Secure Transmission for Multi-UAV-Assisted Mobile Edge Computing Based on Reinforcement Learning

Weidang Lu, *Senior Member, IEEE*, Yandan Mo, Yunqi Feng, Yuan Gao, Nan Zhao, *Senior Member, IEEE,* Yuan Wu, *Senior Member, IEEE*, and Arumugam Nallanathan, *Fellow, IEEE*

***Abstract*—UAV communication has received widespread attention in MEC systems due to its high flexibility and line-of-sight transmission. Users can reduce their local computing pressures and computation delay by offloading tasks to the UAV as an edge server. However, the coverage capability of a single UAV is very limited. Moreover, the data offloaded to the UAV will be easily eavesdropped. Thus, in this paper, we propose two secure transmission methods for multi-UAV-assisted mobile edge computing based on the single-agent and multi-agent reinforcement learning, respectively. In the proposed methods, we first utilize the spiral placement algorithm to optimize the deployment of UAVs, which covers all users with the minimum number of UAVs. Then, to reduce the information eavesdropping by a flying eavesdropper, we utilize the reinforcement learning to optimize the secure offloading to maximize the system utility by considering different types of users' tasks with diverse preferences for residual energy of computing equipment and processing delay. Simulation results indicate that compared with the single-agent method and the benchmark, the multi-agent method can optimize the offloading in a better manner and achieve larger system utility.**

***Index Terms*—UAV communication, mobile edge computing, reinforcement learning, secure transmission.**

I. Introduction

The fifth generation (5G) mobile communications have brought many applications with high latency and bandwidth requirements [1]–[3]. Most of the intelligent services of 5G, e.g., automatic drive and virtual reality, have extremely high requirements on computing delay, which makes it difficult to process the huge number of computing tasks within an very short time. On the other hand, due to the limited energy of mobile devices, the high energy consumption caused by the large number of computing tasks is also a challenging puzzle [4], [5]. To solve these problems, mobile edge computing (MEC) and its offloading methods have been proposed [6], [7]. With the help of MEC, the mobile users on the edge of the network can send tasks to the edge servers for computing, thereby reducing the energy consumption and latency [8], [9].

Equipping edge servers on unmanned aerial vehicles (UAVs) enables rapid network deployment regardless of the terrain factors because the UAV has high mobility and line-of-sight (LoS) transmission capabilities. It can also greatly reduce the computing energy consumption and latency [10]. Under the limitation of energy, Li *et al.* maximized the migration throughput of user tasks by utilizing UAV as the edge server [11]. Zhang *et al.* proposed a UAV-assisted MEC system includes a base station (BS) with MEC server, a UAV-MEC server and multiple mobile devices, in which a weighted system cost including both the energy consumption and delay is minimized through optimizing the offloading [12]. Liu *et al.* jointly optimized the central processing unit (CPU) control, offloading strategy and flight trajectory to minimize the total energy consumption of the UAV [13]. Zhang *et al.* minimized the total energy consumption through jointly optimizing the bit allocation, slot scheduling, power allocation and trajectory in the UAV-assisted MEC system [14]. Zhang *et al.* utilized UAV as the edge server and relay to minimize the average delay through sequentially solving the joint user association and computing resource allocation problem, the communication resource allocation problem and the UAV layout problem [15]. In order to minimize the energy consumption of mobile devices, UAV's propulsion and computing power in the UAV-assisted MEC system, Tun *et al.* jointly optimized the trajectory, resource allocation, and task offloading [16].

Due to the limited coverage, one single UAV can only provide MEC services for a limited number of users within the coverage. In order to serve more users to realize edge computing, multiple UAVs need to be deployed. Haber *et al.* investigated the problem of multi-UAV-assisted ultra-reliable, low-latency computational offloading, which maximized the rate of served requests by optimizing the UAV's positions, offloading decisions and the allocated resources while meeting the requirements of delay and reliability [17]. Wang *et al.* established a multi-UAV collaborative MEC framework and minimized the system's energy consumption under the condition that all offloading tasks are completed [18]. Xiao *et al.* jointly optimized the flight trajectory and offloading allocation to minimize the energy consumption of the multi-UAV-assisted MEC system [19]. Yang *et al.* found the optimal offloading with the goal of global load balancing in the multi-

W. Lu, Y. Mo and Y. Feng are with College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China (e-mail: luweid@zjut.edu.cn, 2112003310@zjut.edu.cn, yqfeng@zjut.edu.cn).

Y. Gao is with the Department of Electronic Engineering, Tsinghua University, Beijing 100084, China (e-mail: yuangao08@tsinghua.edu.cn).

N. Zhao is with the School of Information and Communication Engineering, Dalian University of Technology, Dalian 116024, P. R. China (e-mail: zhaonan@dlut.edu.cn).

Y. Wu is with the State Key Laboratory of Internet of Things for Smart City, University of Macau, Macao SAR, China, and also with the Department of Computer and Information Science, University of Macau (e-mail: yuanwu@um.edu.mo).

A. Nallanathan is with School of Electronic Engineering and Computer Science, Queen Mary University of London, E1 4NS, U.K. (e-mail: a.nallanathan@qmul.ac.uk).

UAV-assisted MEC system [20]. Sun *et al.* optimized the 3-dimensional (3D) deployment of UAVs to reduce the total delay under the condition that UAVs complete all offloading task of users [21]. With the goal of reducing energy consumption, Wang *et al.* jointly optimized the task scheduling and deployment of UAVs for the multi-UAV-assisted MEC system [22]. Wang *et al.* jointly optimized each UAV's trajectory and the offloading decision from all the users to maximize the fairness among all the users and the fairness of user-load of each UAV, as well as minimizing the energy consumption of all the users [23].

It is noticed that in the UAV-assisted MEC system, the data offloaded to the UAV for edge computing can be easily eavesdropped, as it is transmitted through LoS links. Therefore, the secure transmission of offloading data in the UAV-assisted MEC system is significant. Based on the deep reinforcement learning, Jing *et al.* maximized the average secrecy rate of users through the optimization of the flight trajectory in the UAV-assisted MEC system [24]. Gu *et al.* jointly optimized the computing and communication resource to minimize the UAV computing and offloading energy consumption on the premise of satisfying the constraints of secure offloading rate and computation delay [25]. Under the constraints of the secure offloading rate and transmit power, Li *et al.* minimized the total energy consumption of the UAV through the optimization of the task allocation, user transmit power and flight trajectory [26]. Xu *et al.* jointly optimized the computation resource and flight trajectory to maximize the minimum secrecy capacity in the UAV-assisted secure MEC system for both the time division multiple access (TDMA) and non-orthogonal multiple access (NOMA) [27]. Comparing with the ground eavesdroppers, which are deployed at the fixed locations in the above works, UAV eavesdroppers will have much better channel condition due to the LoS transmission. Thus, the offloading data can be easily overheard by UAV eavesdroppers. The major challenges for considering UAV eavesdroppers are to consider the uncertainty of UAV eavesdroppers' position and anti-collision constraint between UAVs. Han and Zhou jointly optimized the UAV's position, transmit power of users, task offloading ratio and UAV jamming power with the target of maximizing the minimum secrecy capacity, when UAV eavesdroppers overhear the offloading data [28], [29].

However, in the existing research works, the secure transmission for UAV assisted MEC system is only considered with one single UAV, while the security of multi-UAV-assisted MEC systems is largely ignored. As a consequence, this paper proposes two secure transmission schemes for multi-UAV-assisted MEC based on the single-agent and multi-agent reinforcement learning, respectively. In the proposed methods, we first utilize the spiral placement algorithm to optimize the deployment of UAVs, covering all the users with the minimum number of UAVs. Then, we utilize the reinforcement learning to optimize the secure offloading to maximize the system utility.

The major contributions of this paper are summarized as follows:

1) We propose a secure transmission model for multiple UAVs assisted MEC, including multiple ground users, multiple UAVs as MEC edge servers, one eavesdropping UAV and one jammer to send artificial jamming.
2) The spiral placement algorithm is utilized to solve the placement problem of UAVs. Then, we formulate an optimization problem to maximize the system utility for the secure offloading, which considers the limitation of the secure offloading transmission rate, computing latency, energy consumption and task types.
3) In order to obtain the optimal secure offloading, we transform the long-term process of offloading decision into the Markov decision process. Then, the single-agent and multi-agent schemes based on the reinforcement learning are proposed to maximize the system utility.

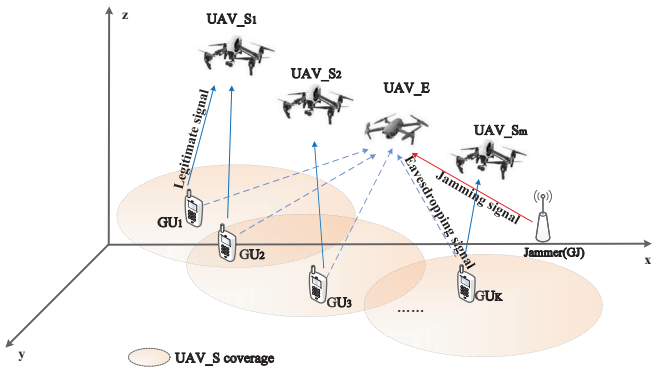


Fig. 1: System model.

The rest of this paper is organized as follows. Section II introduces the system model, secure communication model and computation model. Section III formulates the problem to cope with. Section IV and V give the single-agent scheme and multi-agent scheme based on the reinforcement learning, respectively. Section VI presents the convergence performance of the schemes and gives the simulation results in consideration of different energy limitation. Section VII concludes our works.

II. System Model

In this section, we first introduce the system model. Then, the secure communication model is given. After that, we analyze the local computation and offload computation models. Table I presents the notations in this paper.

A. System Model

We consider a multi-UAV-assisted MEC system with one UAV as the eavesdropper (UAV_E), one ground jammer (GJ) to send the jamming signal, M UAVs as the edge servers (UAV_Ss) and K ground users (GUs) as shown in Fig. 1. We assume that UAV_Ss and the GJ belong to the legitimate network. However, the UAV_E is an eavesdropper which does not belong to the legitimate network. Since UAV_Ss have the perfect knowledge of the jamming signal sent by the GJ via means of synthetic aperture radar [27], [30], they can subtract it from the received signals. However, the jamming signal sent by the GJ can be considered as the noise for the UAV_E,

TABLE I
MAJOR NOTATIONS

Notation	Description
$\mathcal{M}$	The set of UAV_Ss
$\mathcal{K}$	The set of GUs
w_k^{G}	Coordinates of GU_k
w_{J}	Coordinates of GJ
w_{E}	Coordinates of UAV_E
w_m^{S}	Coordinates of $\mathrm{UAV\_S}_m$
$\mathcal{Z}_m$	The set of GUs covered by $\mathrm{UAV\_S}_m$
$w_{z_m}^{\mathrm{GS}}$	Coordinates of GU_{z_m}
R	Coverage radius of UAV_S
$h_{k,m}$	Channel gain between GU_k and $\mathrm{UAV\_S}_m$
$h_{k,e}$	Channel gain between GU_k and UAV_E
$h_{j,e}$	Channel gain between GJ and UAV_E
$r_{k,m}$	The SINR of the received signals from GU_k to $\mathrm{UAV\_S}_m$
$r_{k,e}$	The SINR of the received signals from GU_k to UAV_E
p_k	Transmit power of GU_k
p_j	Transmit power of GJ
δ_m^2	The AWGN at $\mathrm{UAV\_S}_m$
δ_e^2	The AWGN at UAV_E
$R_{k,m}$	Task offloading rate from GU_k to $\mathrm{UAV\_S}_m$
$R_{k,e}$	Eavesdropping rate from GU_k to UAV_E
$R_{k,m}^{\mathrm{sec}}$	Secure offloading rate from GU_k to $\mathrm{UAV\_S}_m$
$\mathcal{G}$	The index, number and the set of task types
L_k^g	The task of GU_k
D_g	Data size of type g task
T_g	The maximum tolerable delay of type g task
f_k^{local}	Computing capacity of GU_k
$T_{k,g}^{\mathrm{local}}$	Local computation delay of task L_k^g
$E_{k,g}^{\mathrm{local}}$	Local energy consumption of task L_k^g
$T_{k,g,m}^{\mathrm{tran}}$	Delay of transmitting task L_k^g from GU_k to $\mathrm{UAV\_S}_m$
W_m	Bandwidth of GU_k
f_m	Computing capacity of $\mathrm{UAV\_S}_m$
$T_{k,g,m}^{\mathrm{comp}}$	Computing delay for task L_k^g at $\mathrm{UAV\_S}_m$
$E_{k,g,m}^{\mathrm{comp}}$	Computing energy consumption for task L_k^g at $\mathrm{UAV\_S}_m$
$T_{k,g}^m$	Total delay for task L_k^g at $\mathrm{UAV\_S}_m$
$T_{k,g}$	Execution delay of task L_k^g
$E_{\max}^g$	The maximum usable energy of GU_k
$E_{\max}^{\mathrm{UAV\_S}}$	The maximum usable energy of $\mathrm{UAV\_S}_m$
$\mathcal{O}_{k,m}$	The set of GUs that have chosen to offload tasks to $\mathrm{UAV\_S}_m$ before GU_k's turn
$R_{\min}^{\mathrm{sec}}$	The minimum secure offloading rate
U^{sys}	System utility
θ_g	Preference of task L_k^g for the delay
η_g	Preference of task L_k^g for the residual energy of computing device

because it does not have any information about the GJ and treats all the received signals as useful ones.

In this paper, the positions of UAV_Ss, UAV_E, GUs and GJ are assumed to be fixed. The coordinates of GU_k ($k \in \mathcal{K}$, $\mathcal{K} = \{1, 2, \ldots, K\}$), GJ and UAV_E are denoted as $w_k^{\mathrm{G}} = (x_k^{\mathrm{G}}, y_k^{\mathrm{G}}, 0)^T$, $w_{\mathrm{J}} = (x_{\mathrm{J}}, y_{\mathrm{J}}, 0)^T$ and $w_{\mathrm{E}} = (x_{\mathrm{E}}, y_{\mathrm{E}}, h_{\mathrm{E}})^T$, respectively, where h_{E} is the flight altitude of UAV_E. The coordinates of $\mathrm{UAV\_S}_m$ ($m \in \mathcal{M}$, $\mathcal{M} = \{1, 2, \ldots, M\}$) are denoted as $w_m^{\mathrm{S}} = (x_m^{\mathrm{S}}, y_m^{\mathrm{S}}, h_m^{\mathrm{S}})^T$, where h_m^{S} is the flight altitude of $\mathrm{UAV\_S}_m$. The set of GUs covered by the $\mathrm{UAV\_S}_m$ is denoted as $\mathcal{Z}_m = \{1, 2, \ldots, Z_m\}$, and the

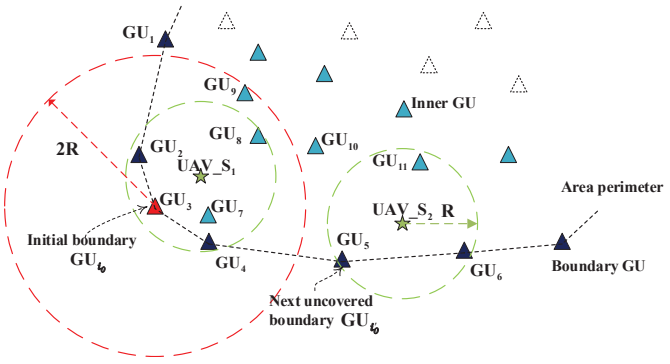


Fig. 2: Illustration of the spiral algorithm.

coordinates are denoted as $w_{z_m}^{\mathrm{GS}} = (x_{z_m}^{\mathrm{GS}}, y_{z_m}^{\mathrm{GS}}, 0)^T$, where $\sqrt{\left(x_{z_m}^{\mathrm{GS}} - x_m^{\mathrm{S}}\right)^2 + \left(y_{z_m}^{\mathrm{GS}} - y_m^{\mathrm{S}}\right)^2} \le R$ with R being the coverage radius of UAV_S. We consider that the UAV_E has better coverage ability than the UAV_S, and thus the UAV_E can cover all the GUs in the system.

The number and coordinates of UAV_Ss are determined by the position distribution of GUs and coverage radius of UAV_S. In order to minimize the number of UAV_Ss to cover all the GUs, we formulate the optimization problem as

$$\min_{w_m^{\mathrm{S}}} \quad M \tag{1}$$

$$s.t. \quad \sqrt{\left(x_{z_m}^{\mathrm{GS}} - x_m^{\mathrm{S}}\right)^2 + \left(y_{z_m}^{\mathrm{GS}} - y_m^{\mathrm{S}}\right)^2} \le R, \forall m, \tag{2}$$

$$\mathcal{Z}_1 \cup \mathcal{Z}_2 \cup \ldots \cup \mathcal{Z}_M = \mathcal{K}, \tag{3}$$

in which the constraint (2) indicates the coverage limitation of the UAV_S, and the constraint (3) guarantees that each GU can be covered by at least one UAV_S.

The above optimization problem can be settled by the spiral placement algorithm [31]. The main idea is to place the UAV_Ss in sequence along the area perimeter, which is defined as the path connecting the extreme points (referred to as the boundary GUs) of the convex hull of all uncovered GUs.

Specifically, we take Fig. 2 as an example to illustrate the steps of the spiral placement algorithm. For placing the first UAV_S, e.g., $\mathrm{UAV\_S}_1$, we randomly select a boundary GU_{i_0} (e.g., GU_3 at the lower left corner denoted by a red triangle) and take $w_{i_0}^{\mathrm{G}}$ as the initial position of $\mathrm{UAV\_S}_1$. Then, for covering as many boundary GUs as possible, we utilize Algorithm 1 to optimize the position of $\mathrm{UAV\_S}_1$ w_1^{S}, which is initialized to w_3^{G}. At this time, we input w_1^{S}, the set $\{\mathrm{GU}_3\}$ as $\mathcal{K}_{prio}$ and the set of uncovered boundary GUs as $\mathcal{K}_{sec}$ to Algorithm 1. Noted that in step 3 of Algorithm 1, the 1-center problem can be solved by several algorithms [32], [33]. After that, as shown in Fig. 2, the boundary GU_2 and GU_4 can be covered while ensuring that the boundary GU_3 can be covered. Then, we proceed to cover as many inner GUs as possible through Algorithm 1 similarly while the boundary GU_2, GU_3 and GU_4 are guaranteed to be covered first. At this time, we input the updated w_1^{S}, the set $\{\mathrm{GU}_2, \mathrm{GU}_3, \mathrm{GU}_4\}$ as $\mathcal{K}_{prio}$ and the set of uncovered inner GUs as $\mathcal{K}_{sec}$ to Algorithm 1. After that, the inner GU_7 and GU_8 will be covered. We use a green pentagram to denote the final position of $\mathrm{UAV\_S}_1$ as shown

in Fig. 2. For placing the second UAV_S UAV_S$_2$, we choose the first uncovered boundary GU$_{i_0'}$ counterclockwisely next to GU$_{i_0}$ as the initial position of UAV_S$_2$, which in this case is GU$_5$. Then, repeat the above steps to find the final position of UAV_S$_2$. The above process repeats until all the GUs are covered.

Algorithm 1 UAV_S Position Optimization Algorithm

Input: The position of the new UAV_S w_m^{S} ($m \in \mathcal{M}$) that to be optimized, the set of GUs which must be covered fist $\mathcal{K}_{prio}$, the set of uncovered boundary/inner GUs $\mathcal{K}_{sec}$.

1: **while** $\mathcal{K}_{sec} \neq \varnothing$ **do**
2: Update set $\mathcal{K}_{sec}$ by excluding GUs more than $2R$ away from any GU in set $\mathcal{K}_{prio}$. Update set $\mathcal{K}_{prio}$ ($\mathcal{K}_{sec}$) by including (excluding) GUs within distance R to w_m^{S}.
3: Find GU$_{i_1} \in \mathcal{K}_{sec}$ with shortest distance to w_m^{S}. Add (remove) GU$_{i_1}$ to (from) $\mathcal{K}_{prio}$ ($\mathcal{K}_{sec}$) if it can be covered by optimizing w_m^{S} via settling the 1-center problem.
4: **end while**

Output: The optimized position of the new UAV_S w_m^{S}.

B. Secure Communication Model

The channels between UAVs and GUs follow the free-space path loss. The channel gain between GU$_k$ and UAV_S$_m$ can be given by

$$h_{k,m} = \frac{\mu}{d_{k,m}^2}, \tag{4}$$

where μ is the channel gain at the reference distance of 1 m, and $d_{k,m} = \sqrt{\left(x_k^{\mathrm{G}} - x_m^{\mathrm{S}}\right)^2 + \left(y_k^{\mathrm{G}} - y_m^{\mathrm{S}}\right)^2 + {h_m^{\mathrm{S}}}^2}$ is the distance between the GU$_k$ and UAV_S$_m$.

Meanwhile, the channel gain between the GU$_k$ and UAV_E can be defined as

$$h_{k,e} = \frac{\mu}{d_{k,e}^2}, \tag{5}$$

where $d_{k,e} = \sqrt{\left(x_k^{\mathrm{G}} - x_{\mathrm{E}}\right)^2 + \left(y_k^{\mathrm{G}} - y_{\mathrm{E}}\right)^2 + {h_{\mathrm{E}}}^2}$ is the distance between the GU$_k$ and UAV_E.

The channel gain between the GJ and UAV_E can be defined as

$$h_{j,e} = \frac{\mu}{d_{j,e}^2}, \tag{6}$$

where $d_{j,e} = \sqrt{\left(x_{\mathrm{J}} - x_{\mathrm{E}}\right)^2 + \left(y_{\mathrm{J}} - y_{\mathrm{E}}\right)^2 + {h_{\mathrm{E}}}^2}$ is the distance between the GJ and UAV_E.

We assume that a group of M non-overlapping channels are assigned to the UAV_Ss for collecting task data from the GUs. The GUs offloaded the data to the same UAV_S will use TDMA to avoid the co-channel interference [21]. Therefore, the signal-and-noise ratio (SNR) of the received signals from the GU$_k$ to UAV_S$_m$ can be given by

$$r_{k,m} = \frac{h_{k,m}p_k}{\delta_m^2}, \forall k, m, \tag{7}$$

where p_k is the transmit power of GU$_k$, and δ_m^2 is the additive white Gaussian noise (AWGN) at UAV_S$_m$.

As the UAV_E cannot distinguish the signals released by GU$_k$ or GJ, the signal-to-interference-and-noise ratio (SINR) of the received signals from the GU$_k$ to UAV_E can be given by

$$r_{k,e} = \frac{h_{k,e}p_k}{h_{j,e}p_j + \delta_e^2}, \forall k, \tag{8}$$

where p_j is the transmit power of GJ, and δ_e^2 is the AWGN at UAV_E.

Thus, the task offloading rate from the GU$_k$ to UAV_S$_m$ can be given by

$$R_{k,m} = \log_2\left(1 + \frac{h_{k,m}p_k}{\delta_m^2}\right), \forall k, m. \tag{9}$$

The eavesdropping rate from the GU$_k$ to UAV_E can be given by

$$R_{k,e} = \log_2\left(1 + \frac{h_{k,e}p_k}{h_{j,e}p_j + \delta_e^2}\right), \forall k. \tag{10}$$

As a result, the secure offloading rate from the GU$_k$ to UAV_S$_m$ can be expressed as

$$R_{k,m}^{\mathrm{sec}} = \left(R_{k,m} - R_{k,e}\right)^+, \forall k. \tag{11}$$

C. Computation Model

Assume that each GU has one certain type of task to be processed. The set of task types is denoted as $\mathcal{G} = \{1, 2, \ldots, G\}$. If the task type of GU$_k$ is g, the GU$_k$ task is denoted as L_k^g, $L_k^g \triangleq (D_g, T_g)$, where D_g is the data size of type g task and T_g is the maximum tolerable delay for this type of task. GUs can choose to compute its task locally, or offload its task to a connectable UAV_S for computation.

1) Local Computing: If the GU$_k$ chooses to complete its task L_k^g locally, we define $T_{k,g}^{\mathrm{local}}$ as the local computation delay, which only includes the computing time depending on the GU$_k$'s CPU. Then, $T_{k,g}^{\mathrm{local}}$ can be expressed as

$$T_{k,g}^{\mathrm{local}} = \frac{D_g C^{\mathrm{local}}}{f_k^{\mathrm{local}}}, \forall k, \tag{12}$$

where C^{local} is the number of CPU cycles required by the local device for computing one bit of task, and f_k^{local} is the computing capacity of the GU$_k$'s CPU.

Assuming that the effective capacitance coefficient of GU$_k$ is q_k, the energy consumption for local computing can be given by

$$E_{k,g}^{\mathrm{local}} = q_k T_{k,g}^{\mathrm{local}} \left(f_k^{\mathrm{local}}\right)^3, \forall k. \tag{13}$$

2) Offloading Computing: If the GU$_k$ offloads the task to the UAV_S$_m$, the offloading process includes three steps as follow.

In the first step, the GU$_k$ sends the task data to the UAV_S$_m$. The delay of transmitting the task L_k^g from the GU$_k$ to the UAV_S$_m$ can be expressed as

$$T_{k,g,m}^{\mathrm{tran}} = \frac{D_g}{W_m R_{k,m}^{\mathrm{sec}}}, \forall k, m, \tag{14}$$

where W_m is the bandwidth of the GU$_k$.

In the second step, the UAV_S$_m$ computes the task data. The computing time for executing the task L_k^g at the UAV_S$_m$ can be expressed as

$$T_{k,g,m}^{\mathrm{comp}} = \frac{D_g C_m}{f_m}, \forall k, m, \tag{15}$$

where C_m is the number of CPU cycles required by the UAV_S$_m$ to compute one bit of task, and f_m is the computing capacity of UAV_S$_m$.

Assuming that the effective capacitance coefficient of UAV_S$_m$ is p_m, the computing energy consumption of UAV_S$_m$ can be expressed as

$$E_{k,g,m}^{\text{comp}} = p_m T_{k,g,m}^{\text{comp}} (f_m)^3, \forall k, m. \tag{16}$$

In the third step, the UAV_S$_m$ sends the data back to the GU$_k$. Since the transmit power of UAV_S is generally much larger than that of GU and the size of task execution results is much smaller than the input data, the delay can be ignored [25].

Therefore, for the offloading computation process, the total execution delay of task L_k^g can be approximated as

$$T_{k,g}^m \approx T_{k,g,m}^{\text{tran}} + T_{k,g,m}^{\text{comp}}, \forall k, m. \tag{17}$$

To sum up, the execution delay of task L_k^g can be given by

$$T_{k,g} = \begin{cases} T_{k,g}^{\text{local}}, & \text{local computing} \\ T_{k,g}^m, & \text{offloading to UAV\_S}_\text{m} \end{cases}, \tag{18}$$

where $T_{k,g}$ should satisfy the constraint $T_{k,g} \leq T_g$. The total delay for all tasks can be denoted as $\sum_{k=1}^{K} T_{k,g}$.

The offloading decision of task L_k^g for the GU$_k$ is represented as $\boldsymbol{a}_g^k = \left[a_0^{g,k}, a_1^{g,k}, a_2^{g,k}, \ldots, a_M^{g,k}\right]$, where $a_0^{g,k} = 1$ means that the GU$_k$ chooses to compute its task locally and $a_m^{g,k} = 1$ means that the GU$_k$ offloads its task to the UAV_S$_m$, which should satisfy the constraint of $\sum_{i=0}^{M} a_i^{g,k} = 1, a_i^{g,k} \in \{0, 1\}$.

III. System Utility Function and Problem Formulation

In this section, the system utility function and the optimization problem are formulated.

In this paper, the system utility includes the delay related utility and the residual energy related utility, which are named as the delay utility and energy utility, respectively.

If the GU$_k$ chooses to accomplish the task locally, the delay utility and energy utility obtained by the local computation can be defined as

$$U_{k,g}^{\text{local,de}} = \frac{T_g - T_{k,g}^{\text{local}}}{T_g}, \tag{19}$$

$$U_{k,g}^{\text{local,en}} = \frac{E_{\max}^g - E_{k,g}^{\text{local}}}{E_{\max}^g}, \tag{20}$$

respectively, where $E_{\max}^g$ is the maximum usable energy of the GU$_k$ with task L_k^g.

Otherwise, if the GU$_k$ offloads the task to the UAV_S$_m$, the delay utility and energy utility obtained by the offloading computation can be defined as

$$U_{k,g,m}^{\text{de}} = \frac{T_g - T_{k,g}^m}{T_g}, \tag{21}$$

$$U_{k,g,m}^{\text{en}} = \frac{E_{\max}^{\text{UAV\_S}} - \sum_{i \in \mathcal{O}_{k,m}} E_{i,g,m}^{\text{comp}}}{E_{\max}^{\text{UAV\_S}}}, \tag{22}$$

respectively, where $E_{\max}^{\text{UAV\_S}}$ is the maximum usable energy of UAV_S$_m$ and $\mathcal{O}_{k,m} = \{1, 2, \ldots, O_{k,m}\}$ is the set of GUs that have offloaded tasks to the UAV_S$_m$ before the GU$_k$'s turn.

The system utility function can be expressed as

$$\begin{aligned} U^{\text{sys}} = &\sum_{k=1}^{K} \left[\theta_g \left(a_0^{g,k} U_{k,g}^{\text{local,de}} + \sum_{m=1}^{M} a_m^{g,k} U_{k,g,m}^{\text{de}} \right) \right] \\ &+ \sum_{k=1}^{K} \left[\eta_g \left(a_0^{g,k} U_{k,g}^{\text{local,en}} + \sum_{m=1}^{M} a_m^{g,k} U_{k,g,m}^{\text{en}} \right) \right], \end{aligned} \tag{23}$$

where θ_g and η_g are the preference of task L_k^g for the delay and residual energy of computing device, respectively.

After solving the deployment problem of UAV_Ss, the objective function is to maximize the system utility by optimizing the offloading decision $\boldsymbol{a}_g^k$. The problem can be expressed as

$$\max_{\boldsymbol{a}_g^k} \quad U^{\text{sys}} \tag{24}$$

$$s.t. \quad \sum_{i=0}^{M} a_i^{g,k} = 1, a_i^{g,k} \in \{0, 1\}, \forall k, \tag{25}$$

$$T_{k,g} \leq T_g, \forall k, , \tag{26}$$

$$R_{k,m}^{\text{sec}} \geq R_{\min}^{\text{sec}}, \forall k, m, \tag{27}$$

$$E_{\max}^{\text{UAV\_S}} \geq \sum_{i \in \mathcal{O}_{k,m}} E_{i,g,m}^{\text{comp}}, \forall k, m, \tag{28}$$

$$E_{\max}^g \geq E_{k,g}^{\text{local}}, \forall k, \tag{29}$$

where the constraint (25) indicates that the task can be either fully calculated locally or completely offloaded to UAV_Ss for computing, the constraint (26) guarantees that the processing delay of task L_k^g cannot exceed its maximum tolerable delay T_g, the constraint (27) ensures that the secure offloading rate cannot be smaller than the minimum secure offloading rate $R_{\min}^{\text{sec}}$, the constraint (28) means that the computing energy consumption for any UAV_S cannot exceed the maximum usable energy $E_{\max}^{\text{UAV\_S}}$, and the constraint (29) ensures that the locally computing energy consumption does not exceed $E_{\max}^g$ for task L_k^g.

The optimization problem of (24) is non-convex, which cannot be solved by conventional methods. Hence, the single-agent and multi-agent schemes based on reinforcement learning are proposed to solve the problem in this paper, which are detailed in the next section.

IV. Single-Agent Scheme Based on Reinforcement Learning

In order to develop the optimal secure offloading scheme, we transform the long-term process of offloading decision into Markov Decision Process (MDP). The four basic elements of MDP are first introduced in this section, and then the single-agent scheme based on the reinforcement learning is proposed to obtain the optimal secure offloading by maximizing the system utility.

A. Basic Elements of MDP

Generally, the MDP model contains four basic elements, including the state, action, state transition probability and reward. In this paper, since the GUs can only get the knowledge

TABLE II
THE RESIDUAL ENERGY RATIO OF UAV_S

$U^{\text{en}}_{k,g,m}$	(0.00, 0.25)	(0.25, 0.50)	(0.50, 0.75)	(0.75, 1.00)
UST_m	1	2	3	4

about the environment state without any prior information, the transition probability of states is unknown. Thus, the process of the task offloading decision is modeled as an MDP without transition probability. The state, action, and reward function in the model are defined as follows.

1) State: Since the system utility is affected by the changeable residual energy of the UAV_Ss, the system state is represented by $S = \{UST_1, UST_2, \ldots, UST_M\}$, where UST_m is the quantitative value of $U^{\text{en}}_{k,g,m}$, which is carried out according to Table II, with a total of four quantization levels [34].

Then, the system utility function in (23) can be converted to

$$U^{\text{sys}'} = \sum_{k=1}^{K}\left[\theta_g\left(a_0^{g,k}U^{\text{local,de}}_{k,g} + \sum_{m=1}^{M} a_m^{g,k}U^{\text{de}}_{k,g,m}\right)\right] + \sum_{k=1}^{K}\left[\eta_g\left(a_0^{g,k}U^{\text{local,en}}_{k,g} + \sum_{m=1}^{M} a_m^{g,k}\frac{UST_m - 1}{3}\right)\right]. \tag{30}$$

The corresponding optimization problem can be expressed as

$$\max_{\boldsymbol{a}_g^k} \quad U^{\text{sys}'} \tag{31}$$

$$s.t. \quad 1 \le UST_m \le 4, \forall k, m. \tag{32}$$

(25)-(27), (29),

2) Action: If the current task is L_k^g, the offloading decision vector $\boldsymbol{a}_g^k$ is utilized to represent the current action.

3) Reward Function: When the current task is L_k^g, the immediate reward earned by the agent after taking action $\boldsymbol{a}_g^k$ can be expressed as

$$r_g^k = \begin{cases} \theta_g U^{\text{local,de}}_{k,g} + \eta_g U^{\text{local,en}}_{k,g} & a_0^{g,k} = 1 \\ \theta_g U^{\text{de}}_{k,g,m} + \eta_g \frac{UST_m - 1}{3}, & a_m^{g,k} = 1 \end{cases}. \tag{33}$$

B. Single-Agent Scheme Based on QL

Q-learning (QL) is one of the representative reinforcement learning algorithms, which uses an agent to establish a Q-table step by step. The agent can take action in the current state according to the expected cumulative discount reward, which is stored in the Q-table. At each step t, the agent will observe the current state s_t and performs the action $\boldsymbol{a}_t$. The environment will feedback an immediate reward $r(s_t, a_t)$ and transform into the next state s_{t+1}. The agent will update its Q-table based on the reward $r(s_t, a_t)$. The Q-value in the Q-table is updated as

$$Q(s_{t+1}, \boldsymbol{a}_{t+1}) = (1-\alpha)\, Q(s_t, \boldsymbol{a}_t) + \alpha\left[r(s_t, \boldsymbol{a}_t) + \gamma \max Q(s_{t+1}, \boldsymbol{a}')\right], \tag{34}$$

where $Q(s_t, \boldsymbol{a}_t)$ and $r(s_t, \boldsymbol{a}_t)$ are the Q-value and immediate reward of the agent when it takes action $\boldsymbol{a}_t$ in the state s_t, respectively. s_{t+1} is the next state after performing the action $\boldsymbol{a}_t$, $\alpha \in [0, 1)$ is the learning rate and $\gamma \in [0, 1)$ is the discount factor.

Then, the single-agent task offloading scheme based on the QL is proposed, which is described in Algorithm 2. First, all the GUs form one agent and their tasks form a queue. At each decision step, the agent explores in the probability ε and utilizes the Q-table in the probability $1 - \varepsilon$. When choosing the action according to the Q-table, the agent always takes the action with the highest Q-value as shown in (34). If the action cannot satisfy the constraints, the agent will choose the action with the highest immediate reward from the actions that meet the constraints. After taking the action, the agent will calculate the next state and the immediate reward, and update the Q-table. The environment will transform into the next state.

Algorithm 2 Single-Agent Task Offloading Algorithm

Input: Learning rate α, discount factor γ, exploration probability ε. The initial value in the Q-table is 0.

repeat
  Reset the initial state s_0.
  while The task queue is not null **do**
    The agent chooses the action using ε-greedy policy. If the action with highest Q-value cannot satisfy the constraints, choose the action with the highest immediate reward instead.
    The agent obtains its immediate reward and calculates the next state s_{t+1}.
    Update the Q-table according to (34).
    Let $s_t \leftarrow s_{t+1}$, remove the first task in the queue.
  end while
until The Q-table converges

Output: The Q-table of agent.

V. MULTI-AGENT SCHEME BASED ON REINFORCEMENT LEARNING

Considering the variety of task types, the multi-agent scheme based on the reinforcement learning is utilized to optimize the secure offloading.

A. Multi-agent Scheme Based on NQL

Nash Q-learning (NQL) is a reinforcement learning algorithm applicable to multiple agents. In the NQL algorithm, the Q-value function for the agent g can be denoted by $Q_g(s, \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_G)$, which is the sum of the immediate reward and the future reward when all the agents adopt the Nash equilibrium strategies. As a result, the Nash Q-function of the agent g can be expressed as

$$Q_g^*(s, \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_G) = r_g(s, \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_G) + \gamma \sum_{s' \in S} p(s'|s, \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_G)\, V_g(s', \pi_1^*, \pi_2^*, \ldots, \pi_G^*), \tag{35}$$

where $r_g(s, \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_G)$ is the immediate reward obtained by the agent g for performing $\boldsymbol{a}_g$ in the state s under the joint actions $(\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_G)$, γ is the discount factor, S is the set of all the environment state, s' is the possible transition state, and $p(s'|s, \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_G)$ is the probability that the

environment state changes to the state s' after the joint actions is performed in the state s. $(\pi_1^*, \pi_2^*, \ldots, \pi_G^*)$ is the joint actions that satisfy the Nash equilibrium, and $V_g(s', \pi_1^*, \pi_2^*, \ldots, \pi_G^*)$ is the total discounted rewards obtained by the agent g in the next state s' under the condition that each agent follow the Nash equilibrium strategy.

The NQL algorithm continuously updates Q-values of the corresponding agent according to the feedback of the environment through the continuous interaction between multiple agents and the environment. After observing the state of the current state s, the system performs the joint actions $(\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_G)$ according to the state. After that, the state of the environment changes to its transition state s', and each agent needs to observe the joint actions and rewards of all the agents. Then, the agent g updates the Q-values in the Q-table by

$$Q_{t+1}^g(s, \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_G) = (1-\alpha) Q_t^g(s, \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_G) + \alpha \left[r_t^g(s, \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_G) + \gamma NashQ_t^g(s') \right], \tag{36}$$

where α is the learning rate, and the $NashQ_t^g(s')$ is the Nash equilibrium reward of the agent g in the state s', which can be defined as

$$NashQ_t^g(s') = \pi_1(s') \cdots \pi_G(s') \cdot Q_t^g(s'). \tag{37}$$

In order to calculate the $NashQ_t^g(s')$, the agent g needs to know $Q_t^1, \cdots, Q_t^G$ at the same time, which is the reason of building G Q-tables for each agent. Similar as (36), the agent g will update the Q-value about the agent j by

$$Q_{t+1}^j(s, \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_G) = (1-\alpha) Q_t^j(s, \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_G) + \alpha \left[r_t^j(s, \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_G) + \gamma NashQ_t^j(s') \right]. \tag{38}$$

Then, the multi-agent task offloading scheme based on the NQL is proposed, which is detailed in Algorithm 3. In the multi-agent system, the length of the task queue in each agent is the number of GUs with the same type task. The set $\{L_1, L_2, \ldots, L_G\}$ is utilized to denote the length of the agents' queues. The calculation of state in the multi-agent scheme is the same as the single-agent scheme, while the definition of the action and reward is different.

1) Action: If the current task is L_k^g, $\boldsymbol{a}_g^k$ is utilized to represent the current action of the agent g.

2) Reward Function: When the current task is L_k^g, the immediate reward earned by the agent g after taking action $\boldsymbol{a}_g^k$ is expressed as (33).

The multi-agent scheme adopts the ε-greedy strategy in the same way as the single-agent scheme. However, when choosing the joint actions according to the Q-tables, the joint actions that satisfy both of Nash equilibrium and constraint conditions may not be found during the learning process. In this case, we choose the joint actions with the highest immediate reward from the actions that meet the constraints.

B. Complexity and Convergence of Algorithm 3

In this subsection, the complexity and convergence for the proposed multi-agent task offloading algorithm are analyzed.

In the multi-agent system, the state space and action space of each agent are denoted as $|\mathbf{S}|$ and $|\mathbf{A}|$, respectively. There holds $|\mathbf{A}_1| = |\mathbf{A}_2| = \cdots = |\mathbf{A}_G| = |\Lambda|$ in the action set, which means that the space capacity of each Q-table is $|\mathbf{S}| |\mathbf{A}|^G$. As a result, the space complexity of Algorithm 3 is $G |\mathbf{S}| |\mathbf{A}|^G$.

The convergence of Algorithm 3 needs to meet the following two assumptions [34].

Assumptions 1. In Algorithm 3, each agent should visit every possible state and action during the learning process.

Assumptions 2. The joint actions $(\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_G)$ taken by each agent and state s in each step t as well as the learning rate should meet the following two conditions.

Condition 1:

$$\begin{aligned} &0 \le \alpha_t(s, \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_G) < 1 \\ &\sum_{t=0}^{\infty} \alpha_t(s, \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_G) = \infty \\ &\sum_{t=0}^{\infty} \left[\alpha_t(s, \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_G) \right]^2 < \infty \end{aligned} \tag{39}$$

Condition 2:

If $(s^t, \boldsymbol{a}_1^t, \boldsymbol{a}_2^t, \ldots, \boldsymbol{a}_G^t) \neq (s, \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_G)$, $\alpha_t(s, \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_G) = 0$.

TABLE III
PARAMETER SETTINGS VARYING BY TASK TYPES

	Type 1	Type 2	Type 3
Data size (Mbit)	40	15	2
Maximum latency tolerance (s)	0.50	0.20	0.05
Maximum usable energy of GU (J)	25.00	9.38	1.25
Preference for delay	0.8	0.4	0.2
Preference for residual energy	0.1	0.2	0.5

Algorithm 3 Multi-Agent Task Offloading Algorithm

Input: Learning rate α, discount factor γ, exploration probability ε and initial system state s_0. The initial value in each Q-table is 0.

repeat
  Reset the initial state s_0.
  while $(L_1 \neq 0 \ \& \ L_2 \neq 0 \ \& \ \ldots L_G \neq 0)$ **do**
    All agents take joint actions using ε-greedy policy. If cannot find joint actions that satisfy both of Nash equilibrium and constraint conditions, choose the joint actions with the highest immediate reward from the actions that meet the constraints.
    for $g = 1 : G$ **do**
      Agent g calculates the immediate reward.
    end for
    Calculate the next state s_{t+1}
    for $g = 1 : G$ **do**
      Agent g updates the Q-table according to (36).
    end for
    Let $s_t \leftarrow s_{t+1}$, remove the first task in each agent queue.
  end while
until The Q-tables converge

Output: Q-tables of all agents.

TABLE IV
SIMULATION SETTINGS

Parameter	Value
Flight altitude of UAVs	50 m
Transmission bandwidth	40 MHz
Reference channel gain	-40 dB
Noise power	-80 dBm
Transmit power of GJ	1 W
Transmit power of GUs	1 W
CPU frequency of GUs	1 GHz
CPU frequency of UAV_Ss	10 GHz
Coverage radius of UAV_Ss	100 m
CPU cycles/bit for GUs	6000
CPU cycles/bit for UAV_Ss	1000
Effective capacitance coefficient	10^{-25}
Minimum secure offloading rate	5 bps/Hz

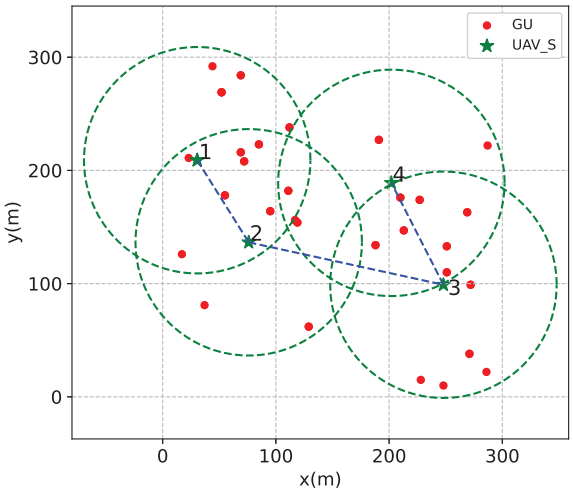


Fig. 3: Results of UAV_S deployment with UAV_S coverage radius R=100 m and 30 GUs.

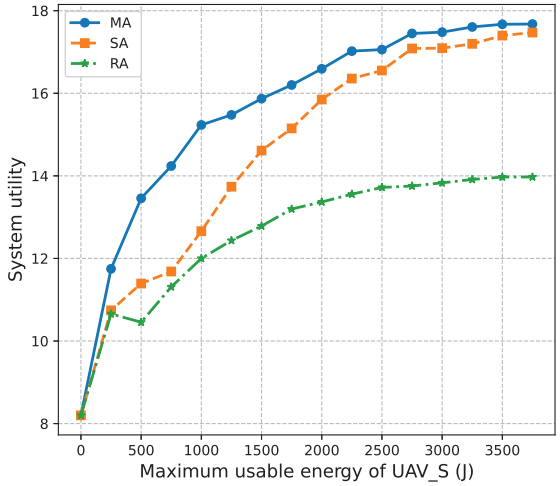


Fig. 4: The system utility versus $E_{\max}^{\text{UAV\_S}}$.

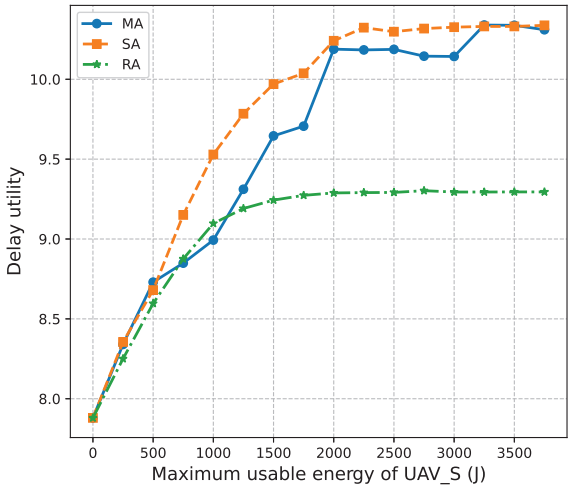


Fig. 5: The delay utility versus $E_{\max}^{\text{UAV\_S}}$.

When the learning rate α_t meets the above conditions, and the mapping $P_t : \mathbb{Q} \to \mathbb{Q}$ satisfies the following conditions, where $\mathbb{Q}$ is the space of all Q-functions.

There is a number $\gamma \in (0,1)$, and a sequence λ_t that tends to 0 with the probability 1, so that $\|P_t Q - P_t Q^*\| \leq \gamma \|Q - Q^*\| + \lambda_t$ is true for $Q^* = E[P_t Q^*]$ and all $Q \in \mathbb{Q}$. Then, the iterative equation $Q_{t+1} = (1-\alpha_t) Q_t + \alpha_t [P_t Q_t]$ converges to Q^* with the probability 1. For multiple agents, it always holds that $\left(Q_t^1, Q_t^2, ..., Q_t^G\right)$ converges to $(Q_1^*, Q_2^*, ..., Q_G^*)$.

Interested readers can refer to [35] for the detailed proof for the convergence of the NQL algorithm, and we omit the details.

VI. SIMULATION RESULTS

In the simulation, 30 GUs are randomly distributed in the area of 300×300 m^2. The coordinates of UAV_E and GJ are fixed at $(131, 114, 50)^T$ and $(131, 114, 0)^T$ in meters, respectively. There are three types of tasks in the multi-agent scheme, and the number of agents G is 3 and each agent has 10 tasks. Parameters that vary by different tasks are shown in Table III, and the other system parameters are shown in Table IV. For the parameters in Algorithm 2 and 3, the learning rate and discount factor is set to 0.01 and 0.8 respectively, the exploration probability is set to decay from 1 to 0.01, in which the step size of the decay is 40 episodes.

Fig. 3 plots the optimization results of the UAV_S placement. UAV_Ss are placed in the numerical order as shown in Fig. 3. The green pentagrams are utilized to indicate the positions of UAV_Ss. In such a case, a total of four UAV_Ss are required and their connecting line (denoted as blue line) looks like a spiral, which is not obvious because of the small number of UAV_Ss.

To prove the superior security computation performance of the proposed singe-agent (SA) scheme and multi-agent (MA) scheme, we compare the performance with the random offloading (RA) scheme in Fig. 4 to Fig. 7, in which each GU randomly selects the action that meets the constraints without learning process.

Fig. 4 shows the system utility versus maximum usable energy of UAV_S $E_{\max}^{\text{UAV\_S}}$. As shown in Fig. 4, it can be found that the MA scheme outperforms the SA and RA schemes. It is because that the MA scheme considers the competition among

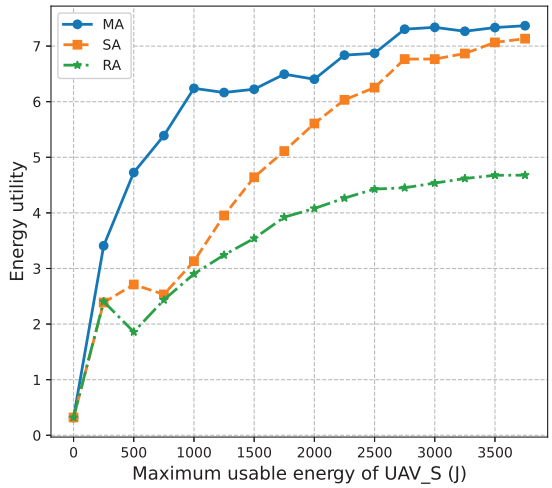
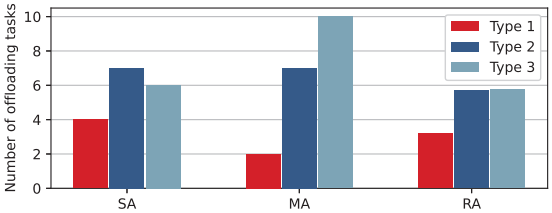
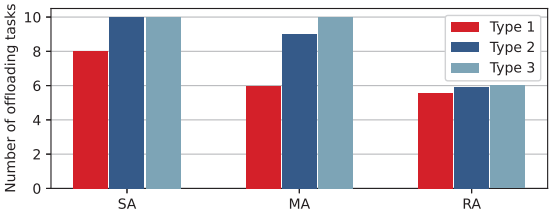


Fig. 6: The energy utility versus $E_{\max}^{\text{UAV\_S}}$.



(a) $E_{\max}^{\text{UAV\_S}} = 750$ J



(b) $E_{\max}^{\text{UAV\_S}} = 1500$ J

(c) $E_{\max}^{\text{UAV\_S}} = 3500$ J

Fig. 7: The number of offloading tasks with different value of $E_{\max}^{\text{UAV\_S}}$.

the three types of tasks, which achieves the optimal offloading strategy in line with Nash equilibrium. We can also find in Fig. 4 that the system utility increases with $E_{\max}^{\text{UAV\_S}}$, which is mainly due to the increase in the number of offloading tasks before $E_{\max}^{\text{UAV\_S}} = 3000$ J. When $E_{\max}^{\text{UAV\_S}}$ is larger than 3000 J, the system utility of the MA and SA schemes tends to be flat. This is because the probability that the task will choose the offloading computation is close to 100% because of the sufficient energy. However, when $E_{\max}^{\text{UAV\_S}}$ is larger than 3000 J, the system utility of the RA scheme barely rises, which can be illustrated from Fig. 7 (c). It is because that the sufficient energy does not make all tasks choose to offload for the RA scheme, but maintains the offloading probability at about 60%.

The delay utility versus maximum usable energy of UAV_S

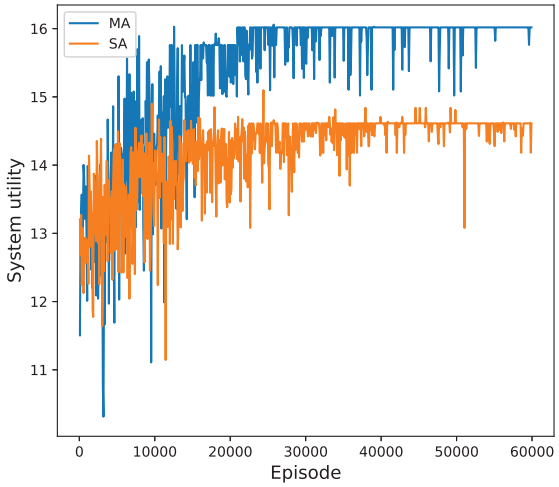


Fig. 8: The system utility during the learning process using the MA and SA schemes.

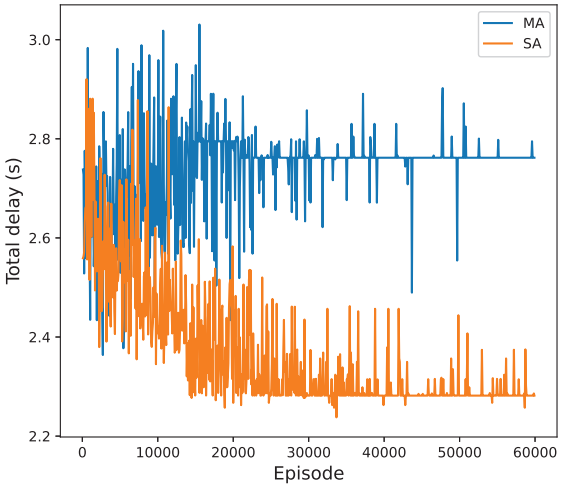
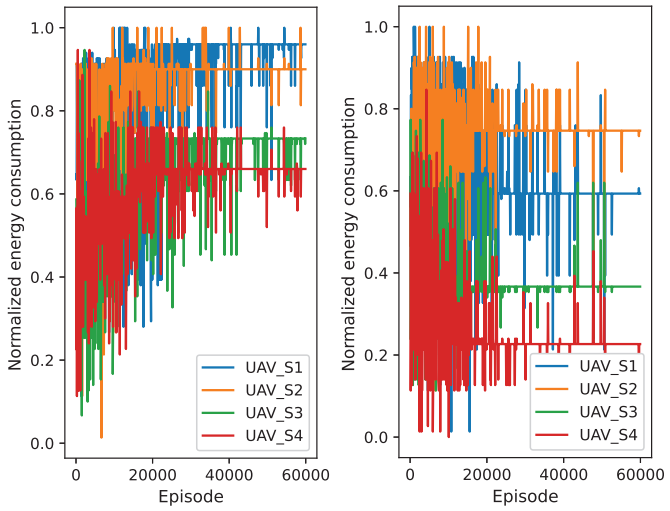


Fig. 9: The total delay during the learning process using the MA and SA schemes.

$E_{\max}^{\text{UAV\_S}}$ is captured in Fig. 5. As we can see, the delay utility obtained by the MA scheme is smaller than the SA scheme, but larger than the RA scheme. This is because when a single agent makes the offloading decision for a task, it often chooses the action with higher delay utility, i.e., the offloading computation. However, when $E_{\max}^{\text{UAV\_S}}$ is larger than 3000 J, almost all tasks in the MA and SA scheme choose for the offloading computation.

Fig. 6 shows the energy utility versus maximum usable energy of UAV_S $E_{\max}^{\text{UAV\_S}}$. In Fig. 6, it can be seen that the energy utility obtained by the MA scheme is always larger than the SA and RA schemes. It is because that in the MA scheme there are fewer tasks choosing to offload computation, resulting in more residual energy.

The number of offloading tasks using different schemes with different value of $E_{\max}^{\text{UAV\_S}}$ is illustrated in Fig. 7. As shown in Fig. 7 (a), it can be found that the number of offloading Type-1 task in the MA scheme is smaller than the SA scheme, and the number of offloading Type-3 task in MA scheme is larger than the SA scheme. It is because the computing energy consumption of Type-1 task is much larger than Type-3 task.



(a) SA scheme (b) MA scheme

Fig. 10: The normalized energy consumption of four UAV_Ss during the learning process using the MA and SA schemes.

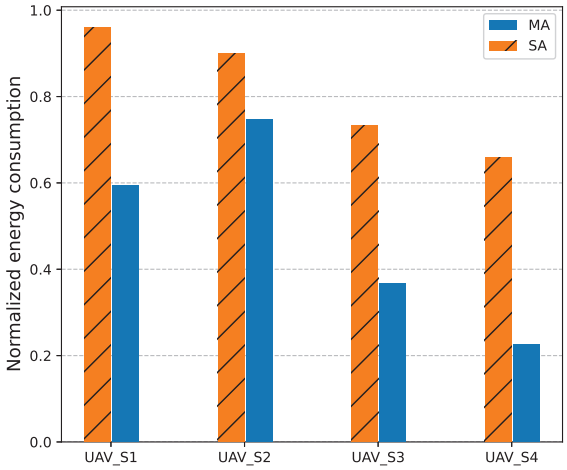


Fig. 11: The final normalized energy consumption using the MA and SA schemes.

Offloading fewer Type-1 tasks is conducive to offload more Type-3 tasks and obtain larger system utility when the energy $E_{\max}^{\text{UAV\_S}}$ is small. In Fig. 7 (b), when $E_{\max}^{\text{UAV\_S}} = 1500$ J, the number of offloading Type-1 and Type-2 tasks in the MA scheme is both smaller than the SA scheme, which will result to larger energy utility. Meanwhile, the number of offloading tasks in the RA scheme gets to the bottleneck. In Fig. 7 (c), all tasks choose to offload to the UAV_S in the MA and SA schemes.

Fig. 8 shows the system utility during the learning process using the MA and SA schemes with the maximum usable energy of UAV_S $E_{\max}^{\text{UAV\_S}} = 1500$ J. In Fig. 8, it can be seen that before the first 20000 episodes, the system utility has a tendency of increases rapidly, accompanied by oscillation. After 20000 episodes, the system utility oscillates within a certain range, which verifies the convergence of the two schemes. The final system utility of the MA scheme is larger than the SA scheme.

Fig. 9 illustrates the total delay during the learning process using the MA and SA schemes. In Fig. 9, with the increase of episodes, the total delay first presents a downward trend and then gradually flattens out. The final total delay of the MA scheme is larger than the SA scheme. It is because the number of offloading tasks in the SA scheme is larger than that in the MA scheme. The delay of offloading computation will always be smaller than locally computing, which results in the delay of the SA scheme smaller than that of the MA scheme. However, the MA scheme achieves better performance than the SA scheme, which can be illustrated from Fig. 6, in which we can find that the energy utility of the MA scheme is much larger than SA scheme.

Fig. 10 shows the normalized energy consumption of the four UAV_Ss during the learning process using the MA and SA schemes. The normalized energy consumption refers to the proportion of the energy consumed by the UAV_S to the maximum usable energy of UAV_S $E_{\max}^{\text{UAV\_S}}$. By comparing the two schemes, it can be found that the energy consumption of UAV_Ss is always larger in the SA scheme due to the larger number of offloading tasks, which will lead to smaller energy utility and larger delay utility. Fig. 11 illustrates the final normalized energy consumption using the MA and SA schemes. We can find that the UAV_Ss placed closely consume similar energy, such as UAV_S_1 and UAV_S_2, which results larger energy utility.

VII. Conclusion

This paper proposes a secure transmission model for the multi-UAV-assisted MEC system, in which the UAV_Ss provide the MEC service to GUs through LoS channels, and the GJ is used to send the jamming signal to interfere the UAV_E. We first utilize the spiral placement algorithm to optimize the deployment of UAV_Ss, which covers all users with the minimum number of UAV_Ss. Then, we formulate a problem to optimize the secure offloading strategy by maximizing the system utility, which considers both the delay and residual energy of the computing device. To solve the problem, we propose the single-agent scheme and the multi-agent scheme based on the reinforcement learning. Simulation results indicate that compared with the single-agent scheme and the random offloading scheme, the multi-agent scheme can achieve better system performance.

References

[1] M. Shafi, A. F. Molisch, P. J. Smith, T. Haustein, P. Zhu, P. De Silva, F. Tufvesson, A. Benjebbour and G. Wunder, "5G: A tutorial overview of standards, trials, challenges, deployment, and practice," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 6, pp. 1201-1221, Jun. 2017.

[2] Z. Xiong, Y. Zhang, D. Niyato, R. Deng, P. Wang and L. Wang, "Deep Reinforcement Learning for Mobile 5G and Beyond: Fundamentals, Applications, and Challenges," *IEEE Veh. Technol. Mag.*, vol. 14, no. 2, pp. 44-52, Jun. 2019.

[3] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo and J. Zhang, "Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738-1762, Aug. 2019.

[4] J. Kang, Z. Xiong, X. Li, Y. Zhang, D. Niyato, C. Leung and C. Miao, "Optimizing Task Assignment for Reliable Blockchain-Empowered Federated Edge Learning," *IEEE Trans. Veh. Technol.*, vol. 70, no. 2, pp. 1910-1923, Feb. 2021.

[5] X. Foukas, G. Patounas, A. Elmokashfi and M. K. Marina, "Network slicing in 5G: Survey and challenges," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 94-100, May 2017.

[6] X. Chen, L. Jiao, W. Li and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795-2808, Oct. 2016.

[7] J. Cao, W. Feng, N. Ge and J. Lu, "Delay characterization of mobile edge computing for 6G time-sensitive services," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3758-3773, Mar. 2021.

[8] H. Ma, Z. Zhou and X. Chen, "Leveraging the Power of Prediction: Predictive Service Placement for Latency-Sensitive Mobile Edge Computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6454-6468, Oct. 2020.

[9] S. Yu, X. Chen, Z. Zhou, X. Gong and D. Wu, "When Deep Reinforcement Learning Meets Federated Learning: Intelligent Multitimescale Resource Management for Multiaccess Edge Computing in 5G Ultradense Network," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2238-2251, Feb. 2021.

[10] X. Pang, M. Sheng, N. Zhao, J. Tang, D. Niyato and K. Wong, "When UAV Meets IRS: Expanding Air-Ground Networks via Passive Reflection," *IEEE Wireless Commun.*, vol. 28, no. 5, pp. 164-170, Mar. 2021.

[11] J. Li, Q. Liu, P. Wu, F. Shu and S. Jin, "Task Offloading for UAV-based Mobile Edge Computing via Deep Reinforcement Learning," *2018 IEEE/CIC International Conference on Communications in China (ICCC)*, 2018, pp. 798-802.

[12] K. Zhang, X. Gui, D. Ren and D. Li, "Energy-Latency Tradeoff for Computation Offloading in UAV-Assisted Multiaccess Edge Computing System," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6709-6719, Apr. 2021.

[13] Y. Liu, K. Xiong, Q. Ni, P. Fan and K. B. Letaief, "UAV-Assisted Wireless Powered Cooperative Mobile Edge Computing: Joint Offloading, CPU Control, and Trajectory Optimization," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 2777-2790, Apr. 2020.

[14] T. Zhang, Y. Xu, J. Loo, D. Yang and L. Xiao, "Joint Computation and Communication Design for UAV-Assisted Mobile Edge Computing in IoT," *IEEE Trans. Industr. Inform.*, vol. 16, no. 8, pp. 5505-5516, Aug. 2020.

[15] L. Zhang and N. Ansari, "Latency-Aware IoT Service Provisioning in UAV-Aided Mobile-Edge Computing Networks," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10573-10580, Oct. 2020.

[16] Y. K. Tun, Y. M. Park, N. H. Tran, W. Saad, S. R. Pandey and C. S. Hong, "Energy-Efficient Resource Management in UAV-Assisted Mobile Edge Computing," *IEEE Commun. Lett.*, vol. 25, no. 1, pp. 249-253, Jan. 2021.

[17] E. E. Haber, H. A. Alameddine, C. Assi and S. Sharafeddine, "UAV-Aided Ultra-Reliable Low-Latency Computation Offloading in Future IoT Networks," *IEEE Trans. Commun.*, vol. 69, no. 10, pp. 6838-6851, Oct. 2021.

[18] Y. Wang, H. Wang and X. Wei, "Energy-Efficient UAV Deployment and Task Scheduling in Multi-UAV Edge Computing," *2020 International Conference on Wireless Communications and Signal Processing (WCSP)*, 2020, pp. 1147-1152.

[19] H. Xiao, Z. Hu, K. Yang, Y. Du and D. Chen, "An Energy-Aware Joint Routing and Task Allocation Algorithm in MEC Systems Assisted by Multiple UAVs," *2020 International Wireless Communications and Mobile Computing (IWCMC)*, 2020, pp. 1654-1659.

[20] L. Yang, H. Yao, J. Wang, C. Jiang, A. Benslimane and Y. Liu, "Multi-UAV-Enabled Load-Balance Mobile-Edge Computing for IoT Networks," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 6898-6908, Aug. 2020.

[21] S. Sun, G. Zhang, H. Mei, K. Wang and K. Yang, "Optimizing Multi-UAV Deployment in 3-D Space to Minimize Task Completion Time in UAV-Enabled Mobile Edge Computing Systems," *IEEE Commun. Lett.*, vol. 25, no. 2, pp. 579-583, Feb. 2021.

[22] Y. Wang, Z. -Y. Ru, K. Wang and P. -Q. Huang, "Joint Deployment and Task Scheduling Optimization for Large-Scale Mobile Users in Multi-UAV-Enabled Mobile Edge Computing," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3984-3997, Sept. 2020.

[23] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam and L. Hanzo, "Multi-Agent Deep Reinforcement Learning-Based Trajectory Planning for Multi-UAV Assisted Mobile Edge Computing," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 1, pp. 73-84, Mar. 2021.

[24] L. Jing, X. Jia, Y. Lv and N. Wan, "Maximizing the Average Secrecy Rate for UAV-assisted MEC: A DRL Method," *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, 2021, pp. 2514-2518.

[25] X. Gu, G. Zhang and J. Gu, "Offloading Optimization for Energy-Minimization Secure UAV-Edge-Computing Systems," *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, 2021, pp. 1-6.

[26] Y. Li, Y. Fang and L. Qiu, "Joint Computation Offloading and Communication Design for Secure UAV-Enabled MEC Systems," *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, 2021, pp. 1-6.

[27] Y. Xu, T. Zhang, D. Yang, Y. Liu and M. Tao, "Joint Resource and Trajectory Optimization for Security in UAV-Assisted MEC Systems," *IEEE Trans. Commun.*, vol. 69, no. 1, pp. 573-588, Jan. 2021.

[28] D. Han and T. Shi, "Secrecy capacity maximization for a UAV-assisted MEC system," *China Commun.*, vol. 17, no. 10, pp. 64-81, Oct. 2020.

[29] 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 Trans. Commun.*, vol. 68, no. 1, pp. 376-388, Jan. 2020.

[30] W. Lu, Y. Ding, Y. Gao, S. Hu, Y. Wu, N. Zhao and Y. Gong, "Resource and trajectory optimization for secure communications in Dual-UAV-MEC systems," *IEEE Trans. Ind. Informat.*, vol. 18, no. 4, pp. 2704-2713, Apr. 2022.

[31] J. Lyu, Y. Zeng, R. Zhang and T. J. Lim, "Placement Optimization of UAV-Mounted Mobile Base Stations," *IEEE Commun. Lett.*, vol. 21, no. 3, pp. 604-607, Mar. 2017.

[32] N. Megiddo, "Linear-time algorithms for linear programming in R3 and related problems," *SIAM J. Comput.*, vol. 12, no. 4, pp. 759-776, Nov. 1983.

[33] J. Elzinga and D. W. Hearn, "Geometrical solutions for some minimax location problems," *Transp. Sci.*, vol. 6, no. 4, pp. 379-394, Nov. 1972.

[34] X. Wang, X. Su and B. Liu, "A Novel Network Selection Approach in 5G Heterogeneous Networks Using Q-Learning," *2019 26th International Conference on Telecommunications (ICT)*, 2019, pp. 309-313.

[35] J. Hu and M. P. Wellman, "Nash Q-Learning for Genera-Sum Stochastic Games," *J. Mach. Learn. Res.*, vol. 4, no. 6, pp. 1039-1069, Aug. 2004.