 |                              |                  |                  |
| improved child node  |                                 |                              |                  |                  |
| selection method     |                                 |                              |                  |                  |
| the weighted D*Lite  |                                 |                              |                  |                  |
| with improved node   |                                 |                              |                  |                  |
| selection            |                                 |                              |                  |                  |

| Original algorithm   | D*Lite                          | 53.953                       | 1350             | 13.978           |
| the algorithm with   |                                 |                              |                  |                  |
| improved child node  |                                 |                              |                  |                  |
| selection method     |                                 |                              |                  |                  |
| the weighted D*Lite  |                                 |                              |                  |                  |
| with improved node   |                                 |                              |                  |                  |
| selection            |                                 |                              |                  |                  |
REFERENCES

[1] L. Lei, “Application research of intelligent optimization algorithm in UAV track planning,” Ship Electron. Eng., vol. 5, no. 2, pp. 34–37, 2017.

[2] J. Canny, “A computational approach to edge detection,” IEEE Trans. Pattern Anal. Mach. Intell., vols. PAMI-8, no. 6, pp. 679–698, Nov. 1986.

[3] P. E. Hart, N. J. Nilsson, and B. Raphael, “Correction to a formal basis for the heuristic determination of minimum cost Paths,” ACM SIGART Bull., no. 37, pp. 28–29, Dec. 1972.

[4] A. Stentz, “The focussed algorithm for real-time re-planning,” in Proc. Int. Joint Conf. Artif. Intell., 2000, pp. 3310–3317.

[5] S. Koenig, M. Likhachev, and D. Furcy, “Lifelong planning A*,” Artif. Intell., vol. 155, nos. 1–2, pp. 93–146, 2004.

[6] S. Koenig and M. Likhachev, “Lite,” in Proc. AAAI, 2002, pp. 476–483.

[7] S. Koenig and M. Likhachev, “Fast re-planning for navigation unknown terrains,” IEEE Trans. Robot., vol. 21, no. 3, pp. 354–363, Oct. 2005.

[8] Y. Zhang, F. Sun, and X. Shi, “Mobile robot path planning based on fast,” Data Commun., vol. 1, no. 1, pp. 46–51, 2018.

[9] W. Zhangq., Z. Xiaoguang, and H. Qingyao, “Mobile robot path planning based on parameter optimization ant colony algorithm,” Procedia Eng., vol. 15, pp. 2738–2741, 2011.

[10] T. Oral and F. Polat, “Lite: An incremental path planning algorithm taking care of multiple objectives,” IEEE Trans. Cybern., vol. 46, no. 1, pp. 245–257, Feb. 2015.

[11] D. Ferguson and A. Stentz, “Algorithm for improved path planning and re-planning in uniform and non-uniform cost environments,” Robot. Inst., Carnegie Mellon Univ., Pittsburgh, PA, US, Tech. Rep. CMU-RI-TR-05-19, Jun. 2005.

[12] X. Chen and D. Liu, “Three-dimensional trajectory planning of unmanned aerial vehicle during target movement using,” Electron. Control, vol. 20, no. 7, pp. 1–5, 2013.

[13] Y. Lian, Y. Fan, H. Yu, and Z. Yang, “Application of improved lite algorithm in virtual soldier path planning,” Modern Electron. Technol., vol. 46, no. 6, pp. 23–27 and 33, 2018.

[14] S. Wang, L. Hu, and Y. Wang, “Path planning of indoor mobile robot based on improved algorithm,” Comput. Eng. Des., vol. 41, no. 4, pp. 1118–1124, 2020.

[15] H. Pan, “Research on autonomous path planning algorithm of mobile robot in complex environment,” Dalian Maritime Univ., Dalian, China, Tech. Rep., 2018.

KAILI XIE received the bachelor’s degree in software engineering from the Harbin University of Technology, Harbin, China. He is currently pursuing the M.S. degree in information system from Space Engineering University, Beijing, China. His research interest includes path planning.

JIE QIANG received the bachelor’s degree in electronic engineering from Shanghai Jiao Tong University, Shanghai, China, in 2014, and the master’s degree in computer science from Space Engineering University, Beijing, China, in 2017, where he is currently pursuing the Ph.D. degree in information system. His research interests include deep learning and reinforcement in network security.

HAITAO YANG received the bachelor’s and master’s degrees in communication engineering and the Ph.D. degree in information system from Space Engineering University, Beijing, China. He is currently an Associate Professor of communication and information system. His research interests include image processing and decision making. He has published his book Design of Complex Information Network Performance.