Received December 5, 2021, accepted December 21, 2021, date of publication December 30, 2021, date of current version January 21, 2022.

Digital Object Identifier 10.1109/ACCESS.2021.3139534

Quality-Oriented Hybrid Path Planning Based on A* and Q-Learning for Unmanned Aerial Vehicle

DONGCHENG LI[1], WANGPING YIN[2], W. ERIC WONG[1], MINGYONG JIAN[2], AND MATTHEW CHAU[1]

[1]Department of Computer Science, The University of Texas at Dallas, Richardson, TX 75080, USA
[2]School of Computer Science, China University of Geosciences, Wuhan, Wuhan 430074 China

Corresponding author: W. Eric Wong (ewong@utdallas.edu)

This work was supported by the USA National Science Foundation under Grant 1757828, Grant 1822137, and Grant 2050869.

ABSTRACT Unmanned aerial vehicles (UAVs) are playing an increasingly important role in people's daily lives due to their low cost of operation, low requirements for ground support, high maneuverability, high environmental adaptability, and high safety. Yet UAV path planning under various safety risks, such as crash and collision, is not an easy task, due to the complicated and dynamic nature of path environments. Therefore, developing an efficient and flexible algorithm for UAV path planning has become inevitable. Aimed at quality-oriented UAV path planning, this paper is designed to analyze UAV path planning from two aspects: global static planning and local dynamic hierarchical planning. Through a theoretical and mathematical approach, a three-dimensional UAV path planning model was established. Based on the A* algorithm, the search strategy, the step size, and the cost function were improved, and the OPEN set was simplified, thereby shortening the planning time and greatly improving the execution efficiency of the algorithm. Moreover, a dynamic exploration factor was added to the exploration mechanism of Q-learning to solve the exploration-exploitation dilemma of Q-learning to adapt to the local dynamic path adjustment for UAVs. The global-local hybrid UAV path planning algorithm was formed by combining the two. The simulation results indicate that the proposed planning model and algorithm can efficiently solve the problem of UAV path planning, improve the path quality, and can be a significant reference for solving other problems related to path planning, such as the reliability, security, and safety of UAV, when embedded into the heuristic function of the proposed algorithm.

INDEX TERMS Unmanned aerial vehicle, quality-oriented path planning, A* algorithm, reinforcement learning, hierarchical planning.

I. INTRODUCTION

Unmanned aerial vehicles (UAVs) are aircraft that can be controlled by a ground station or via onboard electronic equipment and can be fully or partially autonomous. With high maneuverability and good concealment, UAVs are increasingly vital in people's daily lives. At present, the typical uses of UAVs include surveillance, rescue, delivery, communication relay, and airborne early warning [1].

According to whether the obstacle information is known, UAV path planning can be divided into two categories: static planning, in which the locations of all obstacles and threats are known before planning and whereby a reasonable path can be planned before UAV take-off [2]; and dynamic planning, in which the UAV needs to deal with uncertain obstacles or unexpected threats by dynamically resolving the conflicts. Dynamic path planning involves more complex issues and can improve UAV flight efficiency [3].

The existing methods for path planning can be divided into numerical optimization, potential field-based method, heuristics (classical heuristics and group intelligence algorithms), sampling-based method, and deep reinforcement learning [4]–[6]. Table 1 illustrates the characteristics of all the above mentioned algorithms.

Given the status quo, UAV path planning mainly faces the following key technical challenges:

1) Smart algorithms commonly used for UAV path planning often take a long time due to their high complexity [7]. For this reason, these algorithms are time-consuming and thus difficult to implement when solving large-scale schemes of UAV path planning.

2) UAV path planning is subject to many constraints in practice. Restricted by the high complexity and time-consuming process of the model, the existing algorithms

The associate editor coordinating the review of this manuscript and approving it for publication was Zhaojun Steven Li.

TABLE 1. Path planning methods.

Type	Near superiority	Complexity	Characteristics
Numerical optimization	No	No	Low efficiency, heavily depending on the initial point
Heuristic	Yes	Yes	High reliability in dynamic environments, sensitive to uncertainty
Sampling-based	No	YES	Easy to apply to high-dimensional space
Potential field-based	No	No	Easy to implement, yet easy to fall into local optimal
Deep reinforcement learning	Yes	Yes	Suitable for unknown environments

can be used only for experimental research based on simplified models [8]. Thus, this approach cannot truly reflect the actual needs of UAV path planning.

In this paper, through systematic investigation of the problems in UAV path planning, a global-local UAV path planning algorithm based on reinforcement learning was designed according to the simulated scenarios. On combining the constraints and targets, UAV path planning was achieved, and related experimental verification was performed.

The preceding sections of the paper are arranged as follows. Section 2 introduces the design and modeling considerations of UAV path planning. Section 3 presents the proposed solution and algorithmic approach for UAV global-local path planning. The experimental design and result analysis is explained in Section 4. Finally, we conclude the paper and elaborate on potential future directions in Section 5.

II. MODELING FOR UAV PATH PLANNING

A. PROBLEM DESCRIPTION

The UAV path planning problem can be described as follows: There are N UAVs, M types of loads, and K static or dynamic obstacles, and the UAVs can perform tasks such as surveillance, rescue, and delivery. To satisfy the UAV kinematic constraints and various resource constraints, one should choose a path that passes through all target task nodes and can dynamically adjust in real time when encountering obstacles to shorten the task duration and improve the success rate. In this paper, the task nodes were abstracted as the path nodes on the UAV flight path, and the final planned result was a sequence of path nodes.

The path planning schemes are shown in Figure 1.

For UAV path planning, the scheme designed in this paper consists of two levels: the global path planning based on modified A* and the local dynamic path planning based on modified Q-learning.

B. CONSTRAINTS

1) Vertical maximum turning angle constraint

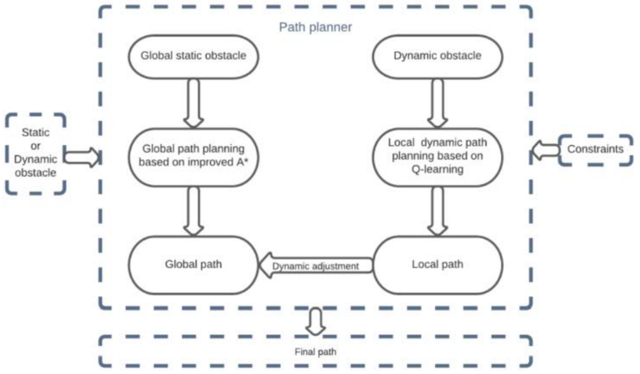

Path planner
Global static obstacle
Dynamic obstacle
Static or Dynamic obstacle
Global path planning based on improved A*
Local dynamic path planning based on Q-learning
Constraints
Global path
Dynamic adjustment
Local path
Final path

FIGURE 1. Global-local hybrid path planning scheme.

Let the present path node be $P_i(x_i, y_i, z_i)$ and the subsequent one be P_{i+1} $(x_{i+1}, y_{i+1}, z_{i+1})$. Then the maximum turning angle constraint in the vertical direction can be expressed as

$$\tan^{-1}\left(\frac{|z_{i+1}-z_i|}{\sqrt{(x_{i+1}-x_i)^2+(y_{i+1}-y_i)^2}}\right) \leq \theta_v, \quad (i = 1, 2, \ldots, n) \tag{1}$$

where θ_v denotes the maximum turning angle of the UAV in the vertical direction.

2) Horizontal maximum turning angle constraint

Let the present path node be $P_i(x_i, y_i, z_i)$ and the subsequent one be $P_{i+1}(x_{i+1}, y_{i+1}, z_{i+1})$. Then the maximum turning angle constraint in the horizontal direction can be expressed as

$$\tan^{-1}\left(\frac{(y_{i+1}-y_i)}{(x_{i+1}-x_i)}\right) \leq \theta_l, \quad (i = 1, 2, \ldots, n) \tag{2}$$

where θ_l denotes the maximum turning angle of the UAV in the horizontal direction.

3) Horizontal flight speed constraint

The horizontal flight speed constraint of the UAV can be expressed as

$$V_{lmin} \leq V_i \leq V_{lmax} \tag{3}$$

where V_{lmin} denotes the minimum horizontal flight speed (excluding the starting phase) of the UAV, V_i denotes the current horizontal flight speed of the UAV, and V_{lmax} denotes the maximum horizontal flight speed of the UAV.

4) Climbing speed constraint

The climbing speed constraint of the UAV can be expressed as

$$0 \leq V_i \leq V_{vmax} \tag{4}$$

where V_i denotes the current climbing speed of the UAV and V_{vmax} denotes the maximum climbing speed of the UAV.

5) Minimum turning radius constraint

The minimum turning radius constraint can be expressed as

$$R_i \geq R_{max}, \quad (i = 1, 2, \ldots, n) \tag{5}$$

where R_i denotes the turning radius at the i-th turn in the path planning result and R_{max} denotes the maximum turning radius of the UAV. R_{min} is calculated as

$$R_{min} = \frac{V_{min}^2}{g \times \sqrt{n_{ymax}^2 - 1}} \tag{6}$$

where V_{min}^2 denotes the minimum flight speed of the UAV and n_{ymax}^2 denotes the maximum normal phase overload of the UAV.

6) Farthest flight length constraint (maximum flight time constraint)

The farthest flight length constraint (the maximum flight time constraint) can be expressed as

$$\Delta_V * T_i \leq L_{max}\ (T_i < T_{max}) \tag{7}$$

where Δ_V denotes the average speed of the UAV during flight, T_i denotes the flight time, T_{max} denotes the maximum flight time, and L_{max} denotes the maximum allowable flight length.

7) Flight height constraint

The flight height constraint can be expressed as

$$H_{min} \leq H_i \leq H_{max}, \quad (i = 1, \ldots, n) \tag{8}$$

where H_{min} denotes the minimum flight height (excluding the take-off and landing phases) of the UAV, H_i denotes the current UAV flight height, and H_{max} denotes the maximum flight height of the UAV.

8) The total energy consumption of any UAV executing tasks shall not exceed its total energy and can be expressed as

$$E_i \leq \mathrm{E}, \quad (i = 1, 2, \ldots, n) \tag{9}$$

where E_i denotes the energy consumption of the i-th UAV and E denotes the total energy.

C. COST FUNCTION

1) ENERGY CONSUMPTION COST

The energy consumption cost of the UAV can be expressed as

$$Cost_e = \sum_{i=1}^{N-1} k * l_i\ (i \geq 2) \tag{10}$$

where $Cost_e$ denotes the energy consumption cost, k denotes the ratio of energy consumption to flight length, N denotes the number of nodes in the resultant path of UAV path planning, and l_i denotes the distance between the i-th node and the $i+1$ node.

2) THREAT AREA COST

There are two radii for the threats: One is the detection radius r_0, and the other is the reaction radius r_1. Then the UAV threat area cost can be expressed as

$$Cost_t(x) = \begin{cases} 0, & x > r_0 \\ k_0 \dfrac{x - r_1}{r_0 - r_1}, & r_0 \geq x \geq r_1 \\ k_1 \dfrac{x}{r_1}, & x \leq r_1 \end{cases} \tag{11}$$

where x denotes the distance between the UAV and the threat and $Cost_t(x)$ denotes the threat cost. The surrounding area of the threat is divided according to the distance from the threat. That is, the closer, the more dangerous; the farther, the safer. However, according to the principle of high risk and high return, the closer the path to the threat area, the smaller the cost in the path.

3) FINAL TARGET FUNCTION

$$min : \delta_1 Cost_e + \delta_2 Cost_t \tag{12}$$

where $Cost_e$ denotes the energy consumption cost of UAV path planning mentioned above, which is positively related to the length of the path planned; $Cost_t$ denotes the threat area cost of the enemy threat environment to the UAV; δ_1 denotes the weight coefficient of the energy consumption cost; and δ_2 denotes the weight coefficient of the threat area cost. The target is to minimize the sum of these two cost functions.

III. DESIGN OF UAV GLOBAL-LOCAL PATH PLANNING ALGORITHM

*A. GLOBAL PATH PLANNING BASED ON MODIFIED A**

1) A* ALGORITHM

The greatest difference between the A* algorithm and other path planning algorithms is the composition of the heuristic function [9]. Let $f(n)$ be the heuristic function of the A* algorithm, as shown in Equation 13:

$$f(n) = g(n) + h(n) \tag{13}$$

where n denotes the current node, $g(n)$ denotes the actual cost value from the starting point to the current point n, and $h(n)$ denotes the estimated cost value from the current node n to the end point. The design of the cost function in the heuristic function of the A* algorithm is directly related to the search performance of the A* algorithm.

2) DYNAMIC WEIGHT ADJUSTMENT BASED ON Q-LEARNING

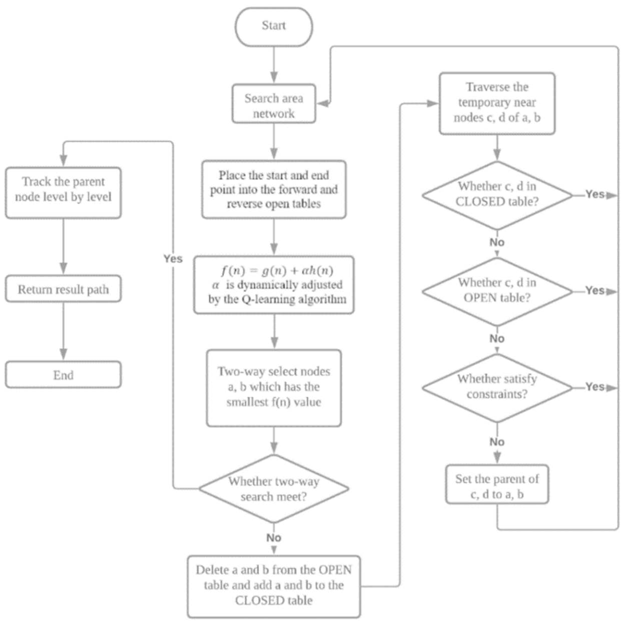
Because the actual cost information $g(n)$ is not considered in the A* algorithm cost function, the path cost of the final planning result will not be the global minimum. In this paper, a cost function of dynamic weight adjustment based on Q-learning was proposed and can be calculated as

$$f(n) = g(n) + \alpha h(n) \tag{14}$$

where α is dynamically adjusted by the Q-learning algorithm to reduce the weight of the estimated cost and increase the weight of the actual cost, which ensures that the algorithm can obtain the path with the minimum comprehensive cost. Moreover, the guiding role of the heuristic factor is retained, which does not slow down the search speed for the path considerably and ensures planning efficiency.

The flow of the global path planning algorithm based on modified A* is shown in Figure 2.

The pseudo code of modified A* is as follows:



Start
Search area network
Place the start and end point into the forward and reverse open tables
$f(n) = g(n) + \alpha h(n)$ α is dynamically adjusted by the Q-learning algorithm
Two-way select nodes a, b which has the smallest f(n) value
Whether two-way search meet?
No
Delete a and b from the OPEN table and add a and b to the CLOSED table
Yes
Track the parent node level by level
Return result path
End
Traverse the temporary near nodes c, d of a, b
Whether c, d in CLOSED table?
Yes
No
Whether c, d in OPEN table?
Yes
No
Whether satisfy constraints?
Yes
No
Set the parent of c, d to a, b

FIGURE 2. Flow chart of modified A*.

Algorithm 1: Modified A*

```
Input: search area, task node Output: path node
Initialize: Q-learning elements, UAV information, constraint information, OPEN table (two-way), CLOSED table (two-way)
While forward search and reverse search have not met do
    Place the forward search node into the forward open table
    Place the reverse search node into the reverse open table
        Place the node with the minimum cost from the forward open table into the forward closed table, delete it from the forward open table, and set the corresponding parent-child relationship
        Place the node with the minimum cost from the reverse open table into the reverse closed table, delete it from the reverse open table, and set the corresponding parent-child relationship
        if (forward and reverse search have not met) then{
        continue;
        }else{
            break;
        }
End While
```

B. LOCAL DYNAMIC PATH PLANNING BASED ON Q-LEARNING WITH MODIFIED EXPLORATION MECHANISM

1) THREE-ELEMENT DESIGN OF MODIFIED Q-LEARNING ALGORITHM

- State space

The state space of modified Q-learning dynamically determines the state according to the UAV path meshing range.

- Action space

According to the Q-value table of the Q-learning algorithm, the corresponding action is selected to increase or decrease in the corresponding state to obtain the state of the next phase. The action space is defined as follows:

$$A = \{a_1, a_2, \ldots, a_{17}\} \tag{15}$$

- Reward mechanism

A reward function that meets the actual application scenario for the agent is designed by analyzing the state after the agent chooses an action. The reward function is calculated as follows:

$$r(i) = \begin{cases} C1, & S_i = S_e \\ -C1, & d_{abs} = 0 \\ 0, & other\ scenario \end{cases} \tag{16}$$

where i denotes the current iteration steps of the algorithm, $r(i)$ denotes the reward function, $\mathrm{S_i}$ denotes the current state of the agent, and $\mathrm{S_e}$ denotes the target state. The distance between the agent and the nearest obstacle is denoted by d_{abs}. C1 is a constant that denotes the reward value obtained after the agent interacts with the environment. Such settings of reward function can be too simple. In most cases, the agent cannot get feedback from the environment and lacks key guidance. For this reason, the time it takes to complete tasks is greatly increased, and energy consumption also increases accordingly.

A new reward function was designed in this paper to solve the abovementioned problems, which categorizes the state–action pair of the agent and returns different reward values for different scenarios. The new reward function adds scenarios: close to the target position and far away from the target position. If the decision made by the agent keeps it away from the target position and no collision occurs, then a small negative feedback value is given to the agent; if the decision made by the agent brings it close to the target position and no collision occurs, then a small positive feedback value is given to the agent. The two newly added scenarios take into account the distance between the agent and the obstacles to dynamically set the reward and punishment function. The modified reward function is calculated as follows:

$$r(i) = \begin{cases} C1, & S_i = S_e \\ -C1, & d_{abs} = 0 \\ C2 * \dfrac{d_i}{D}, & d_i < d_{i-1},\ d_{abs} \neq 0 \\ -C2 * \dfrac{d_i}{D}, & d_i > d_{i-1},\ d_{abs} \neq 0 \\ -C3 * \dfrac{d_i}{D}, & other \end{cases} \tag{17}$$

where d_{i} denotes the distance between the agent and the target location and d_{abs} denotes the distance between the agent and the closest obstacle. C1, C2, and C3 are all constants that denote the specific reward values obtained after the agent interacts with the external environment in different scenarios, C1 > C2 > C3.

2) DYNAMIC EXPLORATION FACTOR

The exploration strategies commonly used in the classic Q-learning algorithm include greedy strategy and ε greedy strategy [10]–[16]. The greedy strategy is to choose the action that maximizes the value for each step in each iteration of the algorithm, that is,

$$Q(s_i, a_i) = argmax_{a_{i+1}} Q(s_{i+1}, a_{i+1}) \tag{18}$$

The ε greedy strategy sets an exploration factor ε to add a random strategy when selecting actions in the algorithm iteration process so that the agent has the probability of ε to choose the action that is most conducive to completing the tasks or maximizes the value. There is a probability of $1 - \varepsilon$ to randomly select an action from the action space, that is,

$$Q(s_i, a_i) \begin{cases} argmax_{a_{i+1}} Q(s_{i+1}, a_{i+1}), & x \le \varepsilon \\ a_i \epsilon A & x > \varepsilon \end{cases} \tag{19}$$

However, the ε greedy strategy also has the problem of unbalanced exploration-exploitation because the value of ε is fixed. Therefore, an exploration method that dynamically adjusts the exploration factor was proposed, which can modify the exploration and exploitation process of the Q-learning algorithm and dynamically adjust in phases, that is,

$$\varepsilon = \begin{cases} \varepsilon_1 * \left(\frac{k}{step_1}\right), & k \le step_1 \\ \varepsilon_1 + \varepsilon_2 * \left(\frac{k - step_1}{step_2}\right) & step_1 \le k \le step_2 \\ \varepsilon = \varepsilon_2 + \varepsilon_3 * \left(\frac{k - step_2}{step_3}\right) & step_2 \le k \le step_3 \end{cases} \tag{20}$$

where ε_1 denotes the value of the initial exploration factor ε in the exploration phase, $\varepsilon_1 \in (0, 0.5)$; ε_2 denotes the value of the initial exploration factor ε in the exploration-exploitation phase, $\varepsilon_2 \in (0, 1)$; ε_3 denotes the value of the initial exploration factor ε in the exploitation phase, $\varepsilon_3 \in (0.5, 1)$; k denotes the current number of iterations of the Q-learning algorithm; $step_1$ denotes the maximum number of iterations of the algorithm in the exploration phase; $step_2$ denotes the maximum number of iterations of the algorithm in the exploration-exploitation phase; and $step_3$ denotes the maximum number of iterations of the algorithm in the exploitation phase.

$k \le step_1$ indicates that the algorithm should be in the exploration phase of experience accumulation. In this phase, because the algorithm has just been iterated, the agent knows nothing about the information of the environment, as well as about how to complete the tasks or maximize the value. In this case, the agent's first choice is to quickly explore the surrounding environment.

$step_1 \le k \le step_2$ indicates that the algorithm should be in the exploration and exploitation phase. In this phase, the agent makes appropriate use of the known environment and accumulated experience in the process of exploration to complete tasks faster or maximize the value. However, because the agent's knowledge of the environment has not yet met the requirements, it is necessary to use existing experience while exploring the unfamiliar environment as much as possible.

The pseudo code of modified Q-learning is as follows:

```
Algorithm 2: Modified Q-Learning
Input: search area, tasks node Output: path node
Initialize: Q table, UAV information, constraint information, action space $A = \{a_1, a_2, \ldots, a_i\}$, action-value function $Q(s_i, a_i)$, where $s_i \in S, a_i \in A$
While $S$ non-terminal state do
    Initialize the state space $S = \{s_1, s_2, \ldots, s_i\}$
    for each step in each round do # the direction of UAV that can be taken in three-dimensional space
        Use the modified search strategy to select an action
        Take the action selected in the previous step to obtain
            the feedback value $r_i$ and the new state $s_{i+1}$
        $(s_i, a_i) = Q(s_i, a_i) + \alpha(r_i + \gamma max_{a_{i+1}} Q(s_{i+1}, a_{i+1}) - Q(s_i, a_i))$
        $s_i \leftarrow s_{i+1}$
    End For
End While
```

$step_2 \le k \le step_3$ indicates that the algorithm has accumulated enough experience and is in the phase of exploiting the experience. In this phase, the agent uses these experiences as much as possible to achieve the goal of completing tasks quickly or maximizing the value. At this time, the value of exploration factor ε becomes greater and approaches 1 as the number of iterations increases.

3) Flow of local dynamic path planning algorithm based on modified exploration mechanism as shown in Figure 3.

IV. SIMULATION EXPERIMENTS

Python was used to perform simulations to verify the performance of the aforementioned global path planning based on modified A* and the local dynamic path planning algorithm based on Q-learning with a modified exploration mechanism. Simulation experiments of path planning were performed on the proposed fusion algorithm using Python, and the algorithm performance was compared before and after fusion. The system configurations for simulation experiments are shown in Table 2.

A. EXPERIMENTAL DESIGN AND ANALYSIS

Scheme 1: The grid size divided by the grid method is 50*50*10, with a total of 25,000 path nodes. The number of static obstacles is randomly set to 1,250, accounting for 5% of the total nodes. The three-dimensional position coordinates of the starting point of the task are set as the origin

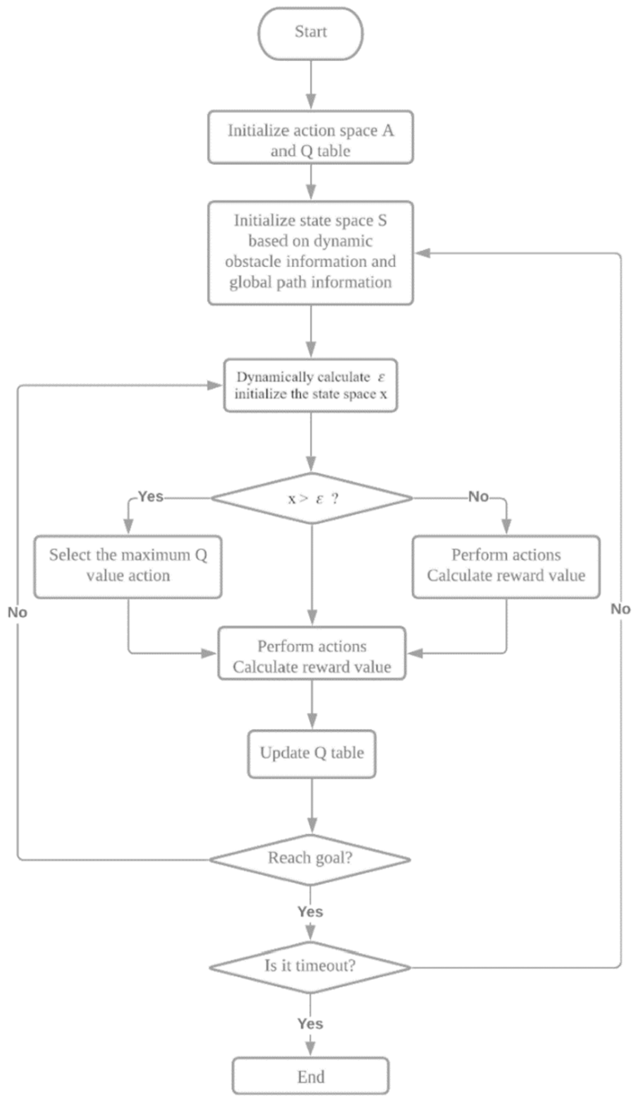

Start
Initialize action space A and Q table
Initialize state space S based on dynamic obstacle information and global path information
Dynamically calculate ε initialize the state space x
x > ε ?
Yes
No
Select the maximum Q value action
Perform actions Calculate reward value
Perform actions Calculate reward value
Update Q table
Reach goal?
No
Yes
Is it timeout?
No
Yes
End

FIGURE 3. Flow chart of modified Q-learning.

TABLE 2. Configuration of the system.

Type	Parameters
Operating system	Windows10
CPU	Intel(R) Core(TM) i9-10900F CPU @ 2.80GHz
Memory	64G
GPU	RTX 3090
Compiler environment	PyCharm 2020.2.2
Development language	Python 3.6.5

S (0, 0, 0) of the grid, and the end point coordinates are set as E (50, 50, 10). The simulation results are shown in Figures 4 and 5.

Scheme 2: The grid size divided by the grid method is 50*50*50, with a total of 125,000 path nodes. The number of static obstacles is randomly set to 6,250, accounting for 5% of the total nodes. The three-dimensional position coordinates of the starting point of the task are set as the origin

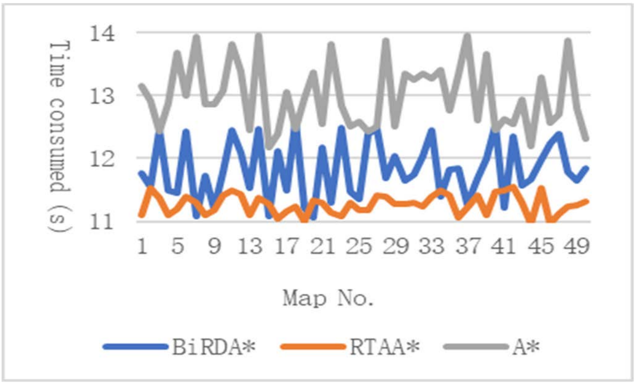


FIGURE 4. Time consumption between algorithms in scheme 1.

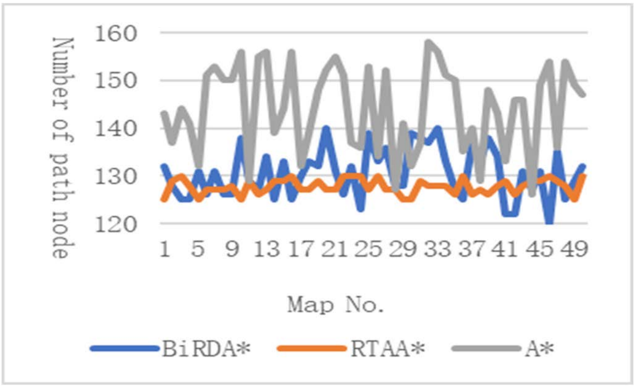


FIGURE 5. Number of path nodes between algorithms in scheme 1.

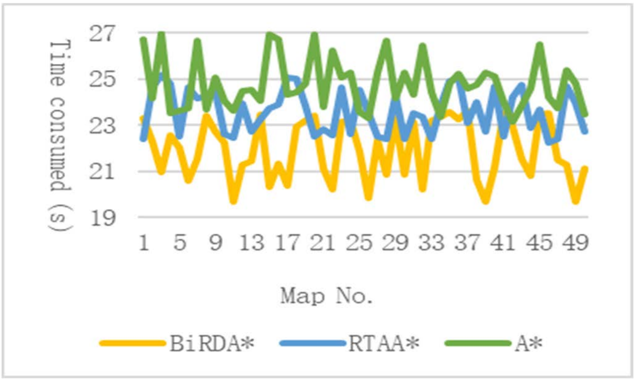


FIGURE 6. Time consumption between algorithms in scheme 2.

S (0, 0, 0) of the grid, and the end point coordinates are set as E (50, 50, 50). The simulation results are shown in Figures 6 and 7.

Scheme 3: The grid size divided by the grid method is 100*100*50, with a total of 50,000 path nodes. The number of static obstacles is randomly set to 25,000, accounting for 5% of the total nodes. The three-dimensional position coordinates of the starting point of the task are set as the origin S (0, 0, 0) of the grid, and the end point coordinates are set as E (100, 100, 50). The simulation results are shown in Figures 8 and 9.

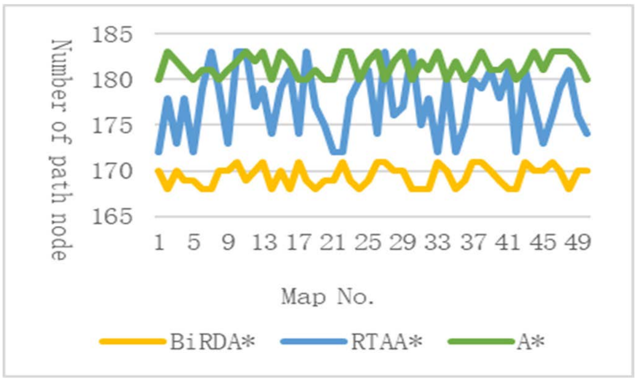


FIGURE 7. Number of path nodes between algorithms in scheme 2.

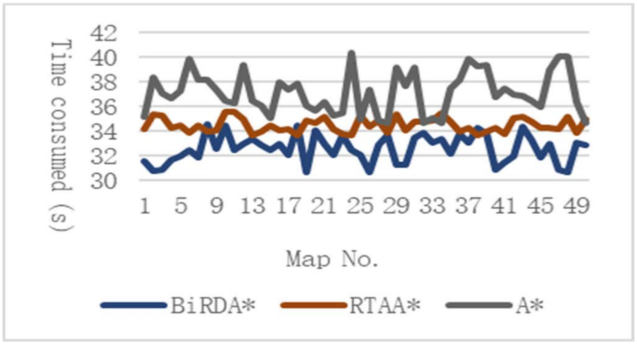


FIGURE 8. Time consumption between algorithms in scheme 3.

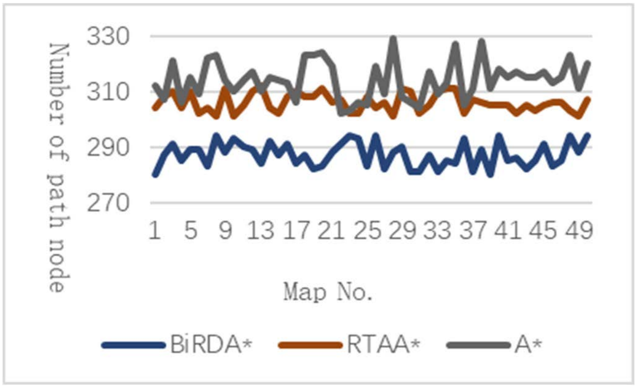


FIGURE 9. Number of path nodes between algorithms in scheme 3.

Path planning was simulated in the established terrain modeling environment. The path planning performances of the modified A* algorithm (BiRDA*), the classic A* algorithm, and the real-time adaptive A* (RTAA*) were compared. Table 3 presents the comparison of the experimental results of the global path planning based on the existing static obstacle information between the two algorithms, including the number of iterations, algorithm time consumption, number of path nodes, and path cost.

Table 3 presents the average values obtained through 50 random initializations of the map and path planning. As Table 3 suggests, compared with the classic A* algorithm

TABLE 3. Comparison of experimental results.

Algorithm	Scheme No.	Iterations	Time consumed (s)	Number of path node	Path cost
A*	Scheme 1	50	13.07	144.2	586.37
RTAA*			11.24	127.7	524.62
BiRDA*			11.85	130.5	539.06
A*	Scheme 2	50	25.09	181.4	761.82
RTAA*			23.72	177.3	745.27
BiRDA*			20.63	169.5	701.98
A*	Scheme 3	50	37.51	314.2	1372.53
RTAA*			34.58	305.9	1286.45
BiRDA*			28.43	287.2	1191.37

and the anytime repairing sparse A* (ARA*) algorithm, the modified BiRDA* algorithm provides shorter planning duration, fewer path nodes, and smaller path cost calculated when the grid space divided by the grid method is large; however, when the scale is small, its performance is not as good as that of the ARA* algorithm.

Figure 4 indicates that some of the classic A* have shorter time consumption because when the map is small, the two-way search increases time consumption. ARA* is more suitable for small-scale situations than A*; therefore, ARA* has the best performance for the scale in Scheme 1. Figure 5 indicates that some of the classic A* and ARA* have fewer path nodes because when the map is small, the modified search strategy with an angle removes some unqualified path nodes. According to Figures 6–9, the overall performance of the algorithm is relatively stable, and the small range of fluctuations that occur are caused by the random initialization of obstacles. The data above shows that the modified A* algorithm has a better convergence speed, shorter path planned, better performance, and better path for path planning on a larger scale; however, when it is applied to a small scale, its time consumption may increase, and its performance is not as good as those of ARA* and classic A*.

The task node information and dynamic threat information are added after setting static obstacles for global path planning in the previous section. When the UAVs are executing tasks such as surveillance, rescue, and delivery, they may encounter interferences to their radar equipment, which are considered threats. For these threats, two radii are randomly initialized: One is the detection radius r_0, and the other is the reaction radius r_1, $r_0 > r_1 > 0$.

Scheme 1: The number of dynamic threats is set to 100, with random locations. The simulation results are presented in Figures 8 and 9.

Scheme 2: The number of dynamic threats is set to 500, with random locations. The simulation results are presented in Figures 8 and 9.

Scheme 3: The number of dynamic threats is set to 1,000. The simulation results are presented in Figures 8 and 9.

The corresponding parameters of the Q-learning algorithm are generated by running dynamic path planning 1,000 times in an environment with the same batch of dynamic obstacles, and their optimal values are determined by the control

TABLE 4. Parameters of Q-learning algorithm.

Name	Value
Maximum iteration $episode_{max}$	500
Maximum number of steps for each iteration $step_{max}$	100000
Learning rate α	0.1
Discount factor γ	0.9
Exploration factor ε	Initialized to 0, dynamically adjusts based on the modified exploration mechanism

TABLE 5. Parameters of dynamic exploration factor.

Name	Value
Exploration factor ε_1 in the initial exploration phase	0.45
Exploration factor ε_2 in the exploration-exploitation phase	0.2
Exploration factor ε_3 in the exploitation phase	0.35
Maximum iterations $step_1$ in the initial exploration phase	225
Maximum iterations $step_2$ in the exploration-exploitation phase	100
Maximum iterations $step_2$ in the exploitation phase	175

TABLE 6. Experimental data design.

Algorithm	Scheme	Iteration(s)	Algorithm time /s	Path node number/n
Q-learning	Scheme 1	500	4.25	12.7
Sarsa			4.58	13.1
AQ-learning			2,79	8.8
Q-learning	Scheme 2	500	7.45	26.5
Sarsa			6.94	24.7
AQ-learning			5.31	19.8
Q-learning	Scheme 3	500	14.87	46.4
Sarsa			12.83	39.5
AQ-learning			12.26	37.9

variable method and the binary search method. Consequently, the parameters of the Q-learning algorithm are obtained, as shown in Table 4, as well as the parameters of the dynamic exploration factor, as shown in Table 5.

In this environment, the path planning simulation is executed, and it is compared with the algorithm based on Q-learning with a modified exploration mechanism (AQ-learning). Table 6 presents the comparison of the performance of the local path planning based on the existing dynamic threat information between the two algorithms, including the number of iterations, algorithm time consumption, and the number of path nodes, in the three schemes.

Table 6 presents the average values obtained through 50 random initializations of the map and 500 iterations of path planning. Because the dynamic adjustment occurs after the UAV global planning is completed, the adjustment extent is small, so the iterations are set to 500 times to take the average value. According to the results in the table, compared with the classic Q-learning algorithm and the Sarsa algorithm,

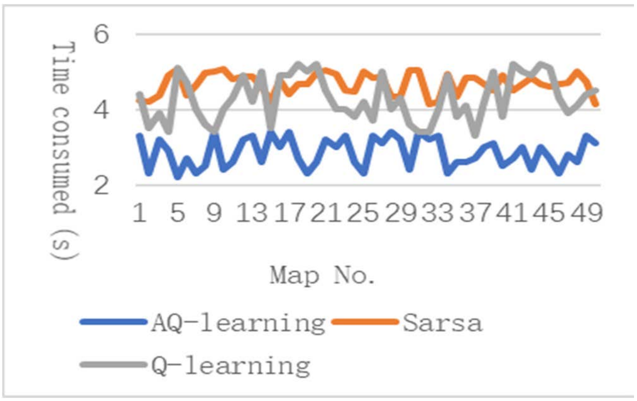


FIGURE 10. Time consumption between algorithms in scheme 1.

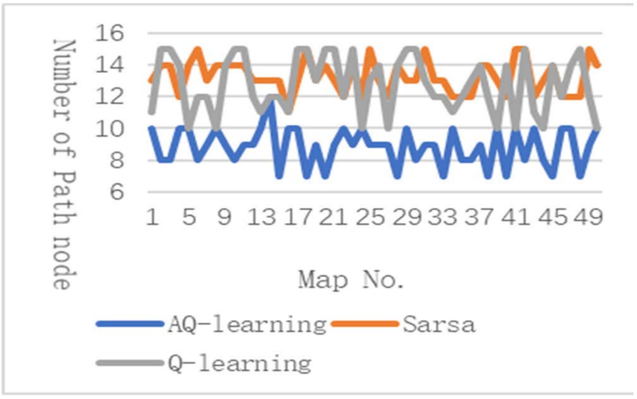


FIGURE 11. Number of path nodes between algorithms in scheme 1.

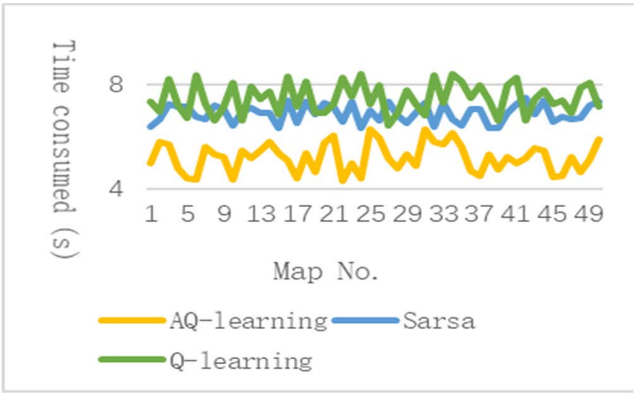


FIGURE 12. Time consumption between algorithms in scheme 2.

the modified Q-learning algorithm has a shorter planning duration and fewer path nodes, but it is not as stable as Sarsa.

As shown in Figure 10, Sarsa runs relatively stable when the scale is small, and the modified Q-learning takes the shortest time but is not as stable as Sarsa. Figure 11 indicates that the modified Q-learning has fewer path nodes. According to Figures 10, 12, and 14, the time consumption of the modified Q-learning fluctuates greatly during dynamic path planning, compared with those of the classic Q-learning and Sarsa. The underlying reason is that although the exploration mechanism

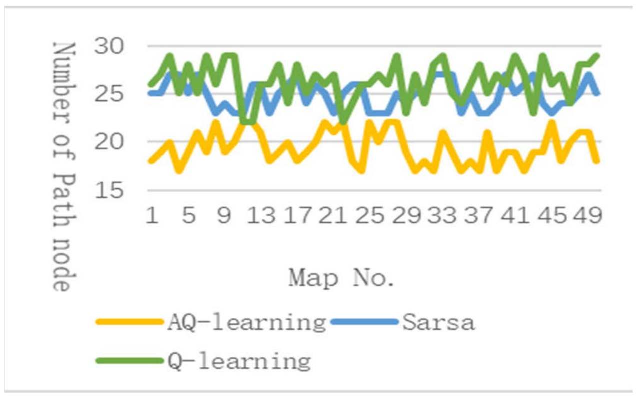


FIGURE 13. Number of path nodes between algorithms in scheme 2.

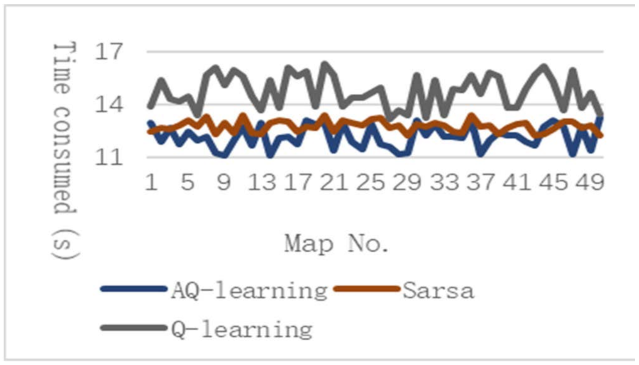


FIGURE 14. Time consumption between algorithms in scheme 3.

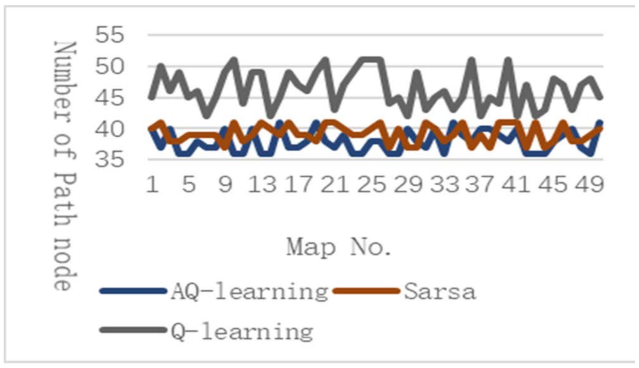


FIGURE 15. Number of path nodes between algorithms in scheme 3.

of Q-learning has been modified, there is still a certain probability for random selection, leading to great fluctuation. Sarsa performs better when the map is large because it is on-policy and infeasible actions are removed when selecting actions. Therefore, its time consumption is better than that of Q-learning at a large scale, but most modified algorithms have advantages. According to Figures 11, 13, and 15, the modified Q-learning algorithm can reduce the number of path nodes to a certain extent. The above shows that the modified Q-learning algorithm has been well modified in response to the limitations of its exploration and exploitation, which can better balance the relationship between exploration and exploitation. When applied to UAV path planning, it also has faster convergence speed and better overall performance.

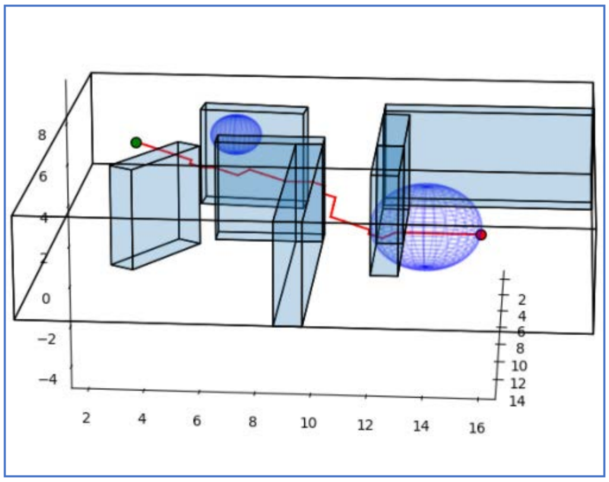


FIGURE 16. Initial path.

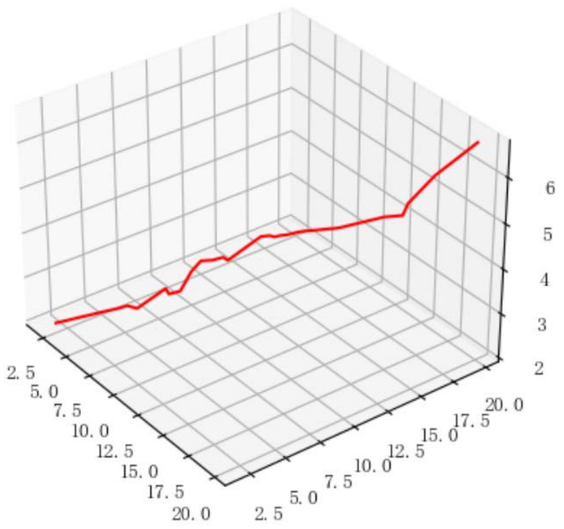


FIGURE 17. Original path.

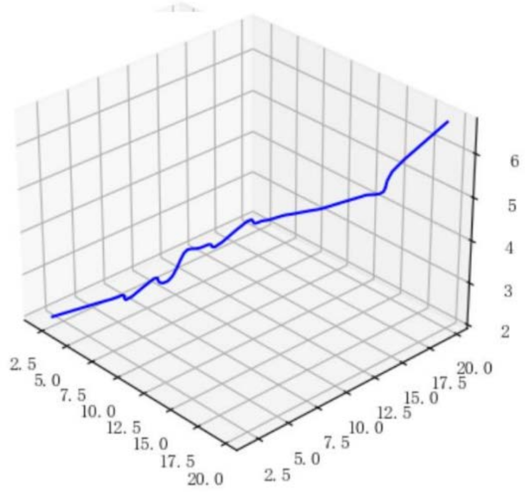


FIGURE 18. Smoothed path.

B. ANALYSIS OF SIMULATION RESULTS

Figure 16 presents the planned path map based on the model in Chapter 2, which converts the latitude and longitude

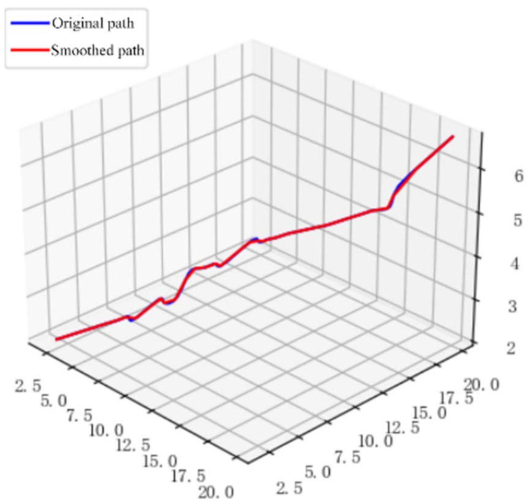


FIGURE 19. Path before and after smoothing.

coordinates of the UAV into three-dimensional coordinates before rasterizing. Figure 19 presents the comparison of the path after the cubic B-spline curve is smoothed.

According to Figures 17–19, the smoothed path has no conflicts with the obstacles, and the overall transition of the path is natural and significantly continuous without sharp corners, which meets the requirements of continuous changes in UAV speed and acceleration.

V. CONCLUSION

In this paper, the research background and key technical problems of path planning were investigated. Through corresponding assumptions, constraints, cost function, evaluation indicators, environmental models, and simulation maps, a global-local UAV path planning model was established, and the effectiveness of the proposed model and algorithms was verified. The main conclusions drawn are as follows:

(1) The global path planning algorithm based on the modified A* and the local path planning algorithm based on the modified Q-learning have positive effects in reducing algorithm time consumption, the number of path nodes, and the path planning cost. Both are better than the classic A* and Q-learning algorithms, with certain guiding significance for UAV path planning.

(2) The proposed algorithm can deliver reasonable and stable path planning for the assigned tasks. Therefore, the proposed algorithm is feasible and effective for solving the problem of UAV path planning.

(3) From the perspectives of the UAV path planning model and corresponding constraint analysis, the proposed model and algorithms are superior to the classic two-dimensional model and the traditional static allocation algorithm (without considering the dynamic threats). Moreover, the UAV environment model considered is a three-dimensional space, and the constraint analysis is more realistic and comprehensive.

Although the proposed model of UAV path planning considers factors such as environment, constraint analysis, and cost function, it lacks a collaborative model of UAV path planning. It is crucial to propose a planning model with multi-UAV collaboration. In addition, the modified exploration mechanism method proposed in this paper adopts a design of phased adjustment and employs some settings for modification when implementing the dynamic exploration factors. One can incorporate better and more complex settings to enhance the effectiveness of the algorithm.

REFERENCES

[1] C. Zhang and W. Fu, "Optimal model for patrols of UAVs in power grid under time constraints," *Int. J. Performability Eng.*, vol. 17, no. 1, pp. 103–113, Jan. 2021.

[2] S. Al-Hasan and G. Vachtsevanos, "Intelligent route planning for fast autonomous vehicles operating in a large natural terrain," *Robot. Auton. Syst.*, vol. 40, no. 1, pp. 1–24, 2002.

[3] A. Richards and J. P. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," in *Proc. Amer. Control Conf.*, vol. 3, May 2002, pp. 1936–1941.

[4] J. Wang, W. Chi, C. Li, C. Wang, and M. Q.-H. Meng, "Neural RRT*: Learning-based optimal path planning," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 4, pp. 1748–1758, Oct. 2020.

[5] N. Ozalp and O. K. Sahingoz, "Optimal UAV path planning in a 3D threat environment by using parallel evolutionary algorithms," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, May 2013, pp. 308–317.

[6] U. Cekmez, M. Ozsiginan, and O. K. Sahingoz, "Multi colony ant optimization for UAV path planning with obstacle avoidance," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2016, pp. 47–52.

[7] M. Radmanesh and M. Kumar, "Grey wolf optimization based sense and avoid algorithm for UAV path planning in uncertain environment using a Bayesian framework," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2016, pp. 168–179.

[8] I. Sung, B. Choi, and P. Nielsen, "On the training of a neural network for online path planning with offline path planning algorithms," *Int. J. Inf. Manage.*, vol. 57, Apr. 2021, Art. no. 102142.

[9] Y. Bengio, "Deep learning of representations: Looking forward," in *Proc. Int. Conf. Stat. Lang. Speech Process.*, in Lecture Notes in Computer Science, vol. 7978, Jul. 2013, pp. 1–37.

[10] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," *IEEE Trans. Neural Netw.*, vol. 9, no. 5, p. 1054, Sep. 1998.

[11] W. Feng and Y. Wu, "DDoS attack real-time defense mechanism using deep Q-learning network," *Int. J. Performability Eng.*, vol. 16, no. 9, pp. 1362–1373, Sep. 2020.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, and J. Veness, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[13] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

[14] G. Sartoretti, J. Kerr, Y. Shi, G. Wagner, T. K. S. Kumar, S. Koenig, and H. Choset, "PRIMAL: Pathfinding via reinforcement and imitation multi-agent learning," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2378–2385, Jul. 2019.

[15] M. Duguleana and G. Mogan, "Neural networks based reinforcement learning for mobile robots obstacle avoidance," *Expert Syst. Appl.*, vol. 62, pp. 104–115, Nov. 2016.

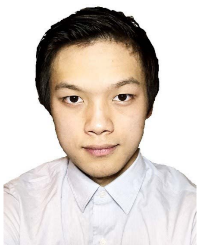
[16] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, and J. Schrittwieser, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

DONGCHENG LI received the B.S. degree in computer science from the University of Illinois of Springfield and the M.S. degree in software engineering from The University of Texas at Dallas, where he is currently pursuing the Ph.D. degree. His research interests include search-based software testing and intelligent optimization algorithms.

WANGPING YIN received the B.S. and M.S. degrees in computer science from the China University of Geosciences, Wuhan. His research interest includes intelligent optimization algorithms.

W. ERIC WONG received the M.S. and Ph.D. degrees in computer science from Purdue University. He is currently a Full Professor and the Founding Director of the Advanced Research Center for Software Testing and Quality Assurance in Computer Science, The University of Texas at Dallas (UTD). He also has an appointment as a Guest Researcher with the National Institute of Standards and Technology (NIST), Agency of the U.S. Department of Commerce. Prior to joining UTD, he was with Telcordia Technologies (formerly Bellcore) as a Senior Research Scientist and the Project Manager, where he was in-charge of dependable telecom software development. In 2014, he was named the IEEE Reliability Society Engineer of the Year. His research interest includes helping practitioners improve the quality of software while reducing the cost of production. In particular, he is working on software testing, debugging, risk analysis/metrics, safety, and reliability. He has very strong experience developing real-life industry applications of his research results. He is the Editor-in-Chief of IEEE Transactions on Reliability. He is also the Founding Steering Committee Chair of the IEEE International Conference on Software Quality, Reliability, and Security (QRS).

MINGYONG JIAN received the bachelor's degree in network engineering from the China University of Geosciences, Wuhan, where he is currently pursuing the master's degree in computer technology. His research interests include intelligent dispatching and information engineering.



MATTHEW CHAU is currently the bachelor's degree in electrical engineering with The University of Texas at Dallas. His research interests include software fault prevention, software reliability, and preventive engineering.