Multi-UAV Collaborative Path Planning using Hierarchical Reinforcement Learning and Simulated Annealing

Yuting Cheng\textsuperscript{a}, Dongcheng Li\textsuperscript{b}, W. Eric Wong\textsuperscript{b}, Man Zhao\textsuperscript{a,*}, and Dengfeng Mo\textsuperscript{a}

\textsuperscript{a}School of Computer Science, China University of Geosciences, Wuhan, 430074, China
\textsuperscript{b}Department of Computer Science, University of Texas at Dallas, 75082, USA

Abstract

In practice, classical path optimization algorithms perform poorly when applied to an unknown environment, swarm intelligence algorithms need further improvement in agility and accuracy to avoid a moving object in dynamic environment, and reinforcement learning algorithm, a usual solution adopted in machine learning, may give rise to curse of dimensionality due to the complexity of scenario. In view of aforesaid practical problems, this paper proposes using MAXQ hierarchical reinforcement learning method to achieve dimensionality reduction by abstraction and combining leader-wingman approach with dynamic dead zone to model after cooperative formation and design triangular form. A novel algorithm based on MAXQ and simulated annealing is designed to solve unmanned aerial vehicle (UAV) path planning problem, which accomplishes grid method-based path planning simulation in 2D scenarios. A comparative analysis is performed on Q-Learning, ε-Q-Learning, standard MAXQ and SA-MAXQ algorithms in terms of their convergence, time consumption and search steps. Moreover, leader-wingman method is combined with dynamic dead zone in modelling triangular form for Multi-UAV collaborative formation. The experimental results indicate SA-MAXQ algorithm yields quicker astringence, lower volatility, better learning effect, less time consumed and optimized searched route.

Keywords: path planning; UAV collaboration; MAXQ hierarchical reinforcement learning; simulated annealing

(Submitted on January 12, 2022; Revised on March 20, 2022; Accepted on May 10, 2022)

© 2022 Totem Publisher, Inc. All rights reserved.

1. Introduction

Path planning is the most researched topic in unmanned aerial vehicle (UAV) navigation with the primary purpose of finding out an optimal route from departure to destination. In addition to finding out an optimal and shortest route, path planning needs also to provide a collision-free environment for UAV. A computation-based safe route path planning is expected to navigate the UAV to avoid any possible obstacle and reach the destination as soon as possible [1].

At present, there are mainly three types of multiagent path planning algorithms. First type consisting of classical path planning algorithms such as A-Star [2] and rapidly-exploring random tree (RRT) [3] which are preconditioned on knowledge about the application environment. Second type is intelligent optimization-based algorithms that plans route by computing relative relation between real-time dynamic environmental factors and agent location [4], including genetic algorithm (GA) [5], and artificial bee colony (ABC) [6] algorithm. Last type is machine learning-based algorithms using autonomous self-learning strategy to search optimal route and it includes neural network algorithm [7], reinforcement learning algorithm (RL) and hierarchical reinforcement learning algorithm [7-10].

A common practice now is to adopt swarm intelligence algorithm for UAV path planning. This type of algorithms though performs well in path planning, it needs prior knowledge about global environment in which the UAV is planned to fly. When dealing with real-time dynamic and complicated flight environment, swarm intelligence algorithms appears to perform poorly in planed route and flight agility [11].

How to plan route in unknown and complicated ever-changing environment is one of the hot topics in current path planning field. In recent years, many scholars have introduced machine learning into path planning problem in an attempt to find new and better solutions. Numerous studies have shown such algorithms (e.g., neural network and deep reinforcement learning
could cope with difficulty in path planning in unknown environment. Although reinforcement learning has made success in UAV flight behavior research, it may lead to “curse of dimensionality” in learning space when being applied to coordinating multiple UAVs in flight. To solve the problem, this paper propose a hierarchical reinforcement learning-based method with simulated annealing strategy for multi-UAV collaborative path planning to meet autonomous flight demand and avoid collision.

The preceding sections of the paper are arranged as follows. Section 2 introduces the design and modelling of multi-UAV collaborative path planning. Section 3 presents the proposed solution and algorithmic approach for UAV 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.

2. A Multi-UAV Collaborative Path Planning Model

2.1. Problem Formulation and Basic Assumptions

Multi-UAV collaborative path planning aims to find an optimal flight path from the start point to the destination point. Along the path, UAV should be able to avoid all static obstacles such as mountain peak and building as well as all the dynamic obstacles like other UAVs and birds. In addition to avoiding obstacles, UAV group should have the capability of collaborative network formations and flight. In multi-UAV path planning, it is necessary to solve two problems, namely collaborative flight and path planning. For the former, during the flight, UAVs in a group or network should not collide with each other in the process of moving; while for the latter, one optimal or sub-optimal path should be obtained to help UAVs to avoid all the obstacles.

Real UAV flight involves diverse scenes and complicated environment, and such factors are challenging to be considered in the modelling. The main goal of this study is to carry out path planning and optimize search steps using hierarchical reinforcement learning. Thus, the model in this study is simplified and some preset conditions were set without influencing multi-UAV collaborative path planning to obtain the optimal path and verify the applicability through experiment simulation. The basic assumptions made in this paper are as follows:

1) Consider UAV, mountain and so on as point in a map.
2) The 2D model scene is used without considering the height of obstacles or UAVs.
3) Consider static obstacles in the environment as fixed points and dynamic obstacles as real-time moving obstacles including other UAVs.
4) Assume narrow flight path which a single UAV cannot pass does not exist in the flight environment.
5) Not take into account the effect of factors such as climate and wind direction on UAV.
6) Not take into account the effect of airflow of leader UAV on rear UAVs within the formation.

2.2. A Multi-UAV Path Planning

This section discuss the constraints mentioned in paper. The restraints in UAV path planning are divided into two parts: those from UAV itself and those from flight environment. To guarantee the result of path planning, max and min turning angles are required in path generation, or UAV’s turning angles should be defined in a way to ensure vehicle safety as it faces constraints from flight direction and its performance in 2D environment.

2.2.1. Path Planning Environment Model Design

The environment state should be modeled first to form search space for UAV path planning. Environment modeling in 2D space usually turns to grid, probabilistic road-map and artificial potential field based-methods. Grid-based method could be divided into grid method and tree method by whether divided spaces remain equal in size. Since grid method is more straightforward more visible, it is employed in this paper to build 2D discrete model of UAV flight space. The grid method converts 2D UAV flight scene into a space made up of several standard square grids, while each grid is represented with a 2D coordinates in which the x-axis is the rightward horizontal line and the y-axis is the downward vertical line. UAV is set to fly from the starting point (1,1). In this paper, the unit length of each grid is set to 10. Grid space of various lengths and widths have been designed to test the effect of UAV path planning.

2.2.2. Hierarchical Structure Design

The modeling is to divide UAV path planning task into several subtasks and distribute those subtasks layer by layer. Root task is divided into avoid subtask and approach subtask. The avoid subtask is further divided into subtasks for avoiding two specific obstacles. Finally, those three subtasks are decomposed into a series of basic movements as shown in Figure 1. A bottom-up UAV learning model is formed to solve UAV path planning. After basic movements are determined, subtasks are solved through layer-wise recursive invocation until the root task is solved, which provide a solution to the overall UAV path planning.
3. Design of UAV Path Planning Algorithm using Hierarchical Reinforcement Learning

3.1. MAXQ Hierarchical Reinforcement Learning

MAXQ is a hierarchical reinforcement learning algorithm based on value function decomposition. It decompose UAV path planning task into a set of subtasks \( \{M_0, M_1, ..., M_n\} \) and strategy \( \pi \) into corresponding subset \( \{\pi_0, \pi_1, ..., \pi_n\}\)\[16\]. The learning task forms a hierarchical structure with \( M_0 \) as the root task. Subtasks are solved layer-wise from bottom to top until the root task is solved. The nodes in hierarchical task model are not arranged in sequence. Instead, it is up to task of previous layer to decide which subtask on next layer should be executed first \[17\]. MAXQ completes a task by solving all subtasks therein. Since subtask demands less state action space, it could be solved rather quickly. Besides, task in previous layer could also adopt subtask strategy to speed up solving process of the overall task.

Suppose \( V_n(i, s) \) as the expected return from each state of subtask \( M_i \), \( P_i(s', N|s,a) \) as state transfer function, \( R_i(s,a) \) as immediate reward function, then evaluation function is generated as shown in Formula (1) in which \( V^\pi(i,s') \) indicates cumulative evaluation value as from departure of \( s' \) to completion of subtask \( M_i \).

\[
V^\pi(i,s') = V^\pi(\pi_i(s),s) + \sum_{s',N} P_i(s',N|s,\pi_i(s)) \gamma^N V^\pi(i,s')
\]  

(1)

State-action value function is illustrated in Formula (2) while completion function in Formula (3),

\[
Q^\pi(i,s,a) = V^\pi(a,s) + \sum_{s',N} P_i(s',N|s,a) \gamma^N Q^\pi(i,s',\pi(s'))
\]  

(2)

\[
C^\pi(i,s,a) = \sum_{s',N} P_i(s',N|s,a) \gamma^N Q^\pi(i,s',\pi(s'))
\]  

(3)

Based on the three formulas above, Q function can be expressed as in Formula (4),

\[
Q^\pi(i,s,a) = V^\pi(a,s) + C^\pi(i,s,a)
\]  

(4)

Suppose root task \( M_0 \) chooses subtask \( M_{a_1} \) as per corresponding action selection strategy and \( M_{a_1} \) chooses \( M_{a_2} \) as per the same strategy. The operation goes on until the basic action \( a_n \) is executed. Evaluation value of state \( s \) in subtask of root node can be decomposed as shown in Formula (5). Top-down choosing action and bottom-up recursive feedback evaluate value are employed to accomplish standard MAXQ algorithm.

\[
V^\pi(0,s) = V^\pi(a_n,s) + C^\pi(a_{n-1},s,a_n) + ... + C^\pi(a_1,s,a_n) + C^\pi(0,s,a_n)
\]  

(5)

At this time, both MAXQ and standard Q-Learning algorithms choose next action based on greedy strategy in solving path planning. When it comes to action setup, standard MAXQ algorithm has different actions available for selection due to its hierarchical structure, though its basic actions are consistent with that available for Q-Learning (namely 8 flight directions for a UAV). As illustrated in Figure 1, there are only two optional actions in UAV path planning: avoid and approach. Under avoid subtask, there are another two actions available for selection: avoid static obstacle and avoid dynamic obstacle. By contrast, under approach subtask, there is only one action available, namely approach target.

3.2. Simulated Annealing Algorithm

To solve a problem using simulated annealing algorithm, the optimal solution is to figure out the lowest energy in a system. Common physical annealing for a solid object demands a temperature rise on the object and brings it to melt first and then
lets it cool down slowly. The cooling process cannot be too fast so that the object could stay balanced at every temperature when it cools down. Annealing should be slow enough; otherwise the object can hardly be crystalized.

In accordance with Monte Carlo method, when object transfers from state $i$ to state $j$, state $j$ with higher internal energy is likely to become the new state at a probability of $\exp(E_j - E_i)/k \cdot T$ in which $k$ is the Boltzmann constant. Therefore, in simulated annealing process, when temperature is $T$, the probability for particle’s internal energy to reach a balance is $\exp(\Delta E)/k \cdot T$ in which $\Delta E$ indicates difference in internal energy before and after temperature change. Monte Carlo method is expressed in Formula (6) in which $\exp$ denotes exponential and $p$’s value range is $(0, 1)$.

$$p(i \rightarrow j) = \begin{cases} 1 & E(i) \leq E(j) \\ \exp \left( \frac{E(i) - E(j)}{k \cdot T} \right) & \text{other} \end{cases} \quad (6)$$

To find the optimal solution, when simulated annealing algorithm such as improved hill climbing algorithm is adopted as shown in Figure 2, the algorithm may take a poor solution at certain probability. For instance, after reaching point C, though solution at this point is not as good as that at point B, simulated annealing may take point C as the current state at certain probability. After taking the poor solution D at certain probability for a few times, the algorithm will be able to jump away from local optimal value at point B and reach the global optimal solution at point E. In this way, it could effectively prevent the algorithm from being caught in local optimal solution.

![Figure 2. Schematic diagram of hill climbing algorithm](image)

The probability for simulated annealing algorithm to accept the less good solution is not constant. It may decline over time before approaching a stable state.

3.3. SA-MAXQ for UAV Path Planning

3.3.1. Modified Action Selection Strategy based on Simulated Annealing Algorithm

Standard MAXQ hierarchical reinforcement learning adopts greedy strategy for selecting actions. It is susceptible to local optimal solution, dissatisfactory convergence effect and longer planned route for UAV. In view of aforesaid shortcomings of greedy strategy, simulated annealing algorithm is adopted hereby to improve hierarchical reinforcement learning-based action selection strategy.

For simulated annealing algorithm, temperature drops slowly. The cooling speed is regulated by a temperature attenuation factor $r$. Larger $r$ indicates longer search time but higher chance of gaining global optimal solution. Otherwise, the search time could be shortened, but global optimal solution could be missed as well while the solving process is caught in local optimal solution. Therefore, attenuation factor $r$ plays a vital role in action selection by simulated annealing algorithm. There are some common forms of attenuation: $T_i$ indicates the temperature at $k$ iteration, $K$ indicates total number of iterations, and $\mu$ ranges within $(0, 1)$. In this paper, the first temperature attenuation formula is chosen. In order to guarantee larger search space, $\mu$ is preset to be 0.95.

$$T_k = \mu^k T_0 \quad (7)$$
$$T_k = \frac{1}{k} T_0 \quad (8)$$
$$T_k = \frac{k^2 - k}{k} T_0 \quad (9)$$

Though whether simulated annealing algorithm could generate the optimal solution is concerned with initial state, higher initial preset temperature could result in larger search space and higher chance of gaining the global optimal solution. To this end, in this paper initial temperature $T_0$ is set to be 100°C to make sure better search effect and higher chance of gaining the global optimal solution.

In MAXQ hierarchical reinforcement learning, simulated annealing algorithm with 0.95 temperature attenuation factor and 100°C initial temperature is chosen. Upon each action selection, a random factor $\in [0, 1]$ will be generated to be compared with current $\exp(\Delta E)/T$. If random factor is lower than $\exp(\Delta E)/T$, an action will be chosen randomly; or the action with higher reward value will be opted. Such selection strategy could raise the chance of gaining a global optimal solution and make the trained path more adapted to operating scenes. The pseudocode of simulated annealing-based MAXQ hierarchical reinforcement learning (SA-MAXQ) algorithm is illustrated in Algorithm 1:
Algorithm 1: SA-MAXQ

Input: action $a$, state $s$, learning rate $\alpha$, iscount rate $\gamma$, StateSeq, timesteps $N$, Action strategy $SA_{act}$
Output: StateSeq

Process:
Repeat (for each episode)
SA-cooldown $T = r*T$
SA-MAXQ:
Initialize StateSeq
If $a$ is a primitive action:
$r_{sum} <- r_{sum} + r$
$V(a,s) <- \alpha*r+(1-\alpha)*V(a,s)//update evaluate function$
Update StateSeq
Else if
newAction <- $SA_{act}(a,s)//use SA Strategy choose action
Initialize childStateSeq
$SA_{MAXQ}(newAction,s)$
$N < - childStateSeq$.length
Repeat:
Update $Q^\pi(a,s,newActiona)$
$N < -$
Until $N=0$
Update StateSeq
Until is terminal
return StateSeq

The actions available for MAXQ hierarchical reinforcement learning algorithm to choose vary with specific hierarchical structure. In this paper, UAV path planning task is decomposed into a three-layer framework, and there are only eight basic actions for a UAV, namely eastward, westward, southward, northward, northeastward, southeastward, northwestward and southwestward as shown in Figure 1.

3.3.2. Hierarchical Framework of Path Planning Task

The execution of basic actions by a UAV should satisfy requirements posed by its own performance parameters. The whole task could be decomposed into the following subtasks:
1) Root node: overall collaborative planning task
2) Approach subgoal: move UAV towards the goal;
3) Avoid static obstacle: enable UAV to avoid any detected fixed obstacles such as mountain peak;
4) Avoid dynamic obstacle: enable UAV to avoid colliding with other UAVs;

In the whole UAV hierarchical framework, each subtask is determined by child action. Completion of one child action indicates completion of the task. In SA-MAXQ, each subtask could choose one child action as per current simulated annealing strategy, while child action is equal to subtask at next layer that chooses corresponding child action as per action strategy. The recursive invocation used here to complete child actions is like a subprogram. The MAXQ nodes in UAV path planning are shown in Figure 3.

In Figure 3, triangular nodes are used to denote $V^\pi(i,s)$ of each task and Q node to represent completion function $C^\pi(i,s,a)$ with parent action being $i$ and child action being $a$ in state $s$.

3.3.3. Reward and Evaluation Function for Path Planning

In reinforcement learning, reward function $R$ denotes immediate reward for selection of action $a$ in given state. Reward function design could directly affect training effect. In this paper, UAV path planning adopts hierarchical structure which consists of three subtasks, namely approach, avoid static obstacle and avoid dynamic obstacle. To cope with those three subtasks, corresponding reward functions $R_{move}$, $R_{avoidS}$ and $R_{avoidD}$ are designed, respectively. They are consistent with previously stated standard MAXQ algorithm and expressed as formulas (10), (11), and (12).

$$R_{move} = \begin{cases} 
20 & \text{Reach target point} \\
-1 & \text{UAV moves} \\
-10 & \text{Collision} \\
0 & \text{other}
\end{cases}$$ (10)
In this paper, SA-MAXQ-based path planning is to divide the whole path planning task $M$ into a set of subtasks $\{M_0, M_1, \ldots, M_n\}$. In this set, there are following subtasks: execute basic actions, avoid static obstacle, avoid dynamic obstacle, move towards child node, avoid, move, and path planning. In the meanwhile, strategy $\pi$ is decomposed to form a strategy set $\{\pi_0, \pi_1, \ldots, \pi_n\}$ in which $\pi_i$ is the strategy of $M_i$. In the hierarchical structure, assume the first action a chosen by strategy $\pi_i$ of avoiding static obstacle subtask is to fly eastward and the system ends in state $s'$ after $N$ time steps. At this time, state transfer probability is $P_i^m(s', N|s, a)$. According to Bellman formula, we could get formula (13):

$$V^\pi(i, s') = V^\pi(\pi_i(s), s) + \sum_{s''} P_i^m(s'', N|s, \pi_i(s)) \gamma^N V^\pi(i, s'')$$

Converting the formula above into Q value expression so that $Q^\pi(i, s, a)$ denotes the cumulative reward from execution of action $a$ in state $s$ to completion of subtask $M_i$ as shown in Formula (14) in which $N$ indicates time step and $\gamma$ indicates discount rate:

$$Q^\pi(i, s, a) = V^\pi(a, s) + \sum_{s''} P_i^m(s'', N|s, a) \gamma^N Q^\pi(i, s'', \pi(s''))$$

In hierarchical reinforcement learning, completion function $C^\pi(i, s, a)$ is defined as the cumulative discount evaluation of completion of subtask $M_i$ through invocation of subtask $M_a$ in state $s$. For example, in state $s$, in order to complete move subtask, move towards goal subtask is invoked and all sorts of basic actions are invoked by the move towards goal subtask. In this process, cumulative discount evaluation value is kept in completion function $C^\pi(i, s, a)$.

$$C^\pi(i, s, a) = \sum_{s''} P_i^m(s'', N|s, a) \gamma^N Q^\pi(i, s'', \pi(s''))$$

Therefore, Q value function could be converted into following formula based on Formulas (13), (14) and (15):

$$Q^\pi(i, s, a) = V^\pi(a, s) + C^\pi(i, s, a)$$

Therefore, in state $s$, execution of subtask $i$’s evaluation function $V^\pi(i, s)$ could be expressed as Formula (17) in which $a$ denotes chosen action strategy, $s'$ denotes post-transfer state, $P$ denotes state transfer probability function, and $r$ denotes reward-punishment function.

$$V^\pi(i, s) = \begin{cases} 
Q^\pi(i, s, a) & \text{i is a compound task of avoidance, tendency, etc.} \\
\sum_{s'} P(s'|s, i) r(s'|s, i) & \text{i is the eight basic movements}
\end{cases}$$
Parameters involved in SA-MAXQ-based UAV path planning algorithm are listed in Table 1.

Table 1. SA-MAXQ-based UAV path planning parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_{move} )</td>
<td>Move towards goal reward function</td>
</tr>
<tr>
<td>( R_{avoid} )</td>
<td>Avoid static obstacle reward function</td>
</tr>
<tr>
<td>( R_{avoid_d} )</td>
<td>Avoid dynamic obstacle reward function</td>
</tr>
<tr>
<td>([M_0, M_1, \ldots, M_n])</td>
<td>Subtask set</td>
</tr>
<tr>
<td>([\pi_0, \pi_1, \ldots, \pi_n])</td>
<td>Strategy set</td>
</tr>
<tr>
<td>( s )</td>
<td>Current state</td>
</tr>
<tr>
<td>( s' )</td>
<td>Next state</td>
</tr>
<tr>
<td>( V^\alpha(t,s) )</td>
<td>Evaluation function</td>
</tr>
<tr>
<td>( P^\alpha )</td>
<td>Transfer probability</td>
</tr>
<tr>
<td>( \gamma )</td>
<td>Discount rate</td>
</tr>
<tr>
<td>( N )</td>
<td>Time step</td>
</tr>
<tr>
<td>( T_0 )</td>
<td>Initial temperature</td>
</tr>
<tr>
<td>( r )</td>
<td>Temperature attenuation factor</td>
</tr>
</tbody>
</table>

4. Experimental design and result analysis

This section primarily verifies multi-UAV collaborative path planning through simulation experiment and comparatively analyzes MAXQ, SA-MAXQ, Q-Learning, and ε-Q-Learning algorithms in terms of convergence, time consumption, steps and other evaluation parameters.

4.1. Experimental Design

In this paper, we assume UAVs fly in a 2D scene without any height limit. Grid method is applied to converting the flight space for UAVs into a scene in 2D coordinate system. We have designed several spatial matrixes of different sizes to compare and analyze path planning effect. Algorithms are compared with evaluation parameters in a 22*10 grid area. Apart from that, three grid areas, namely 22*10, 50*45 and 100*70, are used in simulated plotting.

In the grid area, the initial starting point, goal and static/dynamic obstacles for UAV cluster are determined by virtue of coordinates. As shown in Figure 4, a triangular UAV cluster is formed with three green dots while a red dot indicates the final goal for them. Blue dot indicates static obstacle and yellow dot indicates dynamic one which is generated dynamically during UAV flight.

In this paper, control variable method is employed to design three sets of data to compare algorithms in term of their time consumption in order to find out the parameters with best effect. The designed data are listed in Tables 2-4.

An experiment based on three sets of data as listed above is performed to figure out the parameters with best training effect. Resulting parameters are used as standard ones in modified MAXQ algorithm to compare with other algorithms.

![Figure 4. Grid scenes for UAV path planning](image-url)
4.2. Experimental Results and Analysis

Based on control variable method, this paper computes total time consumed under various variables but in the same scene. As shown in Tables 5-7, total time consumed remains least when learning rate is 0.2, discount factor is 0.95, and attenuation factor is 0.95. Thus, those parameters are standardized in subsequent algorithm comparison and analysis.

Using control variable method, this paper laterally compares standard MAXQ hierarchical reinforcement learning algorithm, simulated annealing-based MAXQ hierarchical reinforcement learning algorithm (SA-MAXQ), Q-Learning algorithm and ε-Q-Learning algorithm [18] in term of their effect in solving path planning issue.

ε-Q-Learning is a modified Q-Learning algorithm which employs the thought of dynamically adjusted greedy factor as per action selection strategy. In this algorithm, greedy factor changes as per success or not of each path training. If path training succeeds, greedy factor decreases to reinforce the purposiveness of path optimization; otherwise, greedy factor rises to prevent the solving process from being caught in local optimal solution. It improves action strategy as well and acts as a competent contrast to SA-MAXQ. Primary parameters used in the experiment are listed in Table 8:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning rate α</td>
<td>0.2</td>
</tr>
<tr>
<td>Discount factor γ</td>
<td>0.95</td>
</tr>
<tr>
<td>Exploratory factor ε</td>
<td>0.1</td>
</tr>
<tr>
<td>Training step</td>
<td>5000 (22*10)</td>
</tr>
<tr>
<td></td>
<td>20000 (50*45)</td>
</tr>
<tr>
<td></td>
<td>50000 (100*70)</td>
</tr>
<tr>
<td></td>
<td>50000 (200*150)</td>
</tr>
<tr>
<td>Annealing attenuation factor r</td>
<td>0.95</td>
</tr>
<tr>
<td>Initial temperature T₀</td>
<td>100</td>
</tr>
<tr>
<td>Move reward</td>
<td>-1</td>
</tr>
<tr>
<td>Target reward</td>
<td>20</td>
</tr>
<tr>
<td>Collision reward</td>
<td>-10</td>
</tr>
<tr>
<td>Growing rate θ</td>
<td>0.005</td>
</tr>
<tr>
<td>Running scale</td>
<td>22<em>10, 50</em>45, 100*70</td>
</tr>
<tr>
<td>Starting point</td>
<td>(1, 1)</td>
</tr>
<tr>
<td>Target point</td>
<td>(9, 16)</td>
</tr>
<tr>
<td></td>
<td>(48, 44)</td>
</tr>
<tr>
<td></td>
<td>50*45</td>
</tr>
<tr>
<td></td>
<td>(90, 69)</td>
</tr>
<tr>
<td></td>
<td>100*70</td>
</tr>
<tr>
<td></td>
<td>(197, 149)</td>
</tr>
<tr>
<td></td>
<td>200*150</td>
</tr>
</tbody>
</table>

4.2.1. Convergence

Convergence is an important indicator for measuring algorithm effect. With parameters listed in Table 8, Q-Learning algorithm, ε-Q-Learning algorithm, standard MAXQ algorithm and SA-MAXQ algorithm are compared from their convergence in a 22*10 grid area as shown in Figures 5-8.

A comparative analysis suggests modified SA-MAXQ appears to have better convergence effect, because it gets converged when training generations reach 1,000. By comparison, standard MAXQ just starts to converge by this time. As a result, SA-MAXQ algorithm gets converged more quickly and presents better convergence effect.

When Q-Learning and ε-Q-Learning algorithms are compared as they undergo 5,000 generations of training, we can find the former starts converging when 1,500 generations have been trained and presents less satisfactory convergence effect than MAXQ hierarchical reinforcement learning. For both non-modified and modified MAXQ algorithms, when training generations reach 2,000, their gained reward value tends to stable with little fluctuation. In contrast, Q-Learning algorithm has extremely higher reward value, indicating it has made some many mistakes in finding out the optimal path and its training
effect is not good. Even when it tends to converge after 2,000 generations of training, it still contains significant fluctuation in reward value.

Therefore, we may draw a conclusion that both standard MAXQ algorithm and Q-Learning algorithm that adopt greedy algorithm as their action selection strategy appears to be less stable than the algorithm with modified action selection strategy. Using greedy algorithm as action selection strategy may bring in local optimal solution issue and cause path planning into a dead end. With simulated annealing strategy, path planning could be spared from the trouble of local optimal solution.

A comparison of four algorithms from their convergence makes it clear that hierarchical reinforcement learning algorithm appears to be more stable than reinforcement learning algorithm and simulated annealing-based action selection strategy may save path planning from being caught in local optimal solution.
4.2.2. Time Consumption

Time consumption could reflect how good the path planning solution is. Therefore, four algorithms are compared in term of total time consumed after every 200 generations of training in a 22*10 grid area during a 5,000-generation training. The comparing results are plotted in Figure 9.

A comparison of SA MAXQ with standard MAXQ algorithm indicates modified SA MAXQ algorithm consumes less time. Thus, SA MAXQ algorithm has higher learning rate. Comparisons of Q-Learning with ε-Q-Learning and standard MAXQ with SA MAXQ reveal greedy strategy to be perplexed by local optimal solution in UAV path planning and to require more time for UAVs to reach the destination. The comparison of Q-Learning with standard MAXQ both of which have chosen greedy strategy implies MAXQ hierarchical structure could speed up optimal path searching through decomposition of task into subtasks in lower scales and bottom-up feedback.

As shown by total time consumption in Figure 9, the aforesaid algorithms all appear to be quite steep in initial stage of training. As the training times grows, this parameter tends to be steady, indicating incomplete learning training in the initial stage causes path found to be caught in a deadlock or troubles. As training times increases, algorithms grow more adapted to the environment, so planned path is more stable, less detours are made, and better learning effect is generated.

![Figure 9. Comparison of four algorithms in total time consumption](image)

4.2.3. Search Steps

Search step could measure effect of path planning algorithms. With 5,000 generations’ iteration as an example, four algorithms are compared in term of search step after every 100 generations in a 22*10 area as shown in Figure 10.

It is clear that all four algorithms demand massive search steps in initial stage of training. As training number rises, the search steps required by four algorithms tend to be stable. However, due to randomness of learning training, an exception may happen amid certain iteration. The search steps of SA MAXQ algorithm are less than that of standard MAXQ algorithm, indicating the former excels the better in learning effect. This may be attributed to the action selection strategy. When search steps grow stable, we can find ε-Q-Learning have almost same search steps as standard MAXQ algorithm while Q-Learning algorithm presents the highest search steps in current scale.

![Figure 10. Comparison of search steps among algorithms](image)
Since Q-Learning appears to be least satisfactory in search steps and ε-Q-Learning and standard MAXQ algorithms that have modified action strategy turn out to have equal search steps, ε-Q-Learning, standard MAXQ and SA-MAXQ algorithms are compared from search path simulation in scenes of different sizes.

Figures 11, 12, and 13 demonstrate the optimal paths generated by ε-Q-Learning, standard MAXQ and SA-MAXQ algorithms for UAV cluster in a 22*10 grid area. The optimal path generated by SA-MAXQ using simulated annealing action selection strategy contains 15 steps, while that generated by standard MAXQ based on greedy strategy contains 19 steps. It is evident SA-MAXQ has generated a better optimal path. Likewise, the optimal path generated by ε-Q-Learning is also made up of 19 steps, suggesting standard MAXQ equals ε-Q-Learning algorithm in current size and both of them are less good than SA-MAXQ.

Figure 11. The optimal path of ε-Q-Learning in a 22*10 area
Figure 12. The optimal path of MAXQ in 22*10 area
Figure 13. The optimal path of SA-MAXQ in 22*10 area

Figure 14. The optimal path by ε-Q-Learning in a 50*45 area
Figure 15. The optimal path by standard MAXQ in a 50*45 area
Figure 16. The optimal path by SA-MAXQ in a 50*45 area

Figure 17. The optimal path by ε-Q-Learning in a 100*70 area
Figure 18. The optimal path by standard MAXQ in a 100*70 area
Figure 19. The optimal path by SA-MAXQ in a 100*70 area

The optimal paths generated by three algorithms in a 50*45 grid area are illustrated in Figures 14, 15, and 16. There are 56 steps in the optimal path by ε-Q-Learning, 54 steps in that by standard MAXQ and 51 steps in that by SA-MAXQ. Thus, in this size, SA-MAXQ with modified action strategy betters standard MAXQ in solving path whereas standard MAXQ betters ε-Q-Learning in the same way.

Figures 17, 18, and 19 demonstrate the optimal paths generated by three algorithms in a 100*70 grid area. There are 99 steps in the optimal path by ε-Q-Learning, 96 steps in that by standard MAXQ and 93 steps in that by SA-MAXQ. In addition to simulation plots against three sizes (Figures 17 to 19), Table 9 lists the max search steps, min search steps and average search steps of three algorithms in several experiments against four sizes.

A comparison of optimal paths generated by three algorithms in four different scenes suggests MAXQ algorithm could generate better path than ε-Q-Learning after experiencing same training sessions. When applied to small scale scenes, two algorithms are comparable in offering a solution. It has testified again that when applied to large-scale scenes, MAXQ may enable UAV to reach destination at a quicker pace as it decreases solving difficulty. Nevertheless, simulated annealing-based algorithm stands out with less steps in the optimal path when being applied to various scenes.
In this study, grid method is employed to form a grid-based model. In addition, grid method is employed in this study to form a 2D simulated scenario in which multi-UAV collaborative path planning is tested. The simulation results indicate that the proposed algorithm and planning model can efficiently solve the problem of multi-UAV path planning and improve the path quality.

5. Conclusion

Aiming at the shortcomings of standard MAXQ algorithms, this paper proposed an improved MAXQ algorithm based on simulated annealing for multi-UAV path planning. Action selection was improved in each training through control of temperature attenuation factor and initial temperature in simulated annealing algorithm so as to yield higher learning rate and optimized path with less time consumed and fewer search steps. In addition, grid method is employed in this study to form 2D simulated scenarios in which multi-UAV collaborative path planning is tested. The simulation results indicate that the proposed algorithm and planning model can efficiently solve the problem of multi-UAV path planning and improve the path quality.

References