ISSN: 2664-9640

Multi-UAV task allocation based on Q-learning algorithm

Qi Dai, Bing He, Junchao Zhao, Ming Li

Rocket Army Engineering University, Xi'an, Shaanxi Province, China

Abstract

Artificial bee colony algorithm [1] is a swarm intelligence algorithm that simulates honeybee honey collecting behavior. It has the advantages of good convergence, few parameters and simple implementation process. However, because each optimization task is isolated, it must be reinitialized when executing a new task, and the past optimization information can not be used well, resulting in too long optimization time of the algorithm, It is difficult to meet the optimization requirements of multiple UAVs in combat scenarios. The classical Q-learning algorithm takes the Markov decision process as the basis of exploratory learning. Each exploratory trial and error can only update one element in the Q-value matrix. When there are many state action pairs in the system, it takes a long time to traverse to obtain the optimal solution. This chapter introduces the idea of bee colony into Q-learning algorithm. When updating the Q-value matrix, multiple bees will explore and learn at the same time. In each iteration, multiple state action pairs will be updated, and the update speed of Q-matrix will be greatly improved, so as to approach the optimal state action pair faster.

Keywords

Q-learning, UAV, swarm.

1. Introduction

In the process of multi-UAV coordinated operations, according to the drone and task coordinates, the task is divided into several[1] sub-tasks according to the principle of shortest distance and resource balance, and then issued[]2 to the corresponding drones, and then each drone The subtasks are executed in sequence according to the issuing order, which forms the original task allocation plan. (It is recommended[3] to draw a framework diagram or flowchart of multi-UAV task allocation here)However, in the application scenario of randomly issuing tasks, the system will randomly issue new tasks. When the drone executes tasks[4] in the order of tasks in the original task allocation plan, it lacks consideration of the important[5] constraint of task completion value. The task completion is low. Therefore, this chapter mainly focuses on the dynamic assignment of multi-UAV tasks based on the Q-learning algorithm in dynamic task scenarios.

2. Problem Description

2.1. Task allocation model

The existing M drones are $U=\{U_1,U_2,\cdots,U_M\}$ a collection of multiple drones. Denote T as a set of tasks, $T=T_1,T_2,\cdots,T_N$, a set of tasks, each target has a different value, and is used to represent the value of the j-th target.

2.2. Restrictions

(1) The total stroke of the fleet is the shortest

An important indicator for the allocation of coordinated attacks on UAV platforms is to "reduce costs as much as possible", in which energy consumption is a key factor. The energy consumed

ISSN: 2664-9640

in flight is mainly related to the flight distance. The shorter the stroke, the less energy is consumed. Therefore, the shortest total fleet travel is as follows.

$$\min D = \sum_{i=1}^{N_v} \sum_{k=1}^{M_i} T_k \tag{2.1}$$

In order to ensure the accuracy and feasibility of the planned results, the amount of fuel combustion/battery energy in the calculation is essential for the initial path planning, so the energy consumption is calculated along the originally planned path.

(2) Maximization of target value

Maximization of target value is also an important indicator platform for UAV mission allocation. In the case of sufficient offensive power, the goal value is maximized. The indicators are as follows:

$$\max V = \sum_{i=1}^{N_V} \sum_{k=1}^{M_i} V_K \tag{2.2}$$

It is the value of the goal.

(3) Threat restraint

Considering the radar detection threat in the combat[6] scenario, it can be simplified and modeled as a hemispherical area. If the coordinates of the center of the sphere of the radar is, and the detection radius is, the threat area can be expressed as:

$$H_{threat} = \left\{ (x, y, z) \mid (x - x_0) + (y - y_0) + (z - z_0) \le r_0^2, z \ge z_0 \right\} \tag{2.3}$$

3. Multi-agent Q learning algorithm based on chain value function

3.1. Conceptualization

The original Q-learning belongs to a single-agent structure, and the learning speed is slow. It usually needs to traverse the entire action space to find the optimal solution. In order to improve the efficiency of original Q-learning in time and space[7], and reduce unnecessary time and space loss, many current researches adopt the method of multi-agent collaboration. However, this method faces a problem that needs to be solved urgently: since each agent has a different reward value after an action, if the value function is updated equally, it will cause the value function to quickly saturate, resulting in a phenomenon of local convergence[8]; if only the optimal value is used Individuals will lose the information diversity of multi-agents, resulting in slower algorithm convergence. For this reason, this article proposes a strategy of "biding ranking", which allocates different value coefficients according to the reward value of the agent. When updating the value function, the reward value contributed by the agent is the product of its actual reward value and the value coefficient. As a preferred method, the value coefficient can adopt an exponentially decayed reward value ranking mode, and it is ensured that the sum of the value coefficients of the agent is 1.

3.2. Q-learning overview algorithm description

Q-learning is based on the discrete-time Markov decision-making process, and realizes online optimization by directly optimizing an iterable calculation of the state-action pair function Q(s, a), so that the expected reward reaches the maximum value. In a nutshell, Q(s,a) refers to the expected value of the reward obtained by the agent performing action a at a certain moment in state s (the environment will get the corresponding reward based on the agent's action feedback) Reward R). So the main idea of the algorithm is to store the Q value by constructing a lookup table of the state s and action a, constructing a lookup table to store the Q value, and then selecting the action that can obtain the greatest benefit as much as possible based on the Q value.

ISSN: 2664-9640

In the topic studied in this article, task allocation can be regarded as the environment, the target and the allocation plan of multiple drones can be regarded as the action of the agent, and the state is the existing resources and constraints of the multiple drones, etc. .

The value function of Q learning satisfies the following formula:

$$Q(s,a)=R(s,s',a)+\gamma\sum_{s'\in S}P(s'\mid s,a)\max_{a\in A}Q(s',a) \tag{2.4}$$

In the formula,, are the current state and the state at the next moment, respectively, are the probability that the state transfers to the state after performing an action, are the reward values obtained from the environment after the state transfers to the state after the action is performed, and are the discount factors, S, A They are state space and action space.

In the iterative calculation of the value function, assuming that the optimal value function of the kth iteration is, and the empirical sample obtained by the agent through this trial learning is, then the Q value of the value function can be iterated according to the following formula:

$$\begin{cases} Q^{k+1}(s_k,a_k)=Q^k(s_k,a_k)+\alpha[R(s_k,s_{k+1},a_k)+ \\ \qquad\qquad \gamma\max_{a'\in A}Q^k(s_{k+1},a')-Q^k(s_k,a_k)] \\ Q^{k+1}(\tilde{s},\tilde{a})=Q^k(\tilde{s},\tilde{a}),\forall(\tilde{s},\tilde{a})\neq(s_k,a_k) \end{cases} \tag{2.5}$$

Where is the learning factor. Initially, the Q value can be given arbitrarily, and is usually set to a smaller value.

In the subject studied in this paper, task allocation can be regarded as the environment, the target and the allocation plan of multi-UAV can be regarded as the action of the agent, and the state can be regarded as the existing resources and constraints of the multi-UAV.

3.3. Chain value function

At present, the state-action space Q of Q learning (that is, the state-action value function Q(s,a)) is mainly realized by the lookup table, and the length of the table is equal to the number of elements of the Cartesian product of S×A. Therefore, when the scale of the problem to be solved becomes larger, the number of elements in the space Q will increase exponentially, which makes iterative calculation difficult and difficult to achieve. For this kind of "curse of dimensionality", the most commonly used methods are to use hierarchical reinforcement learning methods and to generate Q values through the use of neural network methods. The difficulty of the hierarchical learning method is that the decomposition of the optimization task and the hierarchical design and connection are often difficult to determine, which makes the algorithm easy to converge to the local optimal solution; the neural network method is the fusion of common deep learning (Such as DQN), because the learning of neural networks requires a large number of samples, the optimization calculation takes a long time, and when the state-action pair has a complex mapping relationship, a larger network scale is often required to achieve good results. It will increase the difficulty of neural network training.

For this reason, on the optimization problem of discrete finite action space, this paper proposes a method to map high-dimensional state-action space to low-dimensional state-action chain by using chain value function. The discrete finite action space optimization problem can be regarded as a combinatorial optimization problem in essence. The problem has multiple variables, and there are multiple combinations between variables, and each combination constitutes a solution of the problem. The process of the agent's choice of action is essentially a process of choosing a combination between variables. The combined space (that is, the action space) formed by all variables is high-dimensional, and the corresponding state-action space is also a high-dimensional space, but the combined space between the two variables is usually low-dimensional. A low-dimensional state-action space is designed between the two to

ISSN: 2664-9640

correspond to it. All low-dimensional state-action spaces can form a low-dimensional state-action space chain to map the original high-dimensional state-action space.

Specifically, in the m-dimensional action space A (A1, A2, A3,...Am), a Qi is assigned between the variables Ai and Ai+1 to correspond to it, and the variables are related to each other by their respective Qi spaces. The choice of agent action is a Qi-based chain selection process: when the action of the variable Ai is determined, the action is taken as the current state of the next variable, and the action of the next variable is selected according to Qi+1.

In fact, each element of Qi actually corresponds to the action combination composed of the variable Ai-1 and Ai, which is an evaluation of the pros and cons of the adjacent control variable combination. Each element of the Qi space not only represents the pros and cons of the solution selected in the current strategy, but also reflects the closeness of the relationship between the variables. The larger the element value, the closer the relationship between the variables and the higher the evaluation of the corresponding combined action. In the decomposed state-action space, each variable is a small space of lower dimensions, which is convenient for the iterative calculation of the algorithm. The Qi update method after space decomposition is as follows:

$$Q_{k+1}^{i}(s_k^{ij}, a_k^{ij}) = Q_k^{i}(s_k^{ij}, a_k^{ij}) + \alpha[R^{ij}(s_k^{ij}, s_{k+1}^{ij}, a_k^{ij}) + \gamma \max_{a^i \in A_i} Q_k^{i}(s_{k+1}^{ij}, a) - Q_k^{i}(s_k^{ij}, a_k^{ij})] \tag{2.6}$$

In the formula, the superscript i represents the i-th controllable variable, i M, M is the set of controllable variables; the superscript j represents the j-th agent, j N, N is the set of agents; Rij(sk,sk +1, ak) is the reward function value of the environment feedback after the agent performs the action ak from the state sk to sk+1 in the kth iteration; is the learning factor; is the discount factor.

4. Based on artificial ant colony algorithm

Improved action selection strategy

In the Q learning algorithm, the process by which the agent selects actions based on the value of Q is essentially a greedy strategy selection process, as follows:

$$a_{k+1}^{ij} = \begin{cases} \arg\max_{a' \in A^i} Q^i(s_{k+1}^{ij}, a'), & \text{if } q < \varepsilon \\ a_s, & \text{otherwise} \end{cases} \tag{2.7}$$

In the formula, ε is the greedy factor; q is a random number between 0 and 1; it is the action selected by the probability matrix Pi in the global scope. Among them, the element of Pi corresponds to Qi, and the greater the Q value, the greater the probability value corresponding to the action, and thus the greater the probability of being selected.

On the one hand, the greedy strategy can make full use of the experience learned by the agent, but on the other hand, it is also prone to local convergence, or "overfitting". In the training process of neural network, adding samples and noise is an important method to solve the over-fitting sum. This work is inspired by the bee colony algorithm, and uses the method of participating in sampling through multiple harvesting bees, which largely avoids the problem of reinforcement learning entering the local optimum.

Artificial Bee Colony (ABC) is a novel global optimization algorithm based on swarm intelligence proposed by Karaboga in 2005. Its intuitive background comes from the honey-collecting behavior of bees. Bees carry out different activities according to their division of labor, and realize the sharing and communication of bee colony information, so as to find the optimal solution to the problem. The standard bee colony algorithm divides the artificial bee colony into 3 categories by simulating the honey-collecting mechanism of actual bees: collecting bees, observing bees and reconnaissance bees. The goal of the entire colony is to find the nectar

ISSN: 2664-9640

source with the largest amount of nectar. In the standard ABC algorithm, the honey bee uses the previous nectar source information to find a new nectar source and shares the nectar source information with the observing bee; the observing bee waits in the hive and searches for a new nectar source based on the information shared by the bee; the task of the detective bee is to find A new valuable source of nectar, they randomly look for nectar sources near the hive.

The nectar source actually corresponds to the solution in the display problem, the behavior patterns of collecting bees, observation bees, and scout bees, corresponding to the evaluation solution in the algorithm, updating the statistical results, and exploring the solution.

Since the action category has been determined, the nectar search function is omitted in this work, and the evaluation of nectar quality corresponds to the final reward for the action. Specifically, the goal of the entire bee colony is to find the nectar source with the largest amount of nectar, that is, the most valuable task scheduling plan. The third step is to select an action and get the corresponding reward, which can be regarded as a number of bees using the previous information, that is, the probability of action to pick honey, and the observation bees give the value of the task scheduling plan and rank it. The best dispatch plan found was taken down by the recorder. Finally, the corresponding state values and probabilities of these scheduling schemes are updated. Because the solution space is discrete during the specific update, this work uses the Bellman formula of reinforcement learning, and updates the action probabilities according to the action value. The formula for calculating the probability of selecting a nectar source by the traditional bee colony algorithm is:

$$p_i = \frac{fit_i}{\sum_{j=1}^{SN} fit_j} \tag{2.8}$$

Where represents the fitness of a nectar source, and SN is the number of collecting bees and observing bees. This work takes action value as the basis for calculation. Therefore, for the first action selection, the updated probability calculation formula is:,

$$p_i^0 = \frac{Q_i^0}{\sum_{j=1}^{N} Q_j^0} \tag{2.9}$$

Among them, represents the value of the first action selection i, and N represents the number of optional actions; for the t-th action selection, the updated probability calculation formula is:

$$p_{k,i}^t = \frac{Q_{k,i}^t}{\sum_{j=1}^{N} Q_{k,j}^t} \tag{2.10}$$

This formula calculates the probability of i selected for this action when the last action was k.

Specifically, the task assignment and task sequencing of the two dimensions are based on the above two formulas to complete the work of updating the action probability of each step.

5. Multi-UAV task allocation problem solving problem solving

The research content of this article needs to solve two main problems: (1) the allocation of drones, that is, which drone will perform the target task; (2) the order in which the drone executes the target task.

In the UAV distribution, this work has constructed a continuous selection of UAV schemes, through the selection of UAV distribution algorithm, the UAV distribution sequence is finally formed; and the task sorting is based on the continuous selection of task schemes through the task sorting algorithm. Finally get the target task sequence. Combine the two sequences to get each UAV and its corresponding target mission. In this way, the task assignment action and the task sequencing action are separately constructed as independent path planning problems. This work uses the A-star algorithm to gradually search to obtain the final solution, that is, the

ISSN: 2664-9640

number of targets is used as the number of searches, and the next action is determined each time based on the local action of the previous search.

The A-star algorithm is one of the most effective direct search methods for solving the shortest path in a static road network. Its formula is expressed as:

$$f(n)=g(n)+h(n) \tag{2.11}$$

Where f(n) is the evaluation function from the initial point to the target point via node n, g(n) is the actual cost from the initial node to the n node in the state space, and h(n) is the best value from n to the target node The estimated cost of the path. In the A* algorithm to ensure that the shortest path (the optimal solution) is found, the key lies in the selection of the evaluation function h(n). The A-star algorithm generally starts from the initial node when exploring the path. For each step of the search, the current node is used as the base point, and its neighboring nodes are scanned, and the distance h(n) between these nodes and the target point is estimated, and then a f(n) The largest direction is used as the new base point. Since the value of the base point g(n) is fixed at any time of selection, this problem can actually be transformed into the need to estimate h((n)) each time a selection is made, and choose h((n) n) the smallest direction.

However, under the current problem, we don't know what elements the initial node and target node of the task correspond to. On the contrary, what we know and what we know is the number of choices that will be made. Therefore, in this work, only the framework in which the A-star algorithm selects an action according to the estimation is retained by default. Specifically, according to the A-star algorithm, when the first target point is matched with the drone, the drone and the target point will be randomly performed according to the recorded probabilities of each action Px[0], Py[0] s Choice. Each subsequent selection will be based only on the previous action, not the entire historical action or direct selection of the action. The probability of matching at the t-th target point is Px[0][x_{t-1}], Py[0][y_{t-1}], where x_{t-1} and y_{t-1} represent the drone and target point selected when the t-th target point is matched. Finally, after several action selections, the action trajectory in the two dimensions of the distribution of the drone and the sequence in which the drone executes the target task is constructed, and finally combined into a solution.

5.1. State-action value function and reward function design

In the scheduling process of drones, it is mainly necessary to solve the two problems of task allocation and task execution order of drones. Therefore, the allocation variable X1 and the sorting variable X2 are defined. X=(X1,X2) represents a scheduling plan of the problem. Each plan is regarded as an action sequence of the intelligence, and the trajectory set of the agent constitutes the solution of the problem.

The prerequisite for finding the best is to be able to evaluate the current solution. This work proposes three evaluation indicators for the solution. The first is the total distance traveled by the drone. The distance traveled by a drone corresponds to the sum of the distances of two adjacent targets in its target trajectory. The total distance traveled by the drone is obtained by summing the distances of all drones. D(X). The second indicator is the total reward obtained by drones. The reward for each drone is the sum of its target point rewards, R(X). The last is the sum of the penalties of the drones. The penalty for each drone is the part P(X) that exceeds its driving ability in the distance that it is expected to travel.

The final objective function is defined as

$$fit(X) = 0.01 * D(X) + 100/R(X) + P(X) \tag{2.12}$$

A good solution should make the objective function value as small as possible. Therefore, since the goal of reinforcement learning is to maximize the reward, the reciprocal of fit will be used as the reward for scheduling-because the goal of reinforcement learning is to maximize the reward. At the same time, this work also adds a reward coefficient k to this reward. That is to

ISSN: 2664-9640

say, for a solution, which is an action sequence X of reinforcement learning, the corresponding reward is (k/fit(X), k/fit(X), k/fit(X), ...).

The value of this work is based on the estimation of the action value of the Bellman equation, namely

$$Q(s_t, a_t) = \frac{k}{fit(X)} + \gamma \times \max[Q(s_{t+1}, A_1), Q(s_{t+1}, A_2), Q(s_{t+1}, A_3), ...] \tag{2.13}$$

When updating, make the learning rate α

$$Q(s_t, a_t) = (1-\alpha) \times Q(s_t, a_t) + \alpha \times (\frac{k}{fit(X)} + \gamma \times \max[Q(s_{t+1}, A_1), Q(s_{t+1}, A_2), Q(s_{t+1}, a_{t3}), ...]) \tag{2.14}$$

6. Q-learning algorithm simulation experiment and simulation analysis

In order to test the effect of the algorithm proposed in this chapter on cluster task allocation, this section designs simulation experiments for Q-learning algorithm, artificial bee colony algorithm and genetic algorithm respectively to test the algorithm of Q-learning algorithm in handling UAV cluster task allocation problems. performance.

6.1. Simulation condition setting

Assuming that the space grid map model used in this combat area is, and the combat scenario is set to a homogeneous UAV cluster to perform related combat tasks against multiple groups of stationary ground targets. The UAV cluster participating in this battle and the combat target are in a three-dimensional grid space, and the combat target is in a static state. The threat source in the combat area considers the threat of obstacles, and the threat source location is known. The UAV cluster will take measures during the execution of the mission. Avoidance strategy.

6.2. Simulation result analysis

1. Algorithm sensitivity analysis

Reward coefficient (compare the graphs of objective function under different reward coefficients)

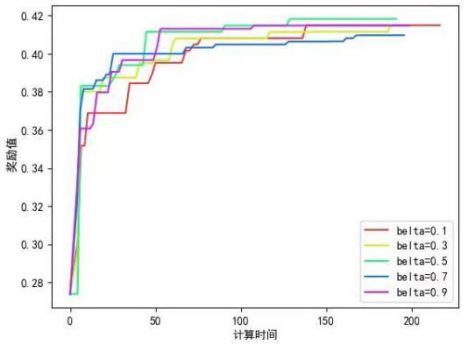

Figure1 Simulation diagram of different reward coefficients

2. Comparison of different algorithms

Convergence curve comparison of objective function (average convergence curve of 10 calculations)

ISSN: 2664-9640

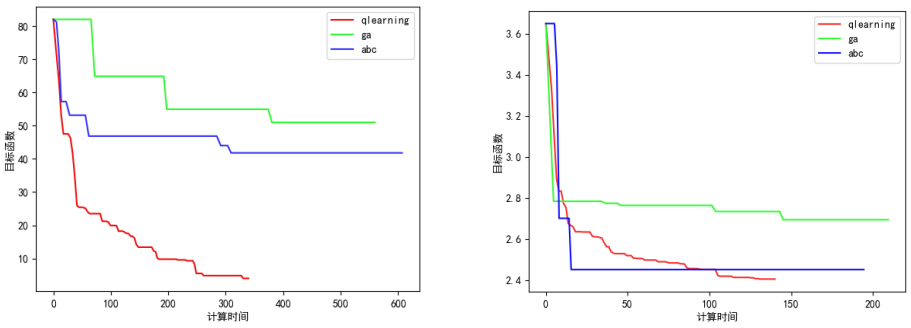

Figure 2 Convergence curve of objective function

3 algorithm effectiveness analysis

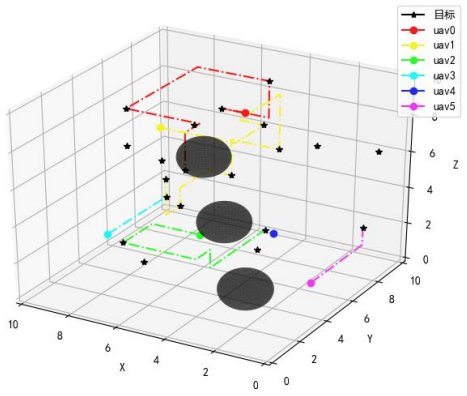

Figure3Q-learning algorithm task allocation effect diagram

7. Conclusion

This chapter establishes a multi UAV task allocation environment with known threat areas, and studies a multi UAV task allocation method based on improved Q-learning, which introduces the idea of bee colony algorithm into the action selection strategy of Q-learning algorithm. Finally, the improved Q-learning algorithm is much better than the traditional intelligent algorithm in convergence speed and optimization path.

References

[1] JIA W T, LI C T. Trajectory optimization of unmanned aerial vehicle and research on its following technolo-gy[J]. Machine Building & Automation, 2020, 49(6):156-161.
[2] LI H, GUO S L, ZHOU Y. Dynamic diversion planning combining dynamic risk map and improved A* algo-rithm[J]. Aeronautical Science & Technology, 2021, 32(5):61-71.
[3] GAO S, AI J L, WANG Z H. Mixed population RRT algorithm for UAV path planning[J]. Systems Engineering and Electronics, 2020, 42(1): 101-107.
[4] ZHANG W. SONG K, RONG X. Coarse-to-Fine UAV target tracking with deep reinforcement learning[J]. IEEE Transactions on Automation Science and Engineering,2019, 16(4): 1522-1530.
[5] LI C, CAO L. CHEN X, et al. Cloud reasoning mod-el-based exploration for deep reinforcement learning[J]. Journal of Electronics & Information Technology.2018, 40(1):244-248.
[6] FENG S, SHU H, XIE B Q. 3D Environment path plan-ning based on improved deep reinforcement learning[J].Computer Applications and Software, 2021, 38(1):250-255.
[7] RODRIGUEZ R, ALEJANDRO S, CARLOS B, et al. A deep reinforcement learning strategy for UAV autono-mous landing on a moving platform[J]. Journal of Intelli-gent & Robotic Systems, 2019, 93(1): 351-366.
[8] ZHANG Y Z, XU J L, YAO K J, et al. Pursuit missions for UAV swarms based on DDPG algorithm[J]. Hang-kongXuebao, 2020, 41(10): 314-326.

ISSN: 2664-9640

Introduction to the first author: Dai Qi (1996—), male, master, 2758043582@qq.com, research direction: multi-UAV task allocation