Hindawi
Security and Communication Networks
Volume 2022, Article ID 4193365, 10 pages
https://doi.org/10.1155/2022/4193365

Research Article

Multiagent Reinforcement Learning for Task Offloading of Space/Aerial-Assisted Edge Computing

Yanlong Li,[1,2,3,4] Lei Liang,[1,2] Jielin Fu ,[1,2] and Junyi Wang[1,2]

[1]*College of Information and Communication, Guilin University of Electronic Technology, Guilin, Guangxi 541004, China*
[2]*Key Laboratory of Cognitive Radio Information Processing of the Ministry of Education, Guilin University of Electronic Technology, Guilin, Guangxi 541004, China*
[3]*Optical Communications Laboratory, Ocean College, Zhejiang University, Zhoushan, Zhejiang 316021, China*
[4]*Ocean Research Center of Zhoushan, Zhejiang University, Zhoushan, Zhejiang 316021, China*

Correspondence should be addressed to Jielin Fu; fujielin@gmail.com

Received 17 December 2021; Revised 14 January 2022; Accepted 21 March 2022; Published 2 May 2022

Academic Editor: Yuyu Yin

The task offloading in space-aerial-ground integrated network (SAGIN) has been envisioned as a challenging issue. In this paper, we investigate a space/aerial-assisted edge computing network architecture considering whether to take advantage of edge server mounted on the unmanned aerial vehicle and satellite for task offloading or not. By optimizing the energy consumption and completion delay, we formulate a NP-hard and non-convex optimization problem to minimize the computation cost, limited by the computation capacity and energy availability constraints. By formulating the problem as a Markov decision process (MDP), we propose a multiagent deep reinforcement learning (MADRL)-based scheme to obtain the optimal task offloading policies considering dynamic computation request and stochastic time-varying channel conditions, while ensuring the quality-of-service requirements. Finally, simulation results demonstrate the task offloading scheme learned from our proposed algorithm that can substantially reduce the average cost as compared to the other three single agent deep reinforcement learning schemes.

1. Introduction

The current in-depth development of fifth generation (5G) and beyond 5G technology is envisioned to build an interconnected world opening up to everyone. The increasing number of various ultradense heterogeneous Internet of things (IoT) devices and the continuous improvement of application requirements have put forward higher requirements for data transmission rate and network coverage [1, 2]. Compared with a fixed terrestrial network, the advantages of versatility, manoeuvrability deployment, as well as seamless coverage make apace-air-ground integrated network (SAGIN) an emerging hot research topic [3, 4]. Meanwhile, mobile edge computing (MEC) is a promising approach to improve the quality of service (QoS) and network performance [5].

Therefore, the MEC technology of the terrestrial network is introduced in SAGIN to provide efficient and flexible computing services by utilizing multilevel and heterogeneous computing resources at the edge of network. Especially, in the case of cellular base station damaging by natural disaster or the case of special senses (e.g., mountainous areas, polar regions, and oceans), UAVs and low earth orbit (LEO) satellite constellation can act as aerial relays or stations, and ground users (GUs) can offload computation tasks for fast processing [6]. In general, cooperative communication by multiple UAVs can be a possible solution to reduce the offloading delay and extend UAVs' service lifetime [7]. However, there introduces more challenging issues to minimizing offloading delay while employing the multiple UAV architecture.

Recently, task offloading has been studied extensively, and task offloading processes are generally modeled as mixed integer programming problems [8], solutions such as heuristic algorithms [9, 10], and convex relaxation [11, 12]. However, these optimization methods require a large

number of iterations to reach a satisfactory local optimum, which makes them unsuitable for real-time offloading decisions when environmental conditions change rapidly and significantly [13, 14]. Meanwhile, deep reinforcement learning (DRL) has been widely used as an effective approach to optimize different problems including offloading policy, which can help overcome the prohibitive computational requirements [15].

The research on task offloading of space/aerial-assisted edge computing has been at its initial stage. In the SAGIN, various single-agent DRL-based task offloading schemes are proposed to maximize the network utility or minimize the computation cost [16]. Considering the limited capacity of MEC server and channel conditions of UAVs, reference [17] proposed a computation offloading scheme based on deep Q-learning network (DQN) to solve the dynamic scheduling problem, and reference [18] adopted a risk-aware reinforcement learning algorithm using actor-critic architecture to minimize the weighted sum of delay and energy consumption. Furthermore, reference [19] proposed a joint resource allocation and task-scheduling methods based on a distributed reinforcement learning algorithm to achieve the optimal partial offloading policy. Reference [20] adopted a deep deterministic policy gradient (DDPG)-based computation offloading scheme to solve high-dimensional state space and continuous action space. Multiagent reinforcement learning (MARL) has been applied in different problems such as path planning [21], dynamic resource allocation [22], and channel access [23]. Compared with single-agent reinforcement learning methods, distributed multiagent systems undoubtedly have better performance. However, the study on task offloading considering the cooperation of space, aerial, and ground multilayer network under multi-UAV multiuser environment is still missing in above research studies. None of above references take full advantage of a possible collaborative framework, but only used multiple parallel deep neural network and decisions are taken independently by each agent of the system. In this paper, a MARL-based method is proposed to solve the cooperative task offloading issue in the space/aerial-assisted edge network. The multiple agents can achieve the offloading optimization collaboratively, in order to reduce the cost of computation tasks. In particular, our main contributions of this work are as follows:

(1) Different from traditional UAV-enabled MEC task offloading scheme, we design a space/aerial assisted edge network for dynamic task offloading in a cooperative environment with multi-UAV.

(2) This paper considers the problem of computation offloading under the SAGIN architecture with the joint communication and computing (C2) service. We formulate the above-mentioned problem as a Markov decision process to minimize the computational cost. We assume each agent shares information with other agents and makes a decision according to current strategies and local real-time observations to select which component of the system to execute.

(3) We propose a multiagent deep deterministic policy gradient (MADDPG)-based task offloading approach. Unlike other DRL algorithms such as Q-learning and DQN which restrict the agent actions to a low-dimensional finite discrete space, the agents in MADDPG can search the best action in an independent consecutive action space and maximize the long-term reward to reduce the computation cost by finding optimal strategy. Furthermore, MADDPG can be decentralized executed once the network has been centralized trained.

The remainder of this paper is organized as follows. In Section 2, the space/serial-assisted edge network architecture and task offloading models are introduced. Section 3 describes the problem formulation and the MADDPG-based solution. The simulation results and analysis are presented in Section 4. Finally, this work is concluded in Section 5.

2. System Model and Problem Formulation

2.1. Network Architecture. As shown in Figure 1, a remote region without cellular coverage is considered; therefore, we provide network access, edge computing, and caching through the aerial segment. We consider a space/aerial-assisted edge computing framework, which consists of N ground users (GUs), I UAVs and, a low earth orbit (LEO) satellite constellation. Figure 1 depicts a multi-UAV, multicomputational node (satellite with remote cloud server, UAV as an aerial MEC server) to provide services to GUs, and let $\mathbf{N} = \{1, 2, \ldots N\}$ be the set of GUs. The SAGIN components that tasks can be offloaded to are denoted by $\mathbf{I} = \{0, 1, 2, \ldots I\}$, where indexes $1, 2, \ldots, I$ and 0 denote UAVs and the LEO satellite constellation, respectively. Considering a discrete time-slotted system with equal slot duration τ. Furthermore, we assume that the overall system has M tasks, denoted by the set $\mathbf{M} = \{1, 2, \ldots, M\}$.The main parameters of this paper are shown in Table 1.

The GU can either execute the computation task locally or offload it to edge server in two ways. Each GU n can determine whether or not to offload its computing task to the edge server k, and let $x_{nmi}(t)$ denote the task offloading decision of task m of GU n. Specifically, $x_{nmi}(t) = 1$ means that the GU n offloads the task m to the edge server k, and $x_{nmi}(t) = 0$ means that the GU n disposes its task locally. There exists constraint (1) indicating the binary constraint of offloading decision:

$$x_{nmi}(t) \in \{0, 1\}. \tag{1}$$

2.2. Computation Model. Without the loss of generality, a tuple (ϕ, γ) is adopted to model the computing tasks from GU devices, where ϕ (in bits) represents the size of computation task, and γ (in CPU cycles per bit) indicates the computing workload means that how many CPU cycles are required to process one bit input data [17]. The delay and energy consumption of downloading can be ignored when the computing results are transmitted back to the GUs by the edge server because the key point of policy is task uploading

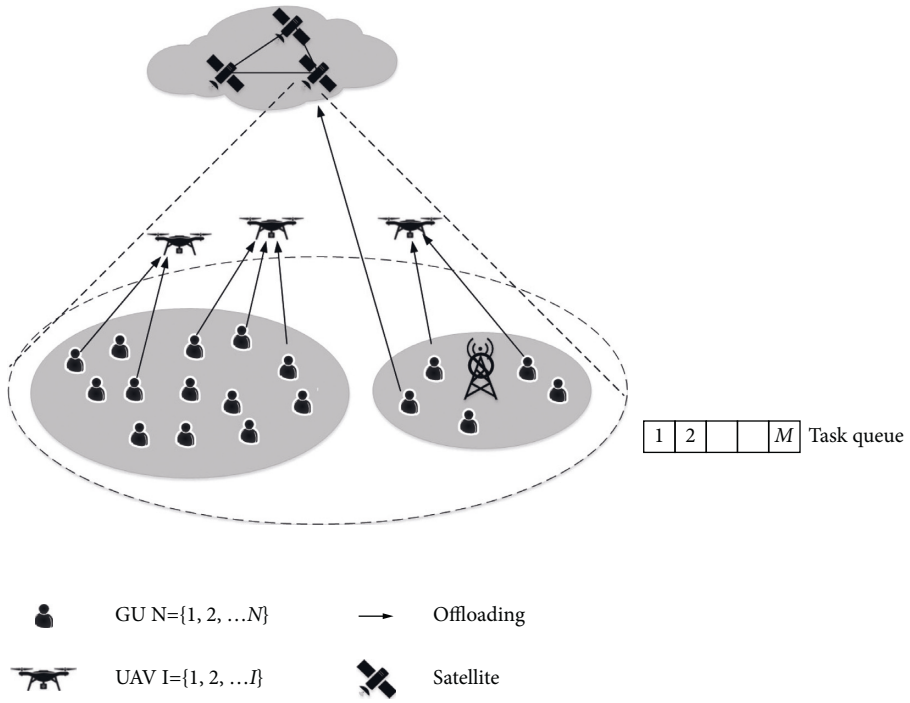

Figure 1: The network model.

Table 1: Notation.

N	Number of GUs
I	Number of UAVs
M	Number of initial offloaded tasks
T	Number of time slot needed to finish offloading
$\mathbf{X}[t] = [x_{nmi}(t)]$	Offloading indication matrix of GUs at time slot t
$f^i, i \neq 0$	Computation Capacity of UAVs
f^0	Computation Capacity of Satellite
f_n	Computation Capacity of GUs
ϕ	Data size (in bits)
γ	Computing workload (in CPU cycles per bit)
$d_n^{\text{e,exec}}, d_n^{\text{l,exec}}$	Computational cost of all SAGIN components
$d_n^{l,wait}$	Queuing time
$\rho_n(t)$	Remained computation tasks at time t
e_n^l	Local execution energy consumption
$P_{n,i}$	Transmission power
B_i	Bandwidth
N_0	Noise power
H	Channel gain of GU to satellite
PL	Path loss of GU n to UAV i
r_n^i	Transmission rate of GU n to UAV i or satellite
$d_{n,i}^{tran}$	Transmission delay
e_n^i	Communication-related energy consumption
ψ	Discount factor
δ	Soft update factor
$\mu(s \mid \theta^\mu)$	Actor online with parameter θ^μ and input s
$\mu'(s \mid \theta^{\mu'})$	Actor target with parameter $\theta^{\mu'}$ and input s
$Q(\mathbf{S}^{all}, \mathbf{A}^{all} \mid \theta^Q)$	Critic online with parameter θ^Q and $(\mathbf{S}^{all}, \mathbf{A}^{all})$
$Q'(\mathbf{S}^{all}, \mathbf{A}^{all} \mid \theta^{Q'})$	Critic target with $\theta^{Q'}$ and $(\mathbf{S}^{all}, \mathbf{A}^{all})$
Z	Minibatch size
$\mathbb{B}$	Replay buffer
$\Delta\mu$	Action random noise

in the considered scenario [9, 24]. In the following, we consider the computation overhead in terms of completion delay and energy consumption for edge computing and local execution.

2.2.1. Edge Computing Model. The computing capability (in CPU cycles per second, the clock frequency of the CPU chip) of edge servers mounted on UAVs and satellite is denoted by f^i $(i \in \{1, 2, \ldots, I\})$ and f^0, respectively. Consequently, the computational cost of all SAGIN components can be calculated as the following equation:

$$d_n^{e,exec}(t) = \sum_{m=1}^{M_t} \sum_{i=0}^{I} \frac{x_{nmi}(t)\phi\gamma}{f^i}. \tag{2}$$

2.2.2. Local Computing Model. Since the limited computing capability of GUs, we assume that the remaining tasks wait to be processed in the queue. The delay of local processing is the sum of the computation execution time and the queuing time. The local execution time of GU n is given by

$$d_n^{l,exec}(t) = \frac{\sum_{m,i}(1 - x_{nmi}(t))\phi\gamma}{f_n}, \tag{3}$$

while the queuing time is calculated as

$$d_n^{l,wait}(t) = \max\left\{\rho_n(t) - \left\lfloor \frac{f_n\tau}{\phi\gamma} \right\rfloor - \sum_{m=1}^{M} \sum_{i=0}^{I} x_{nmi}(t), 0\right\}\tau, \tag{4}$$

where $\rho_n(t) \in [0, M_{\max}]$ denotes the unaccomplished computation task at the beginning of time slot, $M_{\max}$ is the maximum length of the computing queue, $\lfloor \cdot \rfloor$ denotes the

floor function, and f_n is the computing capability of GU n.The local execution energy consumption is given by

$$e_n^l(t) = \xi_n \cdot \sum_{m,i}(1 - x_{nmi}(t))\phi\gamma f_n^2, \tag{5}$$

where ξ_n denotes the effective switched capacitance for the chip architecture [25]. Clearly, the f_n can be adjusted to achieve the optimum computation time and energy consumption by using the DVFS [8].

2.3. Communication Model. Since UAVs and satellites use different frequency bands to communicate, we suppose that there is no interference between UAVs and satellite in this work [26]. Meanwhile, we neglect the propagation delay from GU devices to the UAVs because we assume that the UAV is sufficiently close to GU devices [27]. The aerial to ground communication channel depends on the altitude, angle of elevation, and type of propagation environment [28]. Based on reference [16], the average path loss of the aerial to ground channel can be defined as

$$PL(r,h) = 20\log\left(\frac{4\pi f_c\left(h^2 + r^2\right)^{1/2}}{c}\right) + P_{LoS}\eta_{LOS} + (1 - P_{LoS})\eta_{NLOS}, \tag{6}$$

where P_{LoS} represents the line of sight (LoS) connection probability between GU n and UAV i, and h, r, η_{LOS}, η_{NLOS} denote the UAV flying altitude, horizontal distance between the UAV and the GU, and the additive loss incurred on top of the free space path loss for line-of-sight and not-line-of-sight links [29], respectively. We set the altitude of the UAV to 10m. f_c denotes the carrier frequency, and c denotes the velocity of light. According to [19], the values of $(\eta_{LoS}, \eta_{NLoS})$ are (0.1, 2.1) in remote area. Adopting the Weibull-based channel model [30], we generate the channel gain when $x_{nm0}(t) \neq 0$, which can be given by=

$$H = \frac{G_{tx}G_{rx}\lambda^2}{(4\pi l_{sat})}10^{-F_{\text{rain}}/10}, \tag{7}$$

where G_{tx} and G_{rx} are antenna gains, F_{rain} denotes the rain attenuation, and l_{sat} denotes the distance between GU and the satellite. Consequently, the data rate denoted by $r_i(t)$ is calculated by

$$r_n^i(t) = \begin{cases} B_i\log_2\left(1 + \dfrac{P_{n,i}(t)\cdot|H|^2}{\sigma_S^2}\right), i = 0, \\ B_i\log_2\left(1 + \dfrac{P_{n,i}(t)\cdot 10^{-(PL/10)}}{\sigma_U^2}\right), i \neq 0, \end{cases} \tag{8}$$

where $P_{n,i}$ indicates the transmission power, B_i denotes the channel bandwidth of the aerial-ground link and the ground-satellite link, indexes $1, 2, \ldots, I$, and 0 denotes the UAV swarm and the LEO satellite constellation, respectively. σ_S and σ_U represent the noise power. In line with mentioned earlier, we can define the transmission delay for task offloading over aerial-assisted computing as

$$d_{n,i}^{tran}(t) = \begin{cases} \sum_{m=1}^{M}\left(\lceil x_{nm0}(t)\rceil d_{sat} + \left(\dfrac{x_{nmi}(t)\phi}{r_n^i(t)}\right)\right), i = 0, \\ \sum_{m=1}^{M}\dfrac{x_{nmi}(t)\phi}{r_n^i(t)}, i \neq 0, \end{cases} \tag{9}$$

where d_{sat} denotes the propagation delay between the LEO satellite and GUs, which cannot be ignored. While the communication-related energy consumption can be defined as

$$e_n^i(t) = P_{n,i}d_n^i(t). \tag{10}$$

2.4. Problem Formulation. In line with the computation model and communication model, the computation cost can be defined as the weighted sum of completion delay and energy consumption for completing all tasks of GU n. Generally, the completion delay is the sum of the local execution time, the queuing time, the computational delay of all SAGIN components, and the transmission delay, which is defined as

$$D_n(t) = d_n^{l,\text{exec}}(t) + d_n^{l,\text{wait}}(t) + d_n^{e,\text{exec}}(t) + \sum_{i=0}^{I} d_{n,i}^{\text{tran}}(t), \tag{11}$$

while the energy consumption is given by

$$E_n(t) = e_n^l(t) + \sum_{i=0}^{I} e_n^i(t), \tag{12}$$

Consequently, the computational cost of GU n in space/aerial-assisted network can be calculated as

$$C_n(t) = \omega_n^1 D_n(t) + \omega_n^2 E_n(t), \tag{13}$$

where ω_n^1 and ω_n^2 denote the weights for the energy consumption and the completion delay, respectively, which can be regarded as the trade-off between delay and energy consumption. We can adjust the weights to meet different user demands by using this form of computational cost. Notably, the weights can be further divided into local execution model and edge computing model to increase diversity among these cases. We formulate the optimization problem in our scenario to minimize the computational cost. Therefore, the optimization problem is given by

$$\begin{aligned} &\min_X \lim_{T\longrightarrow\infty}\frac{1}{T}\sum_{t=1}^{T}\omega_n^1 D_n(t) + \omega_n^2 E_n(t) \\ &s.t.\ x_{nmi}(t) \in \{0,1\}, \forall n \in \mathbf{N}, \forall m \in \mathbf{M}, \forall i \in \mathbf{I}, \\ &\sum_{i=0}^{I} x_{nmi}(t) \in [0,1], \forall n \in \mathbf{N}, \forall m \in \mathbf{M}, \\ &\sum_{m=0}^{M_t}\sum_{i=0}^{I} x_{nmi}(t) \leq M_{\max}, \forall n \in \mathbf{N}. \end{aligned} \tag{14}$$

3. MARL for Task Offloading

3.1. MARL Framework. Since above optimization problem in (14) is non-convex and NP hard, we adopt a multiagent reinforcement learning approach to achieve the feasible solution. In thissection, we model the formulated optimization problem as a Markov decision process (MDP) [31], and the purpose of action selection is to maximize the reward function. In the space/aerial-assisted network environment, each GU acts as an agent, chooses an action, and then receives the reward at time slot t. The state space, action space, and reward function are described as follows.

3.1.1. State Space. The state $s_n(t) \in \mathbf{S}$ consists of the channel vectors, the task size randomly generated in the time slot t, the unaccomplished task queue, and the remaining energy. These quantities change over time because of the impact of single and combined actions of this system, so we define the state in our scenario as

$$s_n(t) = \left\{h_t^n, \phi_n^{CPR}, \rho_t^n, E_t^n\right\}. \tag{15}$$

3.1.2. Action Space. Based on current state $s_n(t)$ and other agents' experience, each agent is supposed to select its action to schedule the computation tasks. Formally, we define the vector $a_n(\mathrm{t}) = \{x_{nmi}(t), \forall n \in \mathbf{N}, \forall m \in \mathbf{M}, \forall i \in \mathbf{I}\}$ as the binary offloading decision, where $x_{nmi}(t) \in \{0, 1\}$ indicates that GU n whether to offload its task m to the MEC server i or not. The constraints of the problem (14) are considered as the binary offloading decision strategy, and computation is offloaded to at most one node at time slot t.

3.1.3. Reward Function. In line with reinforcement learning, each agent can select its own action in a decentralized execution to maximize the global reward. The agent's choice is based on the reward function, which specifies the goal of the algorithm. With the objective of long-term weighted sum of delay and energy consumption of all tasks, we define a function to minimize the computation cost as follows:

$$R_n(t) = -\left[\lim_{T\longrightarrow\infty} \frac{1}{T}\sum_{t=1}^{T} C_n(t)|s_n(t)\right]. \tag{16}$$

Let π denotes the stationary policy, and a value function is defined to determine the value of reward, which is given by

$$V_n(s,\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \psi R_n\left(s_t^n, a_t^n\right)|s_t^n = s, \pi\right], \tag{17}$$

where $\psi \in [0, 1]$ denotes the discounting factor, which refers to the cumulative utility. The overall reward of all agents at time slot t can be calculated as

$$R(t) = \sum_{n=1}^{N} R_n(t), \forall n \in \mathbf{N}. \tag{18}$$

The main objective is to minimize the computation cost in the space/aerial-assisted network. We denote the group of GUs' optimal strategies as $\pi^* = \{\pi_1^*, \pi_2^*, \ldots, \pi_n^*\}$. We maximize the long-term reward as

$$\boldsymbol{\pi}^* = \arg\max_a \mathbb{E}_\pi\left[\sum_{t=1}^{\infty} \lambda^{t-1} R(t)\right], \tag{19}$$

where the π_n^* can be expressed as

$$V_n\left(s, \pi_n^*, \pi_{-n}^*\right) \le V_n\left(s, \pi_n, \pi_{-n}^*\right), \tag{20}$$

where π_n and $\pi_{-n}^* = \{\pi_1^*, \ldots, \pi_{n-1}^*, \pi_{n+1}^*, \ldots, \pi_N^*\}$ denote the set of all possible strategies taken by GU n and other agents' strategies, respectively. Therefore, the MADRL algorithm can obtain the optimal policy through convergence.

3.2. MADDPG-Based Task Offloading Scheme. In thissection, the MADDPG [32]-based task offloading scheme is proposed to derive the near-optimum decision by optimizing the continuous variable $\mathbf{X}$. MADDPG not only retains the greatest advantages of DDPG that can consider continuous action space but also solves the shortcomings of Q-learning or policy gradient algorithms that are not suitable for the multiagent environment by extending the DDPG algorithm into a multiagent domain.

As shown in Figure 2, the MADDPG framework is carried out by centralized training and distributed execution. Each GU has a critic and an actor as agent n, critic n can choose the appropriate action a_n according to the observation s_n, and actor n would evaluate the action a_n based on the global observation $\mathbf{S}^{all}$. During the training procedure, each actor n collects the policies of other agents and denotes them as $\mathbf{A}^{all}$. In our MADDPG architecture, the actor is trained to generate a deterministic policy, the critic is trained to evaluate the actor, and the experience replay buffer $\mathbb{B}$ is denoted to effectively avoid the correlated action, which can store the minibatches of samples $(s, a, R, s') \sim U(\mathbb{B})$.The random minibatch size of samples can be denoted by Z.

In the MADDPG algorithm, each agent selects an optimal action $a_n^k = \mu(s_n^k|\theta^\mu)$ as the output of actor network μ. The actor network can be updated in the form of gradient, which is given by

$$\nabla_{\theta^\mu} J \approx \mathbb{E}_{(s,a,R,s')\sim U(\mathbf{B})}\left[\left(\nabla_a Q\left(s, a \mid \theta^Q\right)\nabla_{\theta^\mu}\mu\left(s \mid \theta^\mu\right)\right)\right], \tag{21}$$

and the critic network Q is updated as the following expression:

$$L\left(\theta^Q\right) = \mathbb{E}_{\mathbf{S}^{all}, \mathbf{S}^{all'}}\left[\left(Q\left(\mathbf{S}^{all}, \mathbf{A}^{all} \mid \theta^Q\right) - y\right)^2\right], \tag{22}$$

where $y_n = R_t^n + \psi Q_\pi(S^{all'}, A^{all'}|\theta)|_{a_n^{k+1'}=\mu'(s_n^{k+1})}$, $a_n^{k+1'}$ is the predication of the next action in target actor network. To minimize the policy gradient of agent n, the parameters are soft updated for both the target actor and networks as follows:

$$\begin{cases} \theta_n^{\mu'} \leftarrow \delta\theta_n^\mu + (1-\delta)\theta_n^{\mu'} \\ \theta_n^{Q'} \leftarrow \delta\theta_n^Q + (1-\delta)\theta_n^{Q'} \end{cases}, \tag{23}$$

where δ is the forgetting factor.

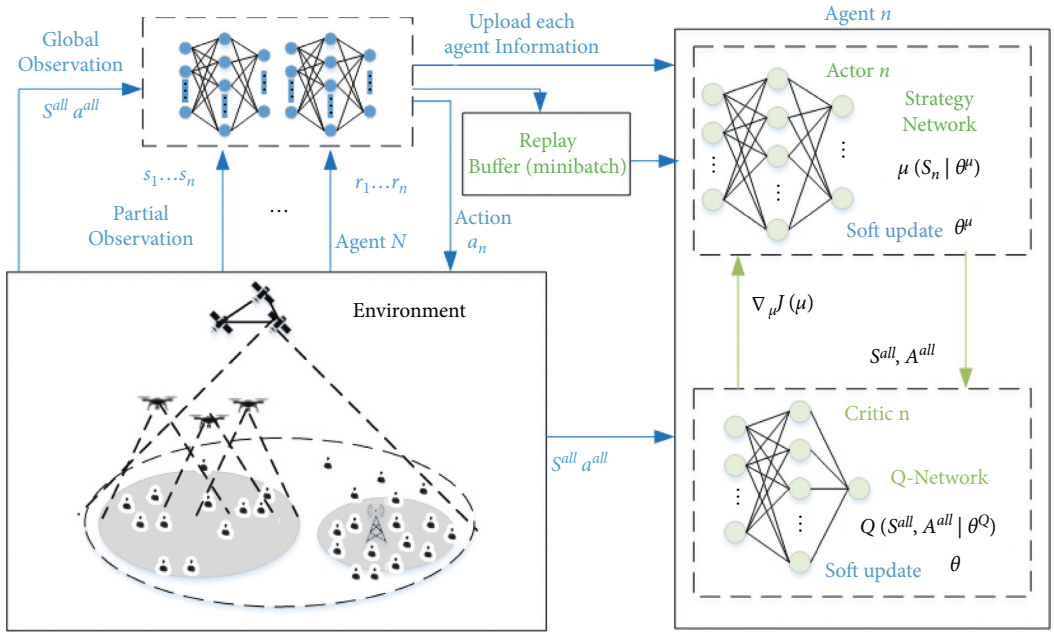

Figure 2: The MADDPG-based task offloading scheme.

The details of MADDPG-based task offloading scheme are shown in Algorithm 1. At first, we initialize four DNNs for each GU, i.e., critic network, actor network, and the two target networks (line 2-3). At the beginning of each episode, each GU obtains its observation state (line 7). Without the loss of generality, we divide each episode into T time slots. For each time slot, agents firstly select an action according to the current policy, and the noise is added into exploration (line 8-9). Afterward, all agents execute their actions, and each agent can receive the corresponding reward and the next state (line 10-11). Then, the experience tuple generated from the above iteration is stored into the replay buffer for parameter update (line 12). Finally, given the sampled minibatch of transitions from the replay buffer, each agent updates the parameter of the critic network by minimizing the loss value, updates the parameter of the actor network by gradient ascent, and updates the parameters of the target networks using (23) (line 15-17).

3.2.1. Analysis of Complexity. The deep neural network of the actor-critic framework can be represented as matrix multiplication. Let N and H define the dimension of output and the number of hidden layers, respectively. We can get the computational complexity between agents $O(N)$, and the complexity of each actor can be expressed as $O(HN^2)$. In our proposed algorithm, the training algorithms can be affected by the agent cost C_n, the number of training episodes K, and batch size Z. Therefore, the computational cost of critic networks and training procedure can be estimated as $O(C_nKN)$ and $O(C_nKZHN^2)$, respectively.

3.2.2. Analysis of Convergence. In the proposed algorithm, the gradient method is adopted to approximate the optimal by updating the weight of target networks. Obviously, the parameters θ_n^{μ} and θ_n^{Q} will converge to a particular value after a finite number of iterations. Therefore, the convergence of our proposed algorithm can be guaranteed. Furthermore, the convergence can be observed through simulations.

4. Simulation Results

4.1. Simulation Settings. In thissection, simulation is carried out to verify the proposed model and algorithm. Specifically, we begin by elaborating on the simulation settings. Afterwards, we present an evaluation on the experiment results. Simulation environment is implemented via Python 3.6 with TensorFlow 2.0 on a personal computer with a AMD R7-4800H CPU. ReLU function is used as the activation function after the fully connected layer, and L2 regularization is used to reduce DNN overfitting. The number of neurons in the two hidden layers are 256 and 128, and we set 2000 and 0.001 to the number of episode and learning rate. Other important constant parameters are listed in Table 2.

4.2. Convergence Analysis. To evaluate the performance of our proposed scheme, we further compare the convergence of three algorithms and the mean computation cost in the system. We adopt the other two benchmark schemes: DDPG [31] and DQN [18]. We first conducted a series of experiments to determine the optimal values of the hyperparameters used in the algorithm. The selection is based on the performance of the algorithm under different learning rates and discount factors. The convergence performance of the algorithm under different learning rates is shown in Figure 3. We can observe that the convergence speed will be reduced when the learning rate is too small, and generally, if the learning rate is too large, the algorithm will not converge normally.

Initialization:
Randomly initialize critic network $Q(s,a \mid \theta_n^Q)$ and actor $\mu(s,a \mid \theta_n^\mu)$ with weights θ_n^Q and θ_n^μ
Initialize target network Q' and μ' with weights $\theta_n^{Q'} \longleftarrow \theta_n^Q$ and $\theta_n^{\mu'} \longleftarrow \theta_n^\mu$
Empty replay buffer $\mathbb{B}$
for episode $k = 1, 2, \ldots, K$ **do**
  Initialize a Gaussian noise $\Delta\mu$ with mean = 0;
  Receive initial observation state $\mathbf{S} = \{s_1, s_2, \ldots, s_N\}$;
  for time slot $t = 1, 2, \ldots, T$ **do**
    Select action $a_n(t) = \mu(s_t^n \mid \theta_n^\mu) + \Delta\mu$ according to the current policy and exploration noise $\Delta\mu$
    Execute action $a_n(t)$ and observe the reward R_n, and the next state $s_n(t+1)$
    Collect the global state $\mathbf{S}^{all}$, S'^{all} and the action $\mathbf{A}^{all}$;
    Store transition $(\mathbf{S}^{all}, \mathbf{A}^{all}, R_n(t), S'^{all})$ in $\mathbb{B}$;
    Sample a random mini-batch of transitions $\left\{(\mathbf{S}^{\text{all}}, \mathbf{A}^{\text{all}}, R_n(t), S'^{\text{all}})\right\}_{z=1}^{Z}$ from $\mathbb{B}$;
    Set $y_n = R_t^n + \psi Q_\pi(S^{\text{all}\prime}, A^{\text{all}\prime} \mid \theta) \mid_{a_n^{k+1\prime} = \mu'(s_n^{k+1})}$;
    Update the critic network $Q(s,a \mid \theta_n^Q)$ by minimize the loss
    $L(\theta^Q) = 1/Z \sum_{z=1}^{Z} ((y_z - Q_\pi(s_z, \mathbf{A}^{\text{all}} \mid \theta_n^Q))^2)$
    Update the actor policy by using the sampled policy gradient
    $\nabla_{\theta_n^\mu} J \approx 1/Z \sum_{z=1}^{Z} ((\nabla_a Q(\mathbf{S}^{\text{all}}, \mathbf{A}^{\text{all}} \mid \theta_n^Q) \mid_{a_z = \mu(s_z)} \nabla_{\theta_n^\mu} \mu(\mathbf{S}^{\text{all}} \mid \theta_n^\mu)))$;
    Update the target networks for each agent n:
    $\theta_n^{\mu'} \longleftarrow \delta\theta_n^\mu + (1-\delta)\theta_n^{\mu'}$ and $\theta_n^{Q'} \longleftarrow \delta\theta_n^Q + (1-\delta)\theta_n^{Q'}$;
  end
end

Algorithm 1: MADDPG algorithm for task offloading in SAGIN.

Table 2: Simulation parameters.

Parameter	Value
Number of GUs N	5
Number of UAVs I	4
Computation Capacity of UAVs $f^i, i \neq 0$	5 GC/s
Computation Capacity of Satellite f^0	10 GC/s
Computation Capacity of GUs f_n	200 MC/s
Data size ϕ	2 MB
Computing workload γ	25 cycles/bit
Transmission power $P_{n,i}$	[1.5, 5]W
Bandwidth B_i	[2, 3]MHz
Noise power N_0	−100 dBm
Discount factor ψ	0.99
Soft update factor δ	0.001

As shown in Figure 4, we show the convergence of our proposed algorithm on average reward with episodes, where the weight index of time delay $\omega^1 = 0.6$, and energy consumption weight index $\omega^2 = 1 - \omega^1$, and the results are averaged from ten numerical simulations, proving the effectiveness of neural networks. The average award values of MADDPG, DDPG, and DQN increase continuously until convergence. Obviously, the MADDPG algorithm become stable earlier than DDPG and DQN algorithms, and the other two schemes become stable after more than 400 training episodes. Moreover, the average reward of final convergence of the MADDPG algorithm is higher than the other two algorithms. Based on the simulation result, we conclude that the MADDPG algorithm outperforms the benchmark schemes in minimizing the long-term cost, which can reduce the wastage of resource by learn the policy of cooperation and maximize the global reward.

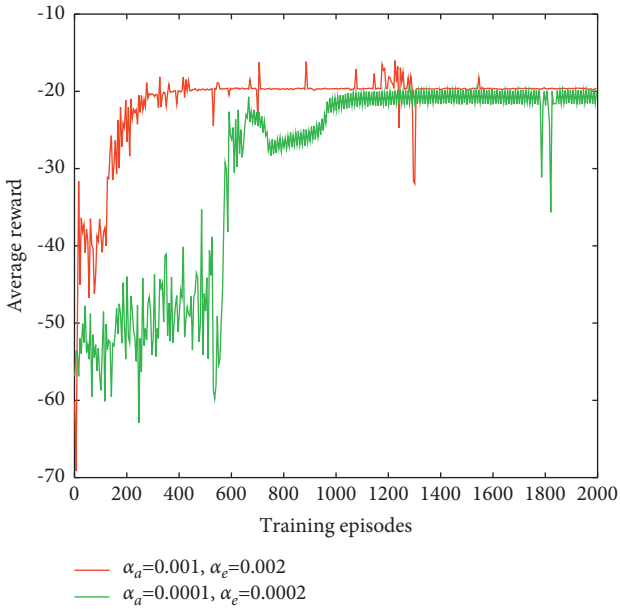

Figure 3: The convergence of the proposed algorithm under different learning states.

4.3. Average Cost. In this section, we discuss the average cost in terms of several number of GUs' devices and the size of offloading tasks. Figure 5 demonstrates the performance of proposed scheme and the three baselines in space/aerial-assisted network to minimize the average computation cost. We provide one other baseline: greedy algorithm [33], and each GU firstly makes its best effort to offload computation tasks, and then the remaining tasks will be processed locally.

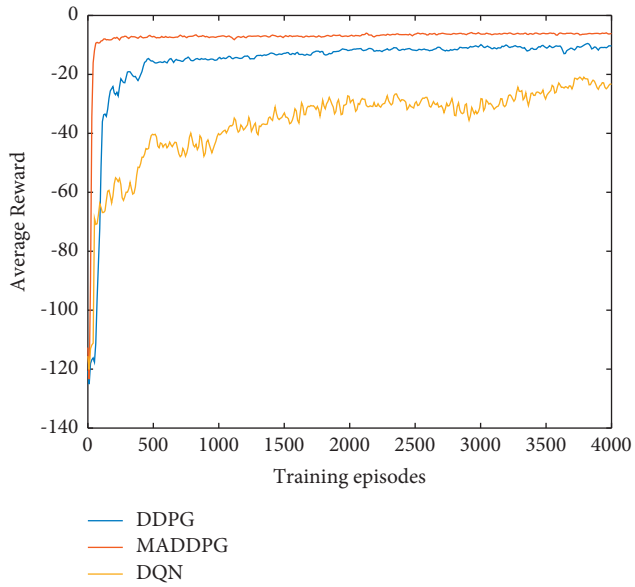

Figure 4: Comparison of the convergence under different algorithms.

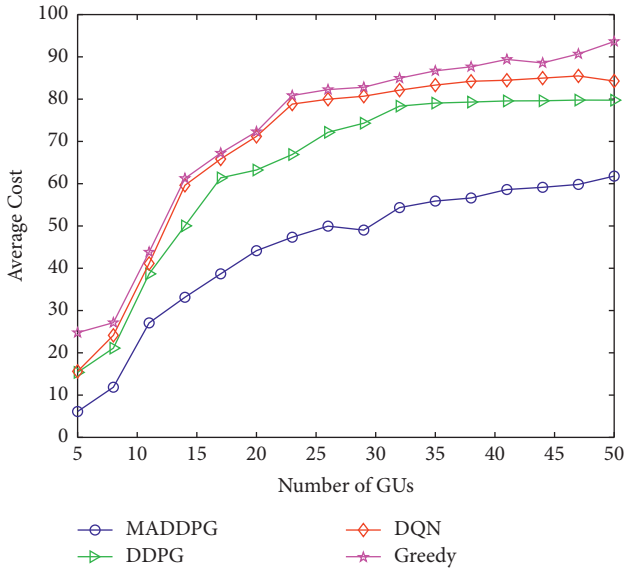

Figure 5: Average cost with number of GUs.

From Figure 5, we can observe that the average computation cost increases as the number of GUs' devices increasing. DQN algorithm, DDPG algorithm, and MADDPG algorithm all use deep reinforcement learning to automatically generate offloading strategies. According to the simulation results, the computation cost is reduced by the MADDPG algorithm, 29.066% compared to the DDPG algorithm, 36.392% compared to the DQN scheme, and 51.602% compared to the greedy scheme. Therefore, the proposed MADDPG scheme can still keep the computation cost lower than benchmark schemes, proving the validity of cooperative policy.

Figure 6 demonstrates the average cost under different sizes of offloading tasks. Obviously, the average cost increases as the size of offloading data size increases. This is because each agent needs to unload a large amount of data, which increases the computational cost of the system. The proposed MADDPG scheme can obtain the best reward to minimize the computation cost than these benchmark schemes. The performance of strategies generated from the DQN algorithm and the DDPG algorithm is general in various scenarios, which is mainly because the training results of the two algorithms are unsuitable in multiagent environment. In contrast, MADDPG can effectively learn stable strategies by decentralized execution and centralized training. Therefore, we conclude that the MADDPG algorithm outperforms the three comparison algorithms in terms of different scenarios.

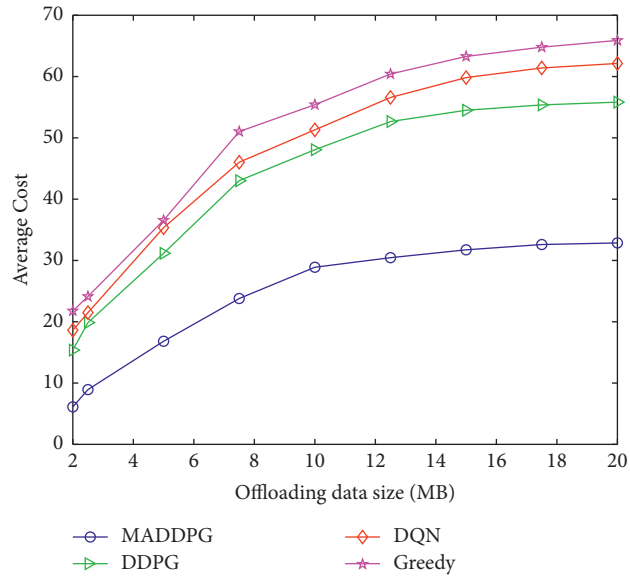

Figure 6: Average cost vs. different size of offloading tasks.

5. Conclusion

In this paper, an efficient task offloading scheme is proposed for the space/aerial-assisted edge computing system in SAGIN. Firstly, we elaborated the SAGIN architecture. Then, we express task offloading as a nonlinear optimization problem with the goal of minimizing the weighted sum of energy consumption and delay. On this basis, we propose an algorithm based on MADDPG to solve this problem. Finally, the simulation results show that the computation cost can be significantly saved by offloading the task to the edge server on the UAV or satellite, and the convergence performance and effectiveness of the proposed scheme in the simplified scenario are also proved by compared with three benchmark schemes.

In the future, we will further consider the mobility management of satellites and UAVs. In addition, the task offloading scheme of SAGIN in areas with rich computing resources is also worth for further study.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under grant (61761014), Guilin University of Electronic Technology Ministry of Education Key Laboratory of Cognitive Radio and Information Processing (CRKL190109).

References

[1] Y. Xu, Y. Wu, H. Gao, S. Song, Y. Yin, and X. Xiao, "Collaborative apis recommendation for artificial intelligence of things with information fusion," *Future Generation Computer Systems*, vol. 125, pp. 471–479, 2021.

[2] H. Gao, Xi Qin, R. J. D. Barroso, W. Hussain, Y. Xu, and Y. Yin, "Collaborative learning-based industrial iot api recommendation for software-defined devices: the implicit knowledge discovery perspective," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 6, no. 1, 2020.

[3] Z. Zhang, Y. Xiao, Z. Ma et al., "6g wireless networks: vision, requirements, architecture, and key technologies," *IEEE Vehicular Technology Magazine*, vol. 14, no. 3, pp. 28–41, 2019.

[4] J. Liu, Y. Shi, Z. M. Fadlullah, and N. Kato, "Space-air-ground integrated network: a survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2714–2741, 2018.

[5] S. S. D. Ali, H. P. Zhao, and H. Kim, "Mobile edge computing: a promising paradigm for future communication systems," in *Proceedings of the TENCON 2018-2018 IEEE Region 10 Conference*, pp. 1183–1187, IEEE, Jeju, Korea (South), October 2018.

[6] N. Cheng, W. Xu, W. Shi et al., "Air-ground integrated mobile edge networks: architecture, challenges, and opportunities," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 26–32, 2018.

[7] A. Gao, Q. Qi, W. Liang, and Z. Ding, "Game Combined Multi-Agent Reinforcement Learning Approach for Uav Assisted Offloading," *IEEE Transactions on Vehicular Technology*, vol. 70, pp. 12888–12901, 2021.

[8] A. Sacco, F. Esposito, M. Guido, and P. Montuschi, "Sustainable Task Offloading in Uav Networks via Multi-Agent Reinforcement Learning," *IEEE Transactions on Vehicular Technology*, vol. 70, 2021.

[9] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Transactions on Wireless Communications*, vol. 17, no. 6, pp. 4177–4190, 2018.

[10] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, 2018.

[11] X. Lyu, W. Ni, H. Tian et al., "Optimal schedule of mobile edge computing for internet of things using partial information," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2606–2615, 2017.

[12] T. Q. Dinh, J. Tang, D. La Quang, and T. Q. S. Quek, "Offloading in mobile edge computing: task allocation and computational frequency scaling," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3571–3584, 2017.

[13] X. Ma, H. Xu, H. Gao, and M. Bian, "Real-time Multiple-Workflow," *Scheduling in Cloud Environment*, vol. 18, no. 4, 2021.

[14] H. Gao, C. Liu, Y. Yin, Y. Xu, and Li Yu, "A hybrid approach to trust node assessment and management for vanets cooperative data communication: historical interaction perspective," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[15] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-based computation offloading for iot devices with energy harvesting," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1930–1941, 2019.

[16] A. M. Seid, G. O. Boateng, B. Mareri, G. Sun, and W. Jiang, "Multi-agent Drl for Task Offloading and Resource Allocation in Multi-Uav Enabled Iot Edge Network," *IEEE Transactions on Network and Service Management*, vol. 18, 2021.

[17] C. Zhou, W. Wu, H. He et al., "Delay-aware iot task scheduling in space-air-ground integrated network," in *Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, Waikoloa, HI, USA, December 2019.

[18] C. Zhou, W. Wu, H. He et al., "Deep Reinforcement Learning for Delay-Oriented Iot Task Scheduling in Space-Air-Ground Integrated Network," 2020, http://arxiv.org/abs/2010.01471.

[19] X. Cheng, F. Lyu, W. Quan et al., "Space/aerial-assisted computing offloading for iot applications: a learning-based approach," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1117–1129, 2019.

[20] Y. Wang, W. Fang, Y. Ding, and N. Xiong, "Computation offloading optimization for uav-assisted mobile edge computing: a deep deterministic policy gradient approach," *Wireless Networks*, vol. 27, no. 4, pp. 2991–3006, 2021.

[21] Q. Han, D. Shi, T. Shen, X. Xu, Li Yuan, and L. Wang, "Joint optimization of multi-uav target assignment and path planning based on multi-agent reinforcement learning," *IEEE Access*, vol. 7, pp. 146264–146272, 2019.

[22] J. Cui, Y. Liu, and A. Nallanathan, "The application of multi-agent reinforcement learning in uav networks," in *Proceedings of the2019 IEEE International Conference on Communications Workshops (ICC Workshops)*, pp. 1–6, IEEE, Shanghai, China, May2019.

[23] Z. Cao, P. Zhou, R. Li, S. Huang, and D. Wu, "Multiagent deep reinforcement learning for joint multichannel access and task offloading of mobile-edge computing in industry 4.0," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6201–6213, 2020.

[24] H. Gao, Y. Zhang, H. Miao, R. J. D. Barroso, and X. Yang, "Sdtioa: modeling the timed privacy requirements of iot service composition: a user interaction perspective for automatic transformation from bpel to timed automata," *Mobile Networks and Applications*, vol. 26, pp. 1–26, 2021.

[25] Q. Tang, Z. Fei, B. Li, and Z. Han, "Computation offloading in leo satellite networks with hybrid cloud and edge computing," *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 9164–9176, 2021.

[26] N. Zhang, H. Liang, N. Cheng, Y. Tang, J. W. Mark, and X. S. Shen, "Dynamic spectrum access in multi-channel cognitive radio networks," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 11, pp. 2053–2064, 2014.

[27] W. Shi, J. Li, W. Xu et al., "Multiple drone-cell deployment analyses and optimization in drone assisted radio access networks," *IEEE Access*, vol. 6, pp. 12518–12529, 2018.

[28] S. Chandrasekharan, K. Gomez, A. Al-Hourani et al., "Designing and implementing future aerial communication

networks," *IEEE Communications Magazine*, vol. 54, no. 5, pp. 26–34, 2016.

[29] A. Al-Hourani, S. Kandeepan, and S. Lardner, "Optimal lap altitude for maximum coverage," *IEEE Wireless Communications Letters*, vol. 3, no. 6, pp. 569–572, 2014.

[30] S. A. Kanellopoulos, C. I. Kourogiorgas, A. D. Panagopoulos, S. N. Livieratos, and G. E. Chatzarakis, "Channel model for satellite communication links above 10ghz based on weibull distribution," *IEEE Communications Letters*, vol. 18, no. 4, pp. 568–571, 2014.

[31] T. P. Lillicrap, J. J. Hunt, P. Alexander et al., "Continuous control with deep reinforcement learning," 2015, http://arxiv.org/abs/1509.02971.

[32] R. Lowe, Yi Wu, Aviv Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," 2017, http://arxiv.org/abs/1706.02275.

[33] F. Wei, S. Chen, and W. Zou, "A greedy algorithm for task offloading in mobile edge computing system," *China Communications*, vol. 15, no. 11, pp. 149–157, 2018.