Smart Anti-jamming Mobile Communication for Cloud and Edge-Aided UAV Network

Zhiwei Li[1], Yu Lu[1]*, Zengguang Wang[2], Wenxin Qiao[1], and Donghao Zhao[1]
[1] Shijiazhuang Campus, Army Engineering University of PLA
Shijiazhuang, 050003, China
[e-mail: arhqs@126.com, luyubj@163.com, qiaowenxin1992@foxmail.com, zhaodonghao1@sina.com]
[2] National Defense University
Shijiazhuang, 050000, China
[e-mail: zengguang_wang@126.com]
*Corresponding author: Yu Lu

Received July 29, 2020; revised November 9, 2020; accepted December 6, 2020; published December 31, 2020

Abstract

The Unmanned Aerial Vehicles (UAV) networks consisting of low-cost UAVs are very vulnerable to smart jammers that can choose their jamming policies based on the ongoing communication policies accordingly. In this article, we propose a novel cloud and edge-aided mobile communication scheme for low-cost UAV network against smart jamming. The challenge of this problem is to design a communication scheme that not only meets the requirements of defending against smart jamming attack, but also can be deployed on low-cost UAV platforms. In addition, related studies neglect the problem of decision-making algorithm failure caused by intermittent ground-to-air communication. In this scheme, we use the policy network deployed on the cloud and edge servers to generate an emergency policy tables, and regularly update the generated policy table to the UAVs to solve the decision-making problem when communications are interrupted. In the operation of this communication scheme, UAVs need to offload massive computing tasks to the cloud or the edge servers. In order to prevent these computing tasks from being offloaded to a single computing resource, we deployed a lightweight game algorithm to ensure that the three types of computing resources, namely local, edge and cloud, can maximize their effectiveness. The simulation results show that our communication scheme has only a small decrease in the SINR of UAVs network in the case of momentary communication interruption, and the SINR performance of our algorithm is higher than that of the original Q-learning algorithm.

Keywords: Anti-Jamming, A3C, Edge Computing, IoT, UAV Network

This research was supported by the National Natural Science Foundation of China under Grant 62071483.

http://doi.org/10.3837/tiis.2020.12.004 **ISSN : 1976-727**

1. Introduction

Low-cost Unmanned Aerial Vehicles (UAV) swarm have broad application prospects in military and civilian fields such as agriculture, logistics, reconnaissance, and communication relay. Due to the open wireless communication environment, UAVs are vulnerable to spoofing, jamming and eavesdropping attacks. Information encryption can effectively defend against spoofing attacks. Artificial noise also has an obvious defense effect against eavesdropping attacks. However, the implementation cost of jamming attacks is extremely low and can be launched without relying on any prior information. This makes jamming attacks the most common security threat to UAV networks.

By applying programmable radio devices such as universal software radio peripherals (USRP), smart jammer can observe the ongoing communication states between UAVs and then choose the optimal jamming strategies accordingly. Compared with the traditional jamming attack, any single mode anti-jamming strategy such as frequency hopping spread spectrum (FHSS) or direct sequence spread spectrum (DSSS) can hardly achieve the expected anti-jamming effect. Therefore, the communication strategies between UAVs must also be smart and programmable. Game theory can well model this kind of decision-making confrontation problem. However, the solution of traditional game problems requires knowledge of a large amount of information such as UAV and jammer strategies, channel models, and revenue models. This information is not only highly confidential, but also dynamically changing. In other words, in a highly dynamic environment, the game model cannot be solved.

The essence of the smart anti-jamming problem of UAV networks is how to make intelligent decisions. With the development of reinforcement learning, especially deep reinforcement learning algorithms, it has brought dawn to the solution of this problem. Although the smart anti-jamming technology based on deep reinforcement learning algorithms continues to make progress, it can only be applied to high-performance single UAV, but not to UAV swarms. This is because UAV swarms are composed of low-cost UAVs, and the deployment of these algorithms directly will face three serious difficulties. First, deep reinforcement learning algorithms cannot perform the model training process on mobile devices. Current deep learning frameworks, such as TensorFlow Lite, can deploy trained deep models to mobile devices, but deep models cannot be trained in real time on mobile devices. In other words, the parameters of the model cannot be updated as the jamming strategy changes. Second, although tabular reinforcement learning algorithms can be deployed on low-cost UAV platforms, the state space and action space of the anti-jamming decision problem are continuous, which will cause dimensional disaster problems. Simple discretization will make it difficult for the tabular models to converge. Finally, there is a solution that seems to be feasible. The UAV platform offloads computing tasks to the cloud or the edge, and UAV receives the results from cloud computing or edge computing and then makes decisions. The problem is that due to the existence of jamming sources, the service quality of the UAV network is extremely unstable, and it is easy to appear that computing tasks cannot be offloaded or calculation results cannot be received.

In this work, we propose an A3C teaches Q-learning (A3CTQ) algorithm based on cloud computing and edge computing to solve the above problems. The main idea of the algorithm is to transfer the intelligence contained in cloud computing power to the UAV platform to help UAVs make reasonable decisions when offline. Specifically, the cloud computing center

gathers the state data collected by all the UAV platforms, and feeds these data to the A3C algorithm for parallel calculation with nearly unlimited computing power to obtain a cloud policy network. Similarly, the edge servers collect regional state data and feed these data to the DDPG algorithm to obtain the edge policy network. The cloud policy network playing the role of expert trains a Q-table and a policy network through an inverse reinforcement learning algorithm. The cloud model helps to update models in UAV platforms and edge servers. This parameter update is the process in which cloud intelligence flows to the UAV platforms and the edge servers. When offline, UAV uses Q-tables updated by the cloud to complete decision-making tasks. When the network is connected, the UAV platform will send all the state data collected during the offline period to the cloud. As long as the network disconnection is not too long, the decision-making ability of the UAV platform will not be significantly reduced.

The main contributions of this paper are summarized as follows:

- We establish a cloud and edge-aided communication scheme to solve the problem of anti-jamming communication of UAV networks to reduce the performance loss caused by intermittent network interruptions.
- We build a game model to solve the computational offloading problem of UAV networks facing intermittent disconnection of communication link.
- We design an A3CTQ algorithm that transfers cloud intelligence to the UAV platform to solve the problem of continuous anti-jamming control of UAV networks with intermittent communication link interruption.

The rest of this paper is organized as follows. We review related work in Section 2 and present the system model in Section 3. We propose a cloud and edge-aided UAV network communication scheme and introduce the details of the A3CTQ algorithm in Section 4. We build a game model to implement decentralized control of computational offloading in Section 5. We provide simulation results in Section 6 and conclude this work in Section 7.

2. Related Work

Smart jamming attack which has the ability of strategy learning makes the anti-jamming effect of relying only on the physical layer technology very unsatisfactory. The problem of smart jamming was first proposed in [1], and game theory proved to be a powerful theory to solve the problem of smart jamming. The authors in [2] transferred this problem of using game theory to model anti-smart jamming to the field of cognitive radio for the first time. Since then, almost all the anti-smart jamming related researches are based on game theory to establish the decision models.

Early studies were based on the perfect observation assumptions, and some excellent works [3-6] based on the imperfect observation assumptions to study anti-jamming problems appeared after that. For example, with considersing the situation where users do not know the type of the jammers, a Bayesian game was used to model the anti-jamming problem[3, 4]. A Bayesian Stackelberg game was used to model the interactions between user and smart jammer[5]. The authors in [6] has formulated the jamming game with imperfect information, i.e., the jammer's bounded rationality and inaccurate observation of the user's action.

The incomplete information game makes the modeling of this problem more realistic, but the solution of the model also encounters the problem of incomplete information. Most of the work introduced before is solved by planning theory and optimization theory. But a lot of

information needed to solve the problem is difficult to obtain, such as channel gain, the position of the jammer, maneuvering mode and jamming power allocation strategy.

Reinforcement learning algorithms, especially model-free reinforcement learning algorithms, have brought dawn to the solution of the this tough problem. The model-free reinforcement learning algorithm continuously collects the interactive data generated by interacting with the environment, and learns anti-interference strategies from these data. Therefore, the algorithm does not require any prior knowledge. In [7], the authors were the first to use Q-learning algorithm to solve the smart jamming problem in cognitive radio network. The problem of multi-agent anti-intelligence interference decision-making has gradually become the focus of researchers. In [8], the authors used a multi-agent Q-learning algorithm called femto-based distributed and sub-carrier-based distributed power controls using Q-learning (FBDPC-Q) to deal with the aggregate macrocell and femtocell capacities. Considering the impact of the channel estimation error, a Q-learning based power control algorithm using non-cooperative game theory was proposed to suppress the joint smart jamming attack[9]. In [10], the authors used UAVs to relay the messages and improve the communication performance of VANETs with a Q-learning based scheme. The application areas of Q-learning algorithms continue to expand. In [11], the authors solve the anti-jamming problem of UAV radar network based on the double greedy Q-learning algorithm. The Q-learning algorithm associated with an onedevice federated jamming detection mechanism [12] played an important role in the defense against smart jamming attack in FANET.

However, the shortcomings of the table-driven Q-learning algorithm have gradually emerged. In the multi-agent system anti-intelligence interference problem, the state space of the problem to be solved is very large. Consider a typical communication system composed of multiple UAVs, where each UAV's receiver has multiple channels and the UAVs can perform autonomous maneuvering. The high dimensionality of the state space of the system can easily cause dimensional disasters, making the Q-learning algorithm unable to work. The combination of deep learning technology and reinforcement learning technology can perfectly solve this problem. The authors in [13] proposed a hotbooting deep Q-network based 2-D mobile communication scheme to address the smart jamming problem. The algorithm is deployed in a robot that can move on the ground, not in a UAV platform. In [14], a deep Q-learning-based UAV power allocation strategy combines Q-learning and deep learning was proposed, but this research only studied the scenario of one UAV rather than UAV network. The authors in [15] extended the DQN algorithm to the scenario of incomplete information game, and proposed a deep recurrent Q-networks (DRQN) in the three-dimension space to obtain the optimal anti-jamming strategy with incomplete channel state information(CSI).

It is worth noting that the research on anti-smart jamming is mainly concentrated in wireless networks, and there is almost no related research on UAV networks, except for UAV-aided networks. In [16], the authors systematically reviewed the applications and challenges of UAV-aided networks. In [17], the authors focused on the anti-jamming techniques in VANETs and proposed a hotbooting policy hill climbing (PHC)-based strategy to deal with this kind of jamming. Beak et al. [18] developed a future UAV-aided tactical data link to improve the reliability of military communication. Further more, Xiao et al. [10] combined the PHC frame and Q-learning algorithm to solve the problem of smart jamming in UAV-aided VANETs without knowing the VANET model and jamming model. The advantages of deep learning are also reflected in UAV-aided networks, on the basis of previous studies, in [19] the authors combined the reinforcement learning algorithm with deep learning techniques to address the issue of dimensional curses.

Later researchers have made continuous progress in theoretical research on anti-smart jamming attacks, but they have not considered enough on how to deploy related algorithms, especially deploted on low cost UAV swarm. We know that low-cost UAV platforms are energy-constrained and cannot deploy deep reinforcement learning algorithms. Zhou et al. [20] proposed a mobile edge computing (MEC) system to support the deployment of decision algorithm. However, the problem of deep reinforcement learning algorithm failure caused by intermittent communication interruption has not been paid attention to by researchers.

By reviewing the above works, we draw the following conclusions. The game theory model is very suitable for solving the problem of resource optimization in wireless communication in a decentralized scenario. However, the overly complicated calculation method of the reward function makes the previous game models cannot directly applicable to low-cost drone platforms. The lightweight computing offloading algorithm is more suitable for UAV swarms. At present, there is almost no related work on smart anti-jamming of UAV swarms, especially when network communication is intermittent.

3. System model

The functional flow diagram of the proposed A3CTQ communication scheme consists of five parts. The first part is data collection. Each UAV collects information such as channel gain, signal to inference plus noise (SINR) value of each receiver, location of jamming sources from the environment or smart jammers. The second part is computing task dispatch. A game theory algorithm is running to dispatch the current computing task to the cloud, the edge servers of local. The third part is decision-making. The cloud strategy is calculated by the A3C algorithm, the edge strategy is calculated by DDPG algorithm and the local strategy is calculated by Q-learning algorithm. And the priority of the strategy decreases sequentially, that is, the cloud strategy has the highest priority and the local strategy has the lowest priority. The fourth part is implementation. The UAV starts a timer after dispatching the calculation task. When the timer expires, the strategy with the highest priority received is selected as the UAV strategy and implemented. The fifth and the final part is update. We use the cloud A3C algorithm as an expert strategy, and use reverse reinforcement learning algorithms to guide offline training of DDPG and Q-learning algorithms to continuously improve offline decision-making capabilities.

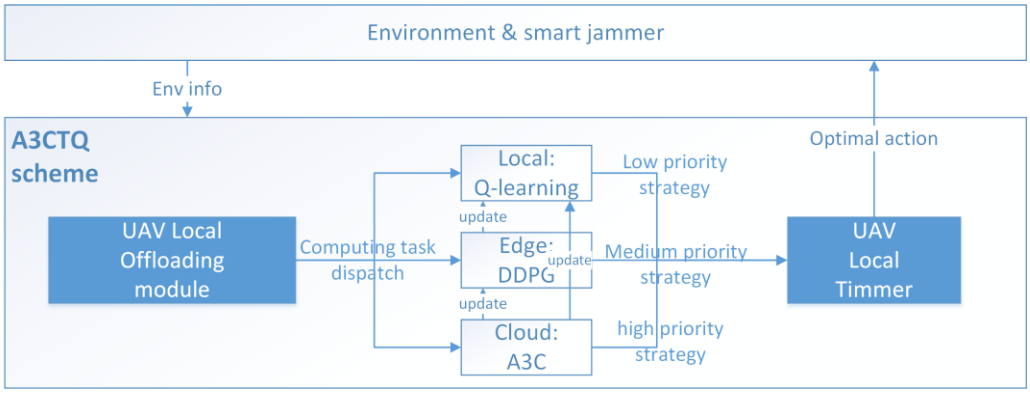

Fig. 1. The function description diagram of A3CTQ scheme

3.1. Communication model

As shown in **Fig. 2**, the communication model consists of three parts. The first part is UAV network, and its network topology is hierarchical mesh network. The backbone mesh network consists of backbone UAVs. The backbone UAVs have strong communication capabilities and a long distance for sending and receiving information. Each backbone UAV establishes a homeland area around it, and there are several small and low-cost mission UAVs in the area. Cross-domain communication of mission UAVs requires relay services from backbone UAVs. GPS is responsible for providing timing and positioning services for UAVs. Jamming UAVs can approach backbone UAVs and mission UAVs and launch smart jamming attack based on cognitive radio technology.

The second part is the ground station network. The ground stations maneuver within the communication range of the backbone UAV, and use data link technology to maintain communication with the backbone UAV. Ground stations maintain communication through microwave relays. Since the ground station is far from the mission area, it is assumed that the communication link between the ground stations cannot be jammed. The working area of mission UAVs is supported by base stations.

The third part is the cloud computing center. The UAV network communicates with the cloud computing center through the base station with 4G or 5G cellular communication technology. The base station is connected to the cloud computing center through an optical fiber link. The base station is in an urban environment with a complex electromagnetic environment, and it is difficult to effectively control jamming sources in the city. Therefore, it is assumed that the air interface links between the base stations and the UAVs will be interfered with and intermittently interrupted. The links between the base stations are fiber-optic and are not subject to interference.

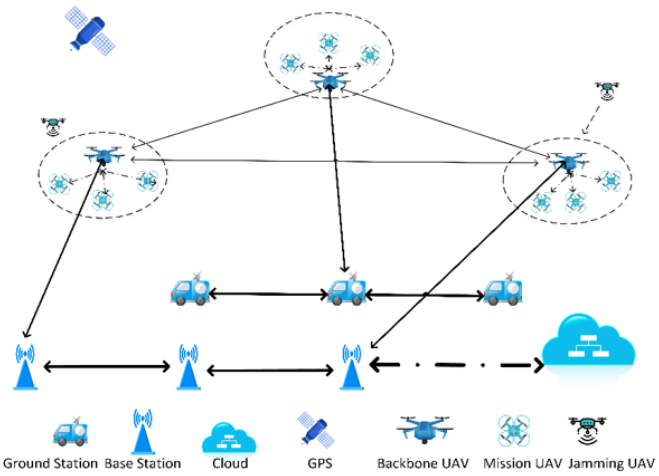

Fig. 2. Topology of UAV network under the support of cloud computing center and edge servers

3.2 Computation model

The computing model studied in this paper includes three types of computing resources, namely local computing resources on UAVs, edge computing resources on ground stations, and cloud computing resources connected to remote cloud computing center. For the first, local computing power is provided by embedded chips on UAVs, such as the ARM Cortex A72 processor for the Raspberry Pi. This computing power has almost no delay, but due to the

limitations of the UAV platform, high-intensity calculations cannot be performed. The second type of computing power is the computing power on the ground station. The on-board platform can provide strong GPU computing power, and it is directly connected to the UAV through the data link, and the delay is small. The third kind of computing power is the remote cloud computing center. The cloud computing center can provide almost unlimited computing power. However, the air interface link between the UAV and the cloud computing center can be easily jammed. The ground link to the cloud computing center needs to be forwarded through multi-hop routing, and the transmission jitter and delay of the data packets are large. The features of the three computing resources are shown in the **Table 1**.

Table 1. Features of three computing resource

Computing Resources	Advantages	Disadvantages
Local UAV	No latency	Poor computing ability;High energy consumption
Ground Station as Edge Server	Saving the energy of UAV; Low latency; Strong computing ability; Node location can be redeployed	Computing power be overloaded
Remote Cloud Computing Center	Saving the energy of UAV; Extremely strong computing ability;	High latency; High jitter; Access points are fixed and vulnerable

The data transmitted from the UAV platform to the ground station and the cloud computing center are real-time environmental state data collected by the UAV sensors. The ground station and cloud computing center receive the state data and feed the data to the A3CTQ distributed deep reinforcement learning algorithm established in Section 4 for intensive calculations. The calculation results are sent back to the UAV platform.

The computing tasks generated by the UAV i are modeled as tuples $\left(C_i^{Local}, C_i^{Edge}, C_i^{Cloud}, D_{state}, D_{par}\right)$. Different algorithms are used to execute the same computing task i on different computing platforms. The three notations $C_i^{Local}, C_i^{Edge}, C_i^{Cloud}$ indicate the number of CPU cycles required for the task i at the local, edge, and cloud computing center. Since the format of the state data to be sent by each UAV is strictly specified, the size of the data is expressed by a constant D_{state}. The state data includes the measurement time, the current position of the UAV, the current speed of the UAV, the transmission power of each channel, and the Signal to Interference plus Noise Ratio (SINR) of each channel. After receiving the state data, the cloud computing center and edge servers run the A3CTQ algorithm to get the updated model parameters. The model parameters include two parts: the parameters for the Q-table of Q-learning algorithm deployed in the UAV platform and the parameters for the edge policy network. Therefore, the size of the model parameters D_{par} is also a constant. Compared with resource-constrained UAV platforms, the computing resources and power resources of ground stations and cloud computing centers are sufficient, so the calculation cost after data unloading is ignored, and the energy consumption cost of data transmission to the UAV platform is negligible.

3.2.1 Local Execution

The time required to calculate locally on the UAV platform is in [20].

$$T_{Local} = C_i^{Local} / F^{Local}, \tag{1}$$

where F^{Local} is the CPU frequency of the local UAV platform. Because computing tasks are offloaded locally, there is no data transfer delay.The energy consumption of the computing task i being offloaded to the local is

$$E_{Local} = C_i^{Local} e^{Local}, \tag{2}$$

where e^{Local} is the energy consumed by each CPU frequency of the UAV platform. The local calculation does not consider the energy consumption of data transmission.

3.2.2 Offloading to Edge

The time required to calculate on edge server is [20].

$$T_{Edge} = C_i^{Edge}\left(1+Q^{Edge}\right)/F^{Edge} + \left(D_{state} + D_{par}\right)/R_{Edge}, \tag{3}$$

where F^{Edge} is the CPU frequency of the edge platform. C_i^{Edge} / F^{Edge} is the edge computing time of the task i. Q^{Edge} is the length of the cache queue for unfinished tasks sent by other UAVs. R_{Edge} is data transfer rate from UAV platform to edge. The total time to receive and send data is $\left(D_{state} + D_{par}\right)/R_{Edge}$ .The energy consumption of the computing task i being offloaded to the edge is

$$E_{Edge} = D_{state} e^{Edge}, \tag{4}$$

where e^{Edge} is the energy consumption of UAV sending each unit of data to the edge. We ignore the energy consumption of receiving data.

3.2.3 Offloading to Cloud

Similar to the case of offloading to the edge, the time required to calculate on cloud is

$$T_{Cloud} = C_i^{Cloud}\left(1+Q^{Cloud}\right)/F^{Cloud} + \left(D_{state} + D_{par}\right)/R_{Cloud}, \tag{5}$$

The energy consumption of the computing task i being offloaded to the edge is

$$E_{Cloud} = D_{state} e^{Cloud}. \tag{6}$$

Due to the limitation of the power resources of the UAV platform, when the power of the UAV is insufficient, the UAV needs to return to the base, and another UAV will take over the UAV

to continue the task. When the number of UAV in a swarm is large, it can be approximated that the occurrence of UAV mission succession events is uniform, so that the cost of energy consumption has a linear correlation with the cost of time consumption. From the analysis, we can establish the following payment function:

$$Z_{cost} = \alpha T + \beta E, \tag{7}$$

where the weight parameters α, β are used to adjust the ratio of calculation cost and energy consumption cost.

3.3 Channel and jamming model

According to [10], the channel model of air-to-air and ground-to-air links for UAVs can be modeled as a log-normal shadowing model with constant channel power gains. The path loss denoted by PL can be modeled by

$$PL(\mathrm{dB}) = \nu(\mathrm{dB}) + 10\rho \lg\left(\frac{d}{d_0}\right),\ d > d_0, \tag{8}$$

where ν is antenna gain constant, and d_0 is the reference distance, and ρ is the pass loss exponent at reference distance.

It can be seen from the communication model in **Fig. 1** that the jammer can interfere with three kinds of links, one is the link between the UAVs, one is the link from the UAV to the ground station, and one is UAV-to-tower link. Among them, the first link is an air-to-air link, and the latter two are air-to-ground links. Among them, the air-to-air link belongs to free space propagation, and we take its path loss exponent as 2, and the ground-to-air link is affected by multipath effect, and we take its path loss exponent as 4.

3.4 Utility model

The utility model comprehensively considers the two factors of communication quality and data offloading quality. This model is the final evaluation model in this paper. The network utility has a positive correlation with the expected SINR of each channel of the UAV platform. However, we believe that the network utility and the offloading cost $Z_{\cos t}$ in Eq. (7) are not simply negative correlations.

$Z_{\cos t}$ can effectively prevent computing tasks from being offloaded to a single computing node. However, it is impossible to describe the effect of offloading on improving the anti-jamming decision. Therefore, we define a delay sensitivity function $f_{latency}(.)$ to characterize this relationship. This function is defined by

$$f_{latency}(t, t_1, t_2) = \begin{cases} 1 & t < t_1 \\ \dfrac{t - t_1}{t_2 - t_1} & t_1 \le t < t_2 \\ 0 & t \ge t_2 \end{cases}, \tag{9}$$

where t_1, t_2 are two time-value thresholds. Specifically, t_1 is called a health threshold. If the UAV can receive the calculation result from the cloud or the edge within t_1 time, there is no value loss. Specifically, after UAV offloads computing tasks to the cloud or edge server, it starts to run local decision-making algorithms. We assume that the expected time required for the local decision algorithm is t_1. Then, when the time for receiving the returned remote calculation result is less than t_1, no additional time delay will be generated, and therefore, no additional time delay will be generated by the calculation offloading. t_2 is called the failure threshold. If the UAV has not received the calculation results until time t_2, it is considered that this offloading has no value. We know that smart jamming attacks can periodically change the attack strategy. We assume that the average time for smart jamming strategy update is t2. If the time required for offloading is greater than t2, it is meaningless when the remote calculation result is returned to the UAV. If the receiving time is between the two, the benefit is considered to be linearly discounted. The value-optimized network utility model is given by

$$Utility = SINR / \left(f_{latency}\left(Z_{cost}, t_1, t_2\right) Z_{cost} \right). \tag{10}$$

When the number of UAVs in the network is not too large, energy cost cannot be simply considered to be proportional to time cost, so it cannot be approximated by parameter β. In this case, the network utility function is modeled as

$$Utility_k = f_{latency}\left(T + T_{battery}^{UAV} \max\left(\mathrm{sgn}\left(E - e_{threshold}\right), 0\right), t_1, t_2\right) SINR \tag{11}$$

where $e_{theroshold}$ is threshold level that requires battery replacement, and E is the remaining power of UAV.

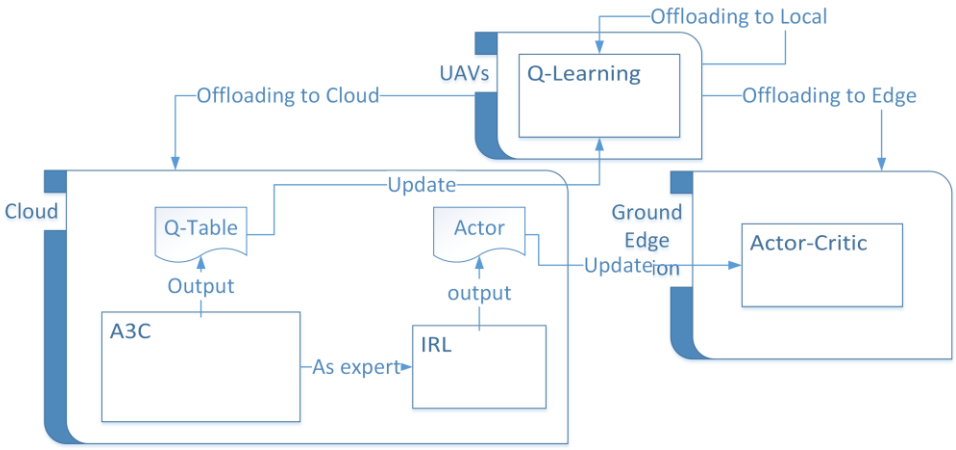

Fig. 3. Cloud and edge-aided UAVs network communication scheme

4. Cloud and edge-aided UAV network communication scheme

As shown in **Fig. 3**, there are three types of communication entities in the communication scheme, namely UAV platform, edge server and cloud computing center, in which the UAV platform is responsible for environment perception, and the edge servers and cloud computing center are responsible for computing power support. In other words, the UAV platform is the source of environmental awareness, and the cloud computing center is the upper limit of computing power. The capabilities of these two entities are completely asymmetric, and the edge servers play a balancing role between them.

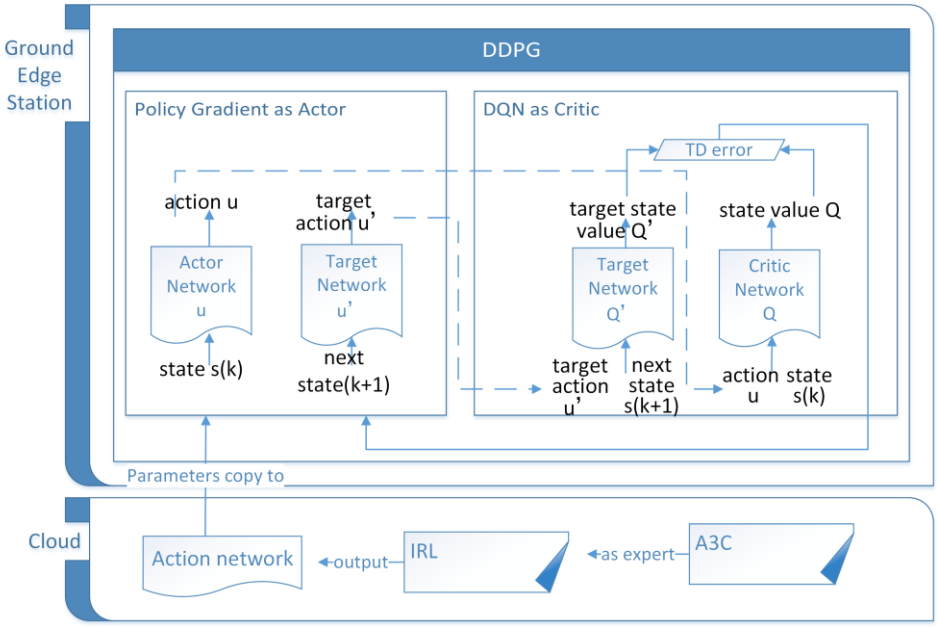

Fig. 4. Schematic diagram of the data flow of the A3CTQ algorithm deployed at the edge

The edge has a compromised computing power and a compromised state data collection capability. The contribution of the edge to the intelligent decision of the system comes from the decentralized deployment of its spatial location. As the source of environmental awareness and the end point of action capabilities, the UAV platform's contribution to system intelligence is reflected in distributed parallel discovery and fast execution. The cloud computing center relies on its powerful computing power to extract action strategies from state data. The cloud computing center plays the role of a factory that processes the input state data into intelligent model parameters.

Based on the above analysis, we have designed an architecture that transmits perception from the UAV platform to the ground through computational offloading, and the intelligence processed by cloud computing is transmitted to the air as model parameters.

Specifically, we designed a distributed A3C teach Q-learning algorithm (A3CTQ). The A3C algorithm is deployed and run in the cloud computing center, and A3C's strategic network is used as an expert. The reverse reinforcement learning algorithm is used to imitate A3C's decision-making ability. A Q-learning form is trained and deployed to the UAV. At the same time, a DDPG algorithm's Actor strategy network is trained by the reverse reinforcement

learning algorithm and updated to the edge server. When the UAV is intermittently disconnected, the UAV is assisted by the Q-learning algorithm on the platform.

The algorithm running on the UAV platform is shown in **Table 2**. The algorithm first initializes the relevant parameters of the Q-learning algorithm, and then runs the calculation power offloading algorithm described in Section 5.3 to determine where the calculation tasks should be offloaded. If local computing power is selected, the drone platform runs the standard q-learning prediction algorithm and updates the Q-table, then reads the sensor's SINR data, calculates the network utility, and forms an experience to send to the cloud computing center and edge. If the cloud or edge computing power is selected, the UAV platform starts a timer, and the cloud and the edge simultaneously calculate the best action strategy at the fastest speed and send it to the UAV platform, and then process the received experience data.

Table 2. Cloud-assisted Q-learning algorithm deployed in UAVs

Algorithm 1: Cloud-Assisted Q-learning Algorithm

Input: jamming power vector $\mathbf{y}^{(k)}$, $\mathbf{SINR}^{(k)}$ calculated by receiver UAV, reward vector $u^{(k)}$ and $\mathbf{x}^{(k)}$ generated by policy models in edge servers or cloud computing center

Output: Q-table deployed in UAV platform

Initialize $\alpha, \gamma, \varepsilon, \mathbf{s}^{(1)}$.

$Q(\mathbf{s}, \mathbf{x}) = 0, V(\mathbf{s}) = 0, \forall \mathbf{s}, \mathbf{x}$.

For $k = 1, 2, 3, \cdots$

  Choose a certain Computing Entity to offloading from UAV, edge or Cloud by gaming algorithm:

    Case offloading to UAV Local:

      Choose $\mathbf{x}^{(k)}$ with ε

      Transmit signals with power $\mathbf{x}^{(k)}$;

      Observe $\mathbf{y}^{(k)}, \mathbf{SINR}^{(k)}$ and $u^{(k)}$;

      Update $Q\left(\mathbf{s}^{(k)}, \mathbf{x}^{(k)}\right)$ and $V\left(\mathbf{s}^{(k)}\right)$;

      $\mathbf{s}^{(k+1)} = \mathbf{y}^{(k)}$

    End case

    Case offloading to Edge:

      Transmit $\mathbf{y}^{(k)}, \mathbf{SINR}^{(k)}$ and $u^{(k)}$ to Edge

      Receive $\mathbf{x}^{(k)}$ from Edge

    End case

    Case offloading to Cloud:

      Transmit $\mathbf{y}^{(k)}, \mathbf{SINR}^{(k)}$ and $u^{(k)}$ to Cloud

      Receive $\mathbf{x}^{(k)}$ from Cloud

    End case

  Update $Q\left(\mathbf{s}^{(k)}, \mathbf{x}^{(k)}\right)$ and $V\left(\mathbf{s}^{(k)}\right)$ of UAV from Cloud

End for

Table 3. Cloud-assisted deep deterministic policy gradient algorithm deployed in edge servers

Algorithm 2: Cloud-Assisted Deep Deterministic Policy Gradient Algorithm

Randomly initialize critic network $Q(\mathbf{s},\mathbf{x}\mid\theta^{Q})$ and actor $\mu(\mathbf{s}\mid\theta^{\mu})$ in every Edge Data Center in Ground Station with weights θ^{Q} and θ^{μ} spontaneously.

Initialize target network Q' and μ' with weights $\theta^{Q'}\leftarrow\theta^{Q},\theta^{\mu'}\leftarrow\theta^{\mu}$ in each Edge Data Center

Initialize replay buffer D

For episode = $1,\cdots,M$ do

  Initialize a UO stochastic process N for action exploration

  Receive initial observation state $\mathbf{s}^{(1)}$

  For $k=1,\cdots,T$ do

    Select action $\mathbf{x}^{(k)}=\mu(\mathbf{s}^{(k)}\mid\theta^{\mu})+\mathrm{N}^{(k)}$ according to the current policy and exploration noise

    Execute action $\mathbf{x}^{(k)}$ and observe reward $u^{(k)}$ and observe new state $\mathbf{s}^{(k+1)}$

    $\mathrm{D}\leftarrow(\mathbf{s}^{(k)},\mathbf{x}^{(k)},u^{(k)},\mathbf{s}^{(k+1)})\cup\mathrm{D}$

    Set $y_i=u^{(i)}+\gamma Q'(\mathbf{s}^{(i+1)},\mu'(\mathbf{s}^{(i+1)}\mid\theta^{\mu'})\mid\theta^{Q'})$

    Send D, y_i to Cloud Data Center

    Update critic by minimizing the loss: $L=\frac{1}{N}\sum_{i}\left(y_i-Q(\mathbf{s}^{(i)},\mathbf{x}^{(i)}\mid\theta^{Q})\right)^2$

    Send the parameters of actor network $\mu(\mathbf{s}\mid\theta^{\mu})$ to Edge Data Center

    Receive the comparison results between the actor network of Edge $\mu(\mathbf{s}\mid\theta^{\mu})$ and of Cloud $\mu_{cloud}(\mathbf{s}\mid\theta^{\mu_{cloud}})$

    If $J(\mu_{cloud})>J(\mu)$

      $\theta^{\mu_{cloud}}\leftarrow\theta^{\mu}$

    End if

    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^{\mu}}J\approx\frac{1}{N}\sum_{i}\nabla_{\mathbf{x}}Q(\mathbf{s},\mathbf{x}\mid\theta^{Q})|_{\mathbf{s}=\mathbf{s}^{(i)},\mathbf{x}=\mu(\mathbf{x}^{(i)})}\nabla_{\theta^{\mu}}\mu(\mathbf{s}\mid\theta^{\mu})|_{\mathbf{s}^{(i)}}$$

    Update the target networks:

$$\theta^{Q'}\leftarrow\tau\theta^{Q}+(1-\tau)\theta^{Q'}$$

$$\theta^{\mu'}\leftarrow\tau\theta^{\mu}+(1-\tau)\theta^{\mu'}$$

  End for

End for

Table 4. Deep inversed RL imitating A3C algorithm deployed in cloud computing center.

Algorithm 3: Deep Inversed Reinforcement Learning Imitating A3C Algorithm at thread level

//Assume global shared parameter vectors θ, θ_{υ} and global shared counter $T=0$

//Assume thread-specific parameter vectors θ' and θ_{υ}'

//Each thread in A3C algorithm corresponds to one UAV

Initialize trajectory τ_i of this thread i which corresponds to UAV i

Initialize thread step counter $t\leftarrow 1$

Repeat

  Reset gradients: $d\theta\leftarrow 0$ and $d\theta_{\upsilon}\leftarrow 0$

  Synchronize thread-specific parameters $\theta'=\theta$ and $\theta_{\upsilon}'=\theta_{\upsilon}$

  $t_{start}=t$

  Get state s_t

  Repeat

    Perform a_t according to policy $\pi(a_t\mid s_t;\theta')$

    Receive reward r_t and new state s_{t+1}

    Store $\{s_t,a_t,r_t,s_{t+1}\}$ to thread trajectory τ_i

    $t\leftarrow t+1$

    $T\leftarrow T+1$

  Until terminal s_t or $t-t_{start}==t_{\max}$

$$R=\begin{cases}0 & \text{for terminal } s_t\\ V(s_t;\theta_{\upsilon}) & \text{for non-terminal } s_t\end{cases}$$

  For $i\in\{t-1,\cdots,t_{start}\}$ do

    $R\leftarrow r_i+\gamma R$

    Accumulate gradient w.r.t

$$\theta':\mathrm{d}\theta\leftarrow\mathrm{d}\theta+\nabla_{\theta'}\log\pi(a_i\mid s_i;\theta')\left(R-V(s_i;\theta_{\upsilon}')\right)$$

    Accumulate gradient w.r.t $\theta_{\upsilon}'\leftarrow\mathrm{d}\theta_{\upsilon}+\partial\left(R-V(s_i;\theta_{\upsilon}')\right)$

  End for

  Perform asynchronous update of θ using $\mathrm{d}\theta$ and of θ_{υ} using $\mathrm{d}\theta_{\upsilon}$

Until $T>T_{\max}$

Send τ_i to global network of A3C in Cloud Data Center and Initialize global trajectory pool τ_{global}

Receive trajectory τ_{edge} from Edge Data Center

Merge all the τ_i and τ_{edge}, $\tau_{global}=\left(\bigcup_{i}\tau_i\right)\bigcup\tau_{edge}$

Take τ_{global} as expert experiences, and use Inverse Reinforcement Learning algorithm to output an reward function $R_{global}(.)$ and an actor network $\mu(\tilde{\pi})$

Using $R_{global}(.)$ to generate Q-table and transmit it to UAV to update

Clone the whole group of network in Edge Data Center and train them using $R_{global}(.)$ and $\mu(\tilde{\pi})$ to generate an actor network and send it from Cloud to Edge to update the actor network there

The algorithms deployed at the edges are shown in **Table 3**. The prototype of the algorithm is a DDPG algorithm for processing continuous action space decision problems. The algorithm consists of two parts, the actor part and the critic part. Each part contains two deep neural networks. The two neural networks in the critic part are target network Q' and critic network Q. The two neural networks in the actor part are target network u' and actor network u. The input of the target network Q' is the next state $s^{(k+1)}$ and target action μ'. Its output is target state value Q'. The input of the critic network Q is current state $s^{(k)}$ and action μ, and its output is state value Q. The TD error can be calculated by Q and Q', which is the key value to update the model parameters. The actor part is also composed of two deep neural networks. The input of target network u' is the next state $s^{(k+1)}$ and its output is target action μ' which is a input of target network Q'. The input of actor network u is state $s^{(k)}$ and its output is action μ which is a input of critic network Q. The parameters of the actor network have two update paths, one is to update the parameters through the back propagation of TD error, and the other is to copy the parameters of the policy network trained by the reverse reinforcement learning algorithm in the cloud. **Fig. 4** clearly illustrates this process.

The A3C algorithm deployed in the cloud is shown in **Table 4**. Compared with the DDPG algorithm deployed on the edge, the A3C algorithm deployed in the cloud has three main advantages, namely asynchronous training framework, network structure optimization, and critic evaluation point optimization. The asynchronous training framework is the biggest optimization.

The cloud network consists of a shared network and n asynchronous networks. The shared network is called Global Network and is responsible for the aggregation of calculation results. Each of the n asynchronous networks corresponds to a thread, and each thread is called a worker. Each thread has the same network structure as the public neural network. Each thread independently interacts with the environment to obtain empirical data. These threads interact with each other.

The cloud can get the state information of the entire network. The cloud can get the status information of the entire network. The n workers of the A3C algorithm correspond to one UAV. With the help of the A3C algorithm, the state information of the entire network is effectively used, and the training process can be performed in parallel. After each thread has accumulated enough UAV experience data, it calculates the gradient of the neural network loss function in its own thread, but these gradients do not update the neural network in its own thread, but instead update the global neural network. In other words, the n threads will independently update the neural network model parameters of the common part using the accumulated gradients. Every once in a while, the thread updates the parameters of its own neural network to the parameters of the global neural network. The parameters of the public network store the intelligence that can guide the UAV to make anti-jamming decisions. When the global network converges, the global network will be used as an expert, that is, as a reward function of the reverse reinforcement learning algorithm, to assist the UAV and the edge to improve intelligence. With this reward function, the cloud can easily make Q-table of Q-learning and policy network of DDPG converge. The detailed algorithm process is shown in the **Table 4**.

5. Distributed computation offloading strategy

Because the network is intermittent, it is difficult to implement a centralized offloading strategy. Therefore, we use game theory to establish a decentralized control calculation

offloading strategy. In Section 3.2, we established a linear cost model of time consumption and energy consumption. In the modeling process, we ignored the secondary factors such as receiving energy consumption and retransmission, in order to build a lightweight model. This is because the calculation offloading decision requires additional calculations, and if you deploy a computationally intensive model, you will lose more than you gain.

Table 5. Cloud and edge-assisted UAV network computing offloading algorithm

Algorithm 4: Computing Offloading

Input: positions of UAVs, delay and SINR of data link to edge servers and cloud computing center, task deque lengths of UAV, edge servers and cloud computing center
Output: tasks offloading strategy (local, edge servers or cloud)

Initialization
Each UAV starts a Timer and select its first strategy $a_i = 0$
Compute the initial value of cost function Z
End initialization
For each uav n
  Do:
    If (timer>t1) $a_i = 0$ and stop sending request to Edge or Cloud
    If (timer>t2) choose $a_i = 0$ as final choose and break the loop
    Request the number of tasks in both buffer of Edge and Cloud
    Select the cost of new strategy $a_i' = 1 or 2$
    Compute the cost of new strategy as Z'
      If $(Z < Z')$ then $Z = Z'$
      Else try the last strategy
    Send a request to update
    If (request accepted) then compute $f(.)$ according to the timer and update strategy a_i
    Else choose $a_i = 0$ as final choose
  Until an Equilibrium is achieved
End for

In Eq. (3) and Eq. (5), we use Q^{Edge} and Q^{Cloud} to describe the phenomenon of centralized offloading. In order to effectively avoid the phenomenon of centralized offloading, we establish a game model $\mu(N, A, G)$, where N denotes the number of UAVs, A denotes the set of offloading policies, and G denotes the cost function which was built in Eq. (7). For the convenience of model analysis, we introduce more notations. $a_i^{u_j}$ denotes the offloading policy of UAV u_j.

We define $A^{u_j} = \{s_i; \forall i \in (0:Local, 1:Edge, 2:Cloud)\}$. u_{-j} denotes all UAVs except u_j, and a_{-i} denotes the offloading policy of u_{-j}. $G^{u_j}\left(a_i^{u_j}, a_{-i}\right)$ denotes the cost function of UAV u_j, which is given by

$$G^{u_j}\left(a_i^{u_j}, a_{-i}\right) = \begin{cases} \alpha T_{Local} + \beta E_{Local}, if\ s_i^{u_j} = 0 \\ \alpha T_{Edge} + \beta E_{Edge}, if\ s_i^{u_j} = 1 \\ \alpha T_{Cloud} + \beta E_{Edge}, if\ s_i^{u_j} = 2 \end{cases}. \tag{12}$$

We use the potential game theory to discuss the convergence of this game model. According to the potential game theory, if we can construct a potential function for the cost function, the game model must converge to the Nash equilibrium point. We denote the cost function of UAV n as u_n, which can be given by

$$u_n\left(a_n, a_{-n}\right) = \frac{\overline{s}_m}{c_m}, \tag{13}$$

where m is equal to 0,1,2, which means local, edge, or cloud respectively, and $\overline{s}_m$ is the expected cost of a certain UAV chosen to offload to m for calculation, and c_m c is the number of UAVs that make the same offloading strategy in the same time except itself.

Lemma: The game model $G: \max u_n\left(a_n, a_{-n}\right), \forall n \in N_u$ is a strict potential game, and there is a purely strategic Nash equilibrium.

Proof: To prove that the game model is a strict potential game, we construct a potential function

$$\phi\left(a_n, a_{-n}\right) = \sum_{m=1}^{M}\sum_{l=1}^{c_m}\varphi_m(l), \tag{14}$$

where $\varphi_m(l) \square \frac{\overline{s}_m}{l}$. If any UAV n changes its offloading strategy from a_n to a_n' independently while other UAVs don't change their offloading strategy, the change amount of the cost function is

$$u_n\left(a_n', a_{-n}\right) - u_n\left(a_n, a_{-n}\right) = \varphi_{a_n'}\left(c_{a_n'}+1\right) - \varphi_{a_n}\left(c_{a_n}\right). \tag{15}$$

Accordingly, the amount of change in its potential function is

$$\begin{aligned} &\phi\left(a_n', a_{-n}\right) - \phi\left(a_n, a_{-n}\right) \\ &= \left(\sum_{l=1}^{c_{a_n'}+1}\varphi_{a_n'}(l) + \sum_{l=1}^{c_{a_n}-1}\varphi_{a_n}(l)\right) - \left(\sum_{l=1}^{c_{a_n'}}\varphi_{a_n'}(l) + \sum_{l=1}^{c_{a_n}}\varphi_{a_n}(l)\right), \\ &= \varphi_{a_n'}\left(c_{a_n'}+1\right) - \varphi_{a_n}\left(c_{a_n}\right) \end{aligned} \tag{16}$$

therefore, we have

$$u_n\left(a_n', a_n\right) - u_n\left(a_n, a_{-n}\right) = \phi\left(a_n', a_{-n}\right) - \phi\left(a_n, a_{-n}\right). \tag{17}$$

Eq. (17) satisfies the definition of strict potential game. It can be seen that the proposed offloading game model for UAVs is a strict potential game and has at least a pure strategy Nash equilibrium. Finally, the expression of the potential function is given by

$$\varphi\left(a_{-i}\right) = \begin{cases} \arg\min_{a_i \in A_j}\left(\alpha T_{Local} + \beta E_{Local}\right), \ if\ s_i = 0 \\ \arg\min_{a_i \in A_j}\left(\alpha T_{Edge} + \beta E_{Edge}\right), \ if\ s_i = 1 \\ \arg\min_{a_i \in A_j}\left(\alpha T_{Cloud} + \beta E_{Cloud}\right), \ if\ s_i = 2 \end{cases}. \tag{18}$$

The successful construction of the potential function ensures that the game process must converge after a finite number of iterations. As shown in **Table 5**, we designed a simple polling algorithm to converge the Nash equilibrium point with less computational overhead. Each UAV in the network is initialized asynchronously, using local computing as the default strategy, and starting a timer. If time t_1 is reached, the UAV stops sending requests to Edge or Cloud, but still keeps listening. If time t_2 is reached, the UAV stops listening and immediately offloads the calculation to the local. If the timer does not exceed the time limit t_1, the UAV sends an offload request to Edge and Cloud, and listens to the number of queued tasks sent back by the edge. Based on the data, UAV calculates the value of the cost function of the three offloading strategies, and changes the current strategy to the best. At the same time, the Edge and Cloud sides also add the new requests to the queue. This process is repeated until the timer expires or the best offloading strategy doesn't change for a while, that is, the Nash equilibrium is reached.

6. Simulation results

We have performed simulation experiments on UAV clusters to verify the performance of the proposed intelligent anti-jamming algorithm. We follow the interface specification of the environment class used by openai gym, and use python and pygame to write a simulated UAV swarm's flight environment. The rendering effect of the simulation program is shown in Fig. 5. The software environment for our simulation experiments is python3.7 +pygame1.8 +pytorch1.4 +cuda10 +cudadnn7 +Gym0.15. We use GPU 2080Ti as a simulation computing power resource. We use multi-process techniques to simulate the concurrent execution of the A3C algorithm. We use delayed batch updates to represent the delay of edge computing and cloud computing. Specifically, the latest batch data obtained by edge computing is delayed by 3 time slots before updating the Q-table on UAV. The latest batch data obtained by cloud computing is dalayed by 15 time slots before updating the Q-table on UAV, and delayed by 5 time slots before updating the buffer of DDPG policy networks on the ground stations.

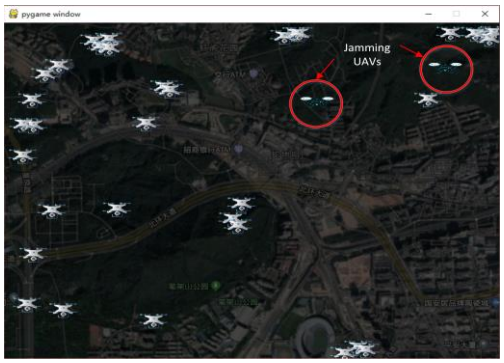

Fig. 5. The rendering effect of the simulation program

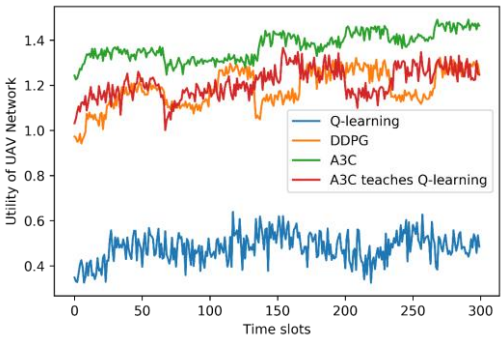

Fig. 6. The utility of UAV network in 4 scenarios

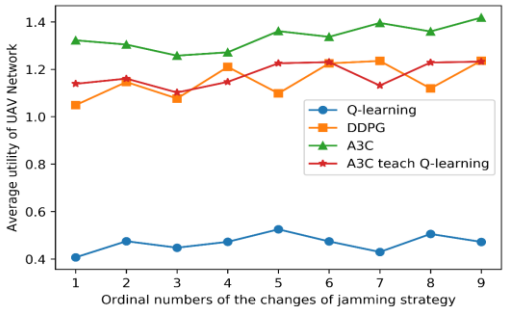

Fig. 7. The average utility of UAV network

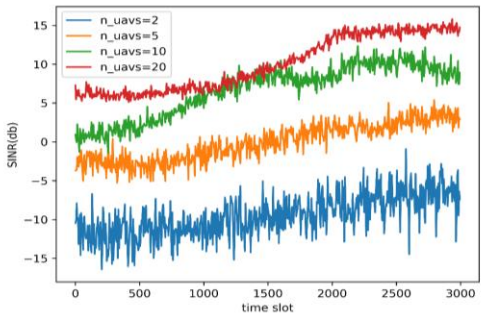

Fig. 8. The average SINR (db) of UAV network

The neural network consists of several fully connected layers and Relu layers. Specifically, the DDPG network is composed of 4 fully connected layers, and the number of neurons in each layer is 256. The A3C network consists of 16 fully connected layers, with 512 neurons per layer. The state space of the reinforcement learning algorithm is represented by a 2-dimensional matrix. Each row of the matrix corresponds to the state of each UAV in the UAV network. The state of each UAV is composed of two parts: channel state and position state. The channel state is a one-dimensional vector, and the dimension is equal to the number of channels of UAV. The value of the channel state vector represents the transmission power allocated by the UAV to each channel. The position status is the position coordinates of the UAV provided by GPS. The normalized channel state and the normalized position state are concatenate into one UAV state vector.

Similarly, the action space is also represented by a 2-dimensional matrix. Each row vector represents the action vector of a UAV. Each row vector is also composed of two parts: the channel vector and the position vector, which respectively represent the target channel power distribution and the target flight position. The actual power allocation strategy of the UAV in the next time slot is the weighted average of the target power distribution and the original power distribution. The velocity of UAV is 0.1m/time slot, and the direction of velocity is the difference between the target position vector and the original position vector. The position of UAV is updated every time slot.

The simulation parameters are chosen similar to [13]. The flying area of the UAVs is 4km * 3km, and the flying height is 150m ~ 500m. There are 3 ground stations with a service radius of 500m. The radio transmission power of the UAV P_T and jammer P_J are both 0.4W. The radio propagation model is given by equation (8), and we use the same parameters as [13] with the specific values $\upsilon = 0.075$, $d_0 = 10$ and $\rho_{free-space} = 2$. The initial position coordinates and directions of velocities of all the UAVs and jamming UAVs are initialized with random values. The position of the ground station is also initialized by a random value. The mobility model of the jammer is a random waypoint model. The number of jammer UAVs N_J is 5. The jamming strategy is frequency hopping jamming. The set of frequency hopping points is updated every 150 time slots according to the statistical rule of the detected signal.

We use communication session loss probability p_{lost} to simulate a communication disconnection. When the communication session is lost, we roll back the Q-table value on the UAV platform to the value 10 time slots ago. When the average utility value of UAV network $U \in [0,0.5)$, $p_{lost} = 0.3$;when $U \in [0,5,1.0)$, $p_{lost} = 0.2$; when $U \in [1.0,1.5)$, $p_{lost} = 0.1$.

In addition to utility, we use Signal to Interference plus Noise Ratio (SINR) as an important indicator to measure communication quality. In the mesh network, each UAV node only

communicates with neighbor nodes. We define the SINR of a UAV as the SINR of the receiver of its closest neighbor UAV node. The signal power is the nearest neighbor transmitter power P_{send} (dbm) plus the transmit antenna gain 5 (dbi) minus the propagation loss calculated by equition (8) (db). The interference power P_{jammer} (dbm) is the power of all jammers plus the antenna gain 5 (dbi), minus the propagation loss (db). Gaussian noise gain value is 5 (db).

We simulated the anti-jamming strategy of the UAV network in four scenarios. In Scenario 1, the Q-learning algorithm is deployed on the UAV platform and the communication session loss mechanism is activated. In scenario 2, the UAV platform deploys the DDPG algorithm, and the communication session loss mechanism is not activated. In scenario 3, the A3C algorithm is deployed on the UAV platform, and the communication session loss mechanism is not activated. In scenario 4, the UAV platform deploys the Q-learning algorithm, the edge servers deploy the DDPG algorithm, the cloud computing center deploys the A3C algorithm, and the communication session loss mechanism is activated. We performed 300 time slots simulations for each algorithm and each time slot samples and learns from the buffer 100 times on a computer with 3.6GHz Intel Core i9-9900, 2080Ti GPU and 48GB of RAM. The initial parameters of each model are the values obtained after 48 hours training between single UAV and single jamming UAV. During the simulation, the jammer randomly changed the jamming strategy 9 times. The utility of UAV network in 4 scenarios are shown in **Fig. 6**, and the average utility of UAV network in 4 scenarios when the jamming UAVs change their jamming strategy are shown in **Fig. 7**. The average utility of uav network against nine times smart jamming attack are shown in **Table 6**.

Table 6. Average utility of UAV network against nine times smart jamming attack

	Q-learning	DDPG	A3C	A3CTQ
1	0.40668	1.04885	1.32220	1.13869
2	0.47486	1.14631	1.30424	1.16067
3	0.44746	1.07666	1.25746	1.10192
4	0.47224	1.20989	1.27152	1.14688
5	0.52493	1.09835	1.36089	1.22525
6	0.47389	1.22492	1.33674	1.23027
7	0.42943	1.23520	1.39541	1.13140
8	0.50541	1.11916	1.35949	1.22849
9	0.47199	1.23591	1.41785	1.23222

Table 7. Cost of UAV network with different CPU cycles

CPU Cycles $(\times 10^4)$	Local	Edge	Cloud	Game theory
1	0.005005	2.503001	3.002500	0.000005
5	0.025025	2.513005	3.002501	0.000125
10	0.050050	2.525510	3.002501	0.000501
20	0.100100	2.550520	3.002502	0.002002
500	2.502500	3.751000	3.002550	1.251250
1000	5.005000	5.001500	3.002600	1.801320

In the simulation of 4 scenarios we conducted, scenarios 1 and 4 can be deployed in a real environment, and scenarios 2 and 3 are for comparison only, and are not realistically feasible. This is because we deployed the DDPG and A3C deep learning algorithm on the UAV platform in scenarios 2 and 3, but neither the power resources nor the computing resources of the UAV platform can support the real-time training of the deep learning algorithm.

The experimental results show that, without the support of cloud computing center or edge servers, the network performance is very poor only relying on the Q-learning algorithm on the UAV platform for anti-jamming decision. For example, **Fig. 7** and **Table 6** show that the maximum average network utility of scenario 1 that does not rely on cloud computing center and edge support is 0.52493, far less than the minimum value of the other three scenarios, which is 1.04885. The simulation results also show that the performance of A3C algorithm

deployed in the cloud is better than that of DDPG algorithm deployed on the edge. When using A3C algorithm, the network utility is 0.18117 higher than DDPG algorithm, with an increase of 15.69%. If we adopt our A3CTQ algorithm which can be deployed in the low-cost UAV swarm, the network utility is only 11.89% lower than that of A3C algorithm, which is close to the performance of undeployed DDPG algorithm (only 1.93% performance gap). With the help of cloud and edge, the performance of our algorithm is improved by 0.70988 (up to 151.87%) compared with the original Q-learning algorithm.

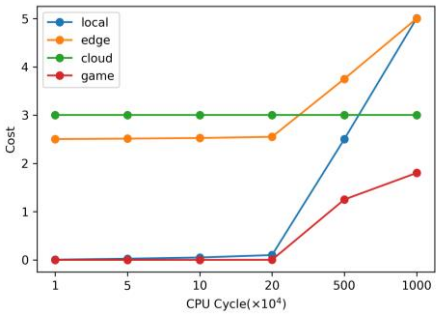

Fig. 9. The cost of UAV network with different CPU cycles where data size is 5kb

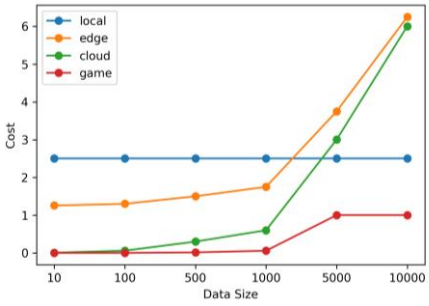

Fig. 10. The cost of UAV network with different data size

In the UAV network, the cluster size has an important impact on the ability of anti-jamming. In order to verify the effect of the number of UAVs on the anti-jamming capability, we set the number of jammers to 5 and the number of UAVs to 2, 5, 10, and 20 respectively for simulation experiments. Each experiment carried out 10 episodes, and each episode carried out 3000 steps. The other parameters of the simulation experiment are the same as the above experiment.

Simulation results show that as the number of UAVs increases, the average SINR of the UAV network continues to increase and the variance continues to decrease. When the number of drones is significantly smaller than the number of jammers, the average SINR of the network is negative, and when the number of UAVs is greater than the number of jammers, the average SINR is positive. Under all parameters, as the time slot increases, the average SINR generally shows an increasing trend, but the network's limit SINR tends to be close to 15db. When the number of UAVs is more than twice the number of jammers, the average SINR of the network increases slowly. Under 4 sets of simulation parameters, the average SINR of the last 1000 time slots are -7.24268、2.39549、9.81378、14.23486 respectively.

Now, we verify the performance of the proposed offloading algorithm. The idea of verifying the algorithm is to fix the data size of the model parameters and examine the cost of the offloading algorithm under different CPU frequencies. After that, we fixed the CPU frequency required for the calculation task, and examined the network cost function value under the data size of different model parameters. There are four algorithms involved in the comparison, namely: (i) Local Computing, (ii) Offloading to Edge servers, (iii) Offloading to Cloud and (iv) proposed game theory algorithm. We choose the network cost as the metric, which consists of two parts, namely delay and energy consumption. We assume the important weight of delay is equals to the important weight of energy consumption, then we set $\alpha = \beta = 1/2$.

We first examine the impact of different CPU cycles required by anti-smart jamming algorithms on our offloading algorithm performance. The UAV platform has weak computing power, so we set the UAV platform CPU frequency to 1Ghz. The edge servers deployed in the ground control station are usually high-performance servers, therefore we set their CPU frequency to 5GHz. The cloud server is a high-performance server cluster, and we set its CPU frequency to 50GHz. The energy consumption is composed of two parts: CPU calculation energy consumption and ground-air communication energy consumption. We assume that the computational energy consumption generated by the UAV platform for each CPU cycle is 1×10^{-6} J, and the computational energy consumption of the edge server is 0.5×10^{-6} J. Since the cloud computing center has sufficient energy, its energy consumption is negligible. In the network model of this article, UAV does not have a direct link to the cloud, so computing offloading to the cloud needs to be forwarded by the edge server. Therefore, this article assumes that the communication energy consumption of the UAV platform to offload a computing task to the edge server is 1.2×10^{-3} J, and the communication energy consumption of offloading to the cloud is 1.0×10^{-3} J. The communication bandwidth from UAV platform to the edge server is 5Mbps/s, and the communication bandwidth to the cloud is 1Mbps/s. We set the health threshold t_1 is 50ms and failure threshold t_2 is 2000ms.

We set the size of data needed to offloading to 5kb, and study the effectiveness of our proposed offloading algorithm under anti-smart jamming algorithm with different calculation complexity where CPU cycles is 1, 5, 10,20, 500 and 1000 ($\times10^4$) respectively. The simulation results are shown in **Fig. 9** and **Table 7**. We can see that the performance of our proposed algorithm based on game theory outperforms the three other algorithm. The cost for offloading game algorithm increases slowly with the CPU cycles. From **Table 7** we find that the performance of offloading to cloud algorithm remains basically unchanged, and the cost of UAV network is about 3.0025. This is because the computing cost of the cloud is extremely low, so when the data size does not change, the cost of the cloud will basically remain the same. When the CPU cycle is between 20 and 500 ($\times10^4$), the performance of offloading to edge algorithm is worse than that of the cloud. This is because as the CPU cycle increases, the intelligent algorithms used become more and more complex, and the gap between the computing performance of edge servers and the cloud gradually appears. For similar reasons, the performance of cloud algorithms is gradually better than UAV local computing. When the CPU cycle is 500 ($\times10^4$), the offloading to cloud algorithm has 0.74845 advantage over the edge algorithm. When CPU cycle is 1000 ($\times10^4$), this cloud algorithm has 1.9989 advantage over local computing. The proposed game algorithm makes the three baseline algorithms work together. The average cost of the proposed game algorithm is 0.7721 lower than the local algorithm, 2.63156 lower than the edge algorithm and 2.49333 lower than the cloud algorithm.

Then we examine the impact of different data size on our offloading algorithm performance. The data size is related to the discretization accuracy of the state space, the number of UAVs and the number of communication channels. We set CPU cycle to 5000000 which is associated with a deep reinforcement learning algorithm, and study the effectiveness of our proposed offloading algorithm where the data size is 10, 100, 500, 1000, 5000 and 10000 bits respectively. The simulation results are shown in **Fig. 10** and **Table 8**. We find that the performance of our proposed algorithm outperforms the three other algorithm. The average cost of the proposed game algorithm is 2.15646 lower than the local algorithm, 2.28891 lower than the edge algorithm and 1.31640 lower than the cloud algorithm. The cost of offloading to local is always 2.50250. This is because local computing has only computational energy consumption but no communication energy consumption. Therefore, the performance of the

local algorithm does not change with the data size. As the data size increases, the cost of communication will increase. As shown in **Fig. 10**, when the data size is greater than 1kb, the cost of the edge algorithm and the cloud algorithm both exceed the local algorithm. This also confirms the necessity of introducing 5G communication technology into the UAV network. The above two simulation scenarios together prove the effectiveness of our proposed algorithm.

Table 8. Cost of UAV network with different data sizes

Data Sizes	**Local**	**Edge**	**Cloud**	Game theory
10	2.50250	1.25550	0.00606	0.00001
100	2.50250	1.30051	0.06010	0.00058
500	2.50250	1.50055	0.30030	0.01443
1000	2.50250	1.75060	0.60055	0.05769
5000	2.50250	3.75100	3.00255	1.00176
10000	2.50250	6.25150	6.00505	1.00176

7. Conclusion

In this paper, we propose an A3CTQ algorithm which can be deployed in the low-cost UAV swarms. This algorithm can effectively use the cloud and edge computing power to maintain a high level of smart anti-jamming decision-making ability when the network communication session may be interrupted by smart jamming attack. In order to improve the feasibility of algorithm deployment, we establish a game model for distributed control of computing offload to cloud and edge. The simulation results show that when the network communication session may be interrupted due to smart jamming attacks, the performance of our algorithm is 151.87% higher than that of the original Q-learning algorithm.

In this paper, we assume that the jamming sources in different geographical locations must use the same channel jamming mode at the same time. In the next research, we will deeply study the situation that the jamming modes of multiple jamming sources are not uniformly selected and the jam.

References

[1] D. Yang, J. Zhang, X. Fang, A. Richa, and G. Xue, “Optimal transmission power control in the presence of a smart jammer,” in *Proc. of 2012 IEEE Global Communications Conference (GLOBECOM)*, pp. 5506-5511, 2012. Article (CrossRef Link)

[2] L. Xiao, J. Liu, Y. Li, N. B. Mandayam, and H. V. Poor, "Prospect Theoretic Analysis of Anti-jamming Communications in Cognitive Radio Networks," in *Proc. of 2014 IEEE Global Communications Conference*, pp. 746-751, 2014. Article (CrossRef Link)

[3] Y. E. Sagduyu, R. A. Berry, and A. Ephremides, “Jamming Games in Wireless Networks with Incomplete Information,” *IEEE Communications Magazine,* vol. 49, no. 8, pp. 112-118, Aug. 2011. Article (CrossRef Link)

[4] A. Garnaev, Y. Liu, and W. Trappe, “Anti-jamming Strategy Versus a Low-Power Jamming Attack When Intelligence of Adversary's Attack Type is Unknown,” *IEEE Transactions on Signal and Information Processing over Networks,* vol. 2, no. 1, pp. 49-56, Mar. 2016. Article (CrossRef Link)

[5] L. Jia, F. Yao, Y. Sun, Y. Niu, and Y. Zhu, “Bayesian Stackelberg Game for Antijamming Transmission With Incomplete Information,” *IEEE Communications Letters,* vol. 20, no. 10, pp. 1991-1994, Oct. 2016. Article (CrossRef Link)

[6] L. Jia, F. Yao, Y. Sun, Y. Xu, S. Feng, and A. Anpalagan, "A Hierarchical Learning Solution for Anti-Jamming Stackelberg Game With Discrete Power Strategies," *IEEE Wireless Communications Letters,* vol. 6, no. 6, pp. 818-821, Dec. 2017. Article (CrossRef Link)
[7] Y. Wu, B. Wang, K. R. Liu, and T. C. Clancy, "Anti-jamming games in multi-channel cognitive radio networks," *IEEE Journal on Selected Areas in Communications,* vol. 30, no. 1, pp. 4-15, 2011. Article (CrossRef Link)
[8] H. Saad, A. Mohamed, and T. ElBatt, "Cooperative Q-learning techniques for distributed online power allocation in femtocell networks," *Wireless Communications & Mobile Computing,* vol. 15, no. 15, pp. 1929-1944, Oct. 2015. Article (CrossRef Link)
[9] C. Li, Y. Xu, J. Xia, and J. Zhao, "Protecting Secure Communication Under UAV Smart Attack With Imperfect Channel Estimation," *IEEE Access,* vol. 6, pp. 76395-76401, 2018. Article (CrossRef Link)
[10] L. Xiao, X. Z. Lu, D. J. Xu, Y. L. Tang, L. Wang, and W. H. Zhuang, "UAV Relay in VANETs Against Smart Jamming With Reinforcement Learning," *IEEE Transactions on Vehicular Technology,* vol. 67, no. 5, pp. 4087-4097, May 2018. Article (CrossRef Link)
[11] Q. H. Wu, H. Q. Wang, X. Li, B. Zhang, and J. L. Peng, "Reinforcement Learning-Based Anti-Jamming in Networked UAV Radar Systems," *Applied Sciences-Basel,* vol. 9, no. 23, pp. 22, Dec. 2019. Article (CrossRef Link)
[12] N. I. Mowla, N. H. Tran, I. Doh, and K. Chae, "AFRL: Adaptive federated reinforcement learning for intelligent jamming defense in FANET," *Journal of Communications and Networks,* vol. 22, no. 3, pp. 244-258, 2020. Article (CrossRef Link)
[13] L. Xiao, D. Jiang, D. Xu, H. Zhu, Y. Zhang, and H. V. Poor, "Two-Dimensional Antijamming Mobile Communication Based on Reinforcement Learning," *IEEE Transactions on Vehicular Technology,* vol. 67, no. 10, pp. 9499-9512, Oct. 2018. Article (CrossRef Link)
[14] L. Xiao, C. Xie, M. Min, and W. Zhuang, "User-Centric View of Unmanned Aerial Vehicle Transmission Against Smart Attacks," *IEEE Transactions on Vehicular Technology,* vol. 67, no. 4, pp. 3420-3430, Apr. 2018. Article (CrossRef Link)
[15] N. Gao, Z. Qin, X. Jing, Q. Ni, and S. Jin, "Anti-Intelligent UAV Jamming Strategy via Deep Q-Networks," *IEEE Transactions on Communications,* vol. 68, no. 1, pp. 569-581, 2020. Article (CrossRef Link)
[16] B. Alzahrani, O. S. Oubbati, A. Barnawi, M. Atiquzzaman, and D. Alghazzawi, "UAV assistance paradigm: State-of-the-art in applications and challenges," *Journal of Network and Computer Applications,* vol. 166, pp. 44, Sep. 2020. Article (CrossRef Link)
[17] X. Z. Lu, D. J. Xu, L. Xiao, L. Wang, W. H. Zhuang, and IEEE, "Anti-Jamming Communication Game for UAV-aided VANETs," in *Proc. of 2017 IEEE Global Communications Conference*, pp. 1-6, 2017. Article (CrossRef Link)
[18] H. Baek and J. Lim, "Design of Future UAV-Relay Tactical Data Link for Reliable UAV Control and Situational Awareness," *IEEE Communications Magazine,* vol. 56, no. 10, pp. 144-150, 2018. Article (CrossRef Link)
[19] X. Lu, L. Xiao, C. Dai, and H. Dai, "UAV-Aided Cellular Communications with Deep Reinforcement Learning Against Jamming," *IEEE Wireless Communications,* vol. 27, no. 4, pp. 48-53, 2020. Article (CrossRef Link)
[20] Y. Zhou, C. Pan, P. L. Yeoh, K. Wang, M. Elkashlan, B. Vucetic, and Y. Li, "Secure Communications for UAV-Enabled Mobile Edge Computing Systems," *IEEE Transactions on Communications,* vol. 68, no. 1, pp. 376-388, 2020. Article (CrossRef Link)

Zhiwei Li was born in Shijiazhuang, China, in 1986. He received the M.S. degree in communication and information system from the Shijiazhuang Campus of Army Engineering University, Shijiazhuang, China, in 2015, where he is currently pursuing the Ph.D. degree with the UAV Engineering Department. His research interests are in the areas of communication scheme of UAVs network, Deep Reinforcement Learning and software-defined UAVs network.

Yu Lu was born in Luoyang, Henan, China, in 1960. He received the B.S. and M.S. degrees in communication engineering from National Defense Technology University, Changsha, China, and the Ph.D. degree in communication and information engineering from the Beijing University of Aeronautics and Astronautics, Beijing, China, in 2003. He is currently a Full Professor with the UAV Engineering Department, Shijiazhuang Campus of Army Engineering University, Shijiazhuang, China. His research interests are in the areas of communication scheme of UAVs network and software defined security.

Zengguang Wang received his Master's degree from Ordnance Engineering College and the Ph.D degree in the Equipment Command and Administration Department, Shijiazhuang Campus of Army Engineering University, Shijiazhuang, China. Now, he is an engineer at the National Defense University. His current interests are network security and defense, equipment support informatization.

Wenxin Qiao was born in Luoyang, Henan, China, in 1992. He received the M.S. degree in software engineering from the Shijiazhuang Campus of Army Engineering University, Shijiazhuang, China, in 2016, where he is currently pursuing the Ph.D. degree with the Information Engineering Department. His research interests include networking and communication.

Donghao Zhao received his MS degree in communication and information system from Army Engineering University, Shijiazhuang, China. He is currently a lecturer in Shijiazhuang Campus, Army Engineering University, China. His current interest are in areas of edge computing and blockchain enabled UAV communication.