Hindawi
Mathematical Problems in Engineering
Volume 2022, Article ID 5719190, 13 pages
https://doi.org/10.1155/2022/5719190

Research Article

"Fed-DRL": A Timeliness Optimization Method for Dynamic Data Acquisition System Based on Mobile Edge Computing

Huiji Zheng, Sicong Yu, Xinyuan Qiu, Xiaolong Cui, and Li Zhu

College of Information Engineering, Engineering University of PAP, Xi'an 710086, China

Correspondence should be addressed to Huiji Zheng; 1344261395@qq.com

Received 24 March 2022; Accepted 17 June 2022; Published 31 July 2022

Academic Editor: Qingling Wang

Age of Information (AoI) is a metric to describe the timeliness of a system proposed in recent years. It measures the freshness of the latest received data from the perspective of the target node in the system. This work studies a kind of dynamic data acquisition system for urban security that can update and control the situation of urban environmental security by collecting environmental data. The collected data packets need to be uploaded to the cloud center in time for data update, which has high requirements on the timeliness of the system and freshness of data. However, due to the limited computing capacity of mobile terminals and the pressure of bandwidth for data transmission, problems such as high data execution delay and transmission interruption are caused. Emerging mobile edge computing (MEC), a new model of computing that extends cloud computing capabilities to the edge network, promises to solve these problems. This work focuses on the timeliness of the system, as measured by the average AoI across all mobile terminals. First, a timeliness optimization model is defined, and a multi-agent deep reinforcement learning (DRL) algorithm combined with an attention mechanism is proposed to carry out computing offloading and resource allocation through the continuous interaction between agent and environment; then, in order to improve algorithm performance and data security, the federated learning mode is proposed to train agents; finally, the proposed algorithm is compared with other main baseline algorithms based on deep reinforcement learning. The simulation results show that the proposed algorithm not only outperforms other algorithms in optimizing system timeliness, but also improves the stability of training.

1. Introduction

The concept of MEC originated from the content delivery network (CDN) and was developed in traditional cloud computing. Its key idea is to deploy servers and storage devices at the edge of the network, making the edge of the network also have powerful computation and storage capabilities [1]. Its overall framework can be divided according to its computation capacity and service functions, as shown in Figure 1. There are three layers of cloud, edge, and end, respectively. The cloud layer is mainly composed of large servers with powerful computation resources and super-computation capacity. The edge layer is mainly made up of base stations, edge gateways, edge servers, etc., which is slightly weaker than the cloud layer. The end layer is mainly composed of sensors, mobile devices, etc., which are widely distributed and numerous, and generate a large amount of raw data. However, due to the limited computation and storage capacity of a single device, the computation task often needs to be offloaded to the server. Computing offloading refers to the process in which the terminal device allocates the intensive computation tasks to the server with sufficient computation resources for processing through the wireless channels according to certain strategies, and the server returns the computation results to the terminal devices [2]. The traditional way is to upload to a cloud server. However, with the explosive growth of user data, this method has exposed some shortcomings, namely, high transmission costs and delay issues caused by transmission bandwidth, data processing, and physical distance. Thus, edge computing comes into being, which is a new computation model, but edge computing is not meant to replace cloud computing, but rather to extend cloud computing.

MEC technology can be applied to many promising real-time applications, such as Internet of vehicles, smart city, and smart home. Traditional performance indicators, such

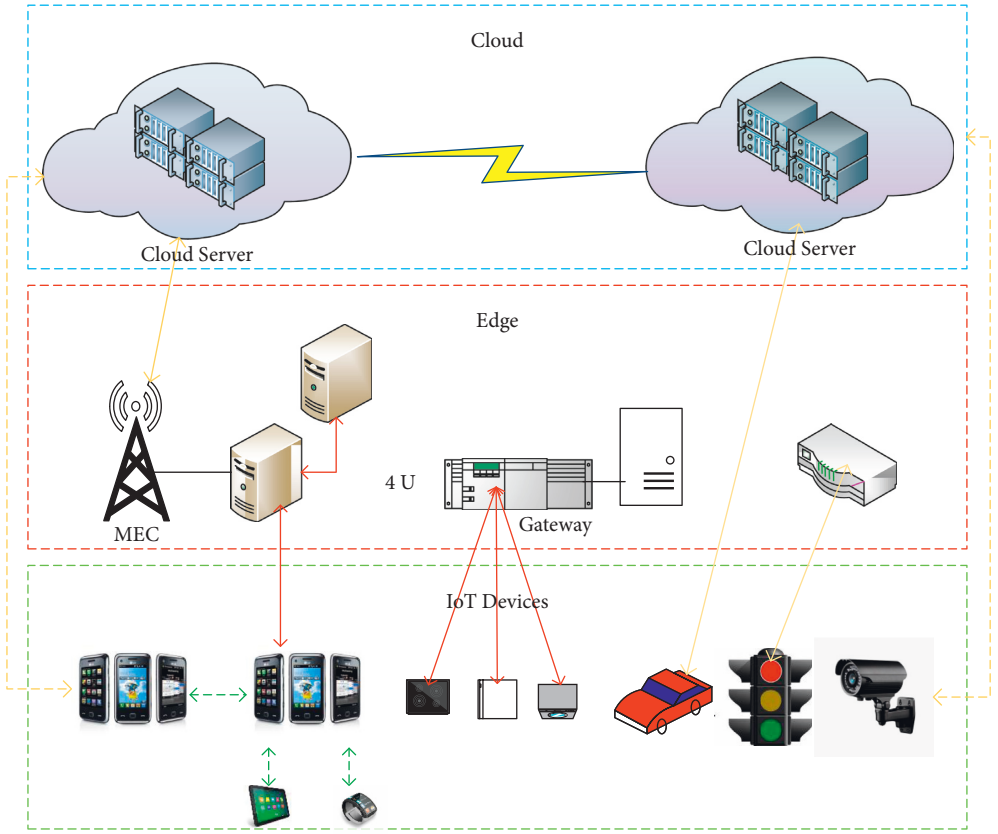

Figure 1: MEC architecture [1].

as latency and throughput, do not fully measure the freshness of the data. On the basis of this, the concept of Age of Information was proposed. It is defined as the time elapsed since the information was generated from the source node at the time of observation. AoI measures the freshness of the data from the perspective of the target node. It should be noted that it takes into account the frequency of the data, which differs from previous indicators [3, 4].

The dynamic data acquisition system for urban security belongs to the state update scenario, which has high requirements on the timeliness of the system and the freshness of the data, the collected data packets need to be uploaded to the cloud center in time, and if the collected data packets are delayed or fail to be sent due to bandwidth or channel, the status update will lose its significance. The main contributions of this work are listed as follows:

(1) AoI is proposed to analyze the timeliness of the dynamic data acquisition system for urban security, which is of great significance for the data system sensitive to timeliness and data freshness.

(2) The optimization problem of minimizing the average AoI of all mobile terminals is proposed and described by the corresponding system model and Markov decision process (MDP) for further research.

(3) A multi-agent deep reinforcement learning algorithm combined with attention mechanism is proposed. Through the continuous interaction between the agent and the environment and the federated learning, agents are trained to perform computing offloading and resource allocation.

2. Related Studies

2.1. Age of Information. As a new indicator, AoI was first proposed by Kaul et al., which is used to measure data freshness from the perspective of target node [5]. Since then, many researchers have carried out relevant studies on AoI from the perspectives of data source update frequency and data generation rate from different application scenarios combined with different queuing models and optimization algorithms. In literature [6], the computation and iteration process of AoI is studied by combining queuing theory, and in literature [7] the influence of service rate on average AoI is studied. Literature [8, 9] compares and analyzes AoI under different computation modes for edge computing scenarios. Literature [10] studies the optimization method of AoI under throughput and energy constraints. The data generation mode is generally random (such as Poisson distribution and exponential distribution), and queuing rules usually include first-come-first-served (FCFS), etc. In literature [11, 12], AoI under different computation modes is analyzed, respectively, for MEC scenarios with single source node and single target node. In the literature [13, 14], data are transmitted through a single-channel wireless network. In literature [15], in a multi-channel symmetric network containing multi-source nodes and a single target node, multiple links are simultaneously scheduled to send real-

time data to optimize the average AoI, and channel conflict in the process of data transmission is considered. The literature [8, 16] considers the energy constraint in wireless sensor networks, constructs the transmission model under the energy constraint as an MDP, and decouples complex scheduling problems. Talak et al. also minimize the average AoI in the network when the channel state is known and unknown, respectively.

At present, researches on AoI optimization mostly focus on single-channel wireless network, and most of them are about the offloading strategies when the data generation time is known. However, due to the complexity of the environment, some algorithms are hard to consider the link between the entity and the environment or estimate the constraints of the environment on the entity; in order to solve this problem, DRL, which combines deep learning with reinforcement learning, is adopted. It can solve high-dimensional state and action space and solve more complex optimization problems.

2.2. Multi-Agent Deep Reinforcement Learning Algorithm. It is difficult to obtain the information needed globally to optimize the timeliness of the system when there are a large number of mobile terminals involved. There are typical policy-based algorithms such as AC and DDPG in DRL, which are used to deal with continuous action space problems. Double neural networks are used to approximate the value function and the policy function, respectively.

2.2.1. Deep Reinforcement Learning. Reinforcement learning is an important branch of machine learning, mainly by modeling the target as an agent, which initially may learn nothing and perform a random action, waiting for the environment to update its status and give it a reward as feedback. Through this reward, the agent can constantly adjust its policy to adapt to the variation in the environment, making the action more scientific, so as to obtain the maximum reward. We are concerned about the agent getting a maximum long-term return. This process is usually described as an MDP, (S, A, R, T, γ), where S is the agent's state, A is the action performed by the agent, R is the reward, T is the probability of environment state transition, and γ is the reward discount factor. The agent policy function represents the mapping from the state space to the action space. The goal of reinforcement learning is to continuously optimize the agent's policy so as to obtain the maximum return, and the policy optimization is usually by optimizing the value function.

Deep learning extracts the features of the original input through deep neural network, so as to have powerful perceptual ability; combining with the learning ability of reinforcement learning, it can be used to deal with decision problems in high-dimensional and complex environments [17, 18]. The DRL is divided mainly into value-based learning and policy-based learning. Value-based learning uses a deep neural network to approximate the value function and generally uses the temporal difference (TD) algorithm or the SARSA (State, Action, Reward, State, Action) algorithm to iterate and update the value network to optimize agent's policy. Similarly, a policy network is established based on policy gradient learning, and the network parameters are updated by the policy gradient. Because the former needs to sample the action, it can only deal with the discrete action space, but the policy gradient-based learning directly uses the policy network to search for actions, so it can deal with the problem of continuous actions. In this work, the actor-critic is a combination of the two algorithms. Actor network uses the policy gradient algorithm to choose the action, and the critic network scores the action. It is noteworthy that the two networks have the same structure but different parameters. During training, the parameters of the two networks are updated alternately.

2.2.2. Multi-Agent Collaboration. In recent years, DRL has been widely used in multi-agent collaboration. Multi-agent collaboration refers to the cooperation of agents with each other to achieve a common goal, to obtain a joint reward [19]. There are two main methods. One direct method is to regard other agents as part of the environment and train each agent individually to maximize the cumulative return of each agent. However, due to the instability of the environment, this method may make it difficult for each agent to completely converge. Another method is to regard all agents as a whole, and its action space is the action set of all agents, but it needs an efficient communication mechanism, which is difficult to achieve. Later, some researchers proposed the multi-agent deep deterministic policy gradient (MADDPG) algorithm. Each agent is independent and generates actions by observing local environment, while the critic network observes global information and optimizes itself by combining information from itself and other agents.

3. System Model

The dynamic data acquisition system is shown in Figure 2; according to the three-level architecture of the MEC network, its end layer is composed of mobile terminals, mobile terminals dynamically collect and preprocess data in a certain area, the edge layer is mainly composed of edge base stations, mainly responsible for data offloaded from mobile terminals to edge base stations, and the cloud layer consists mainly of remote cloud centers that gather all data for storage and analysis.

3.1. Computation Model. We consider an MEC scenario consisting of N mobile terminals and M edge base stations. The edge base stations are vehicle-mounted and move in a certain area. The mobile terminals collect data and offload the data to the appropriate edge base stations. We divide the continuous time into time slots, denoted by an integer t. A list $D_n(t) = [b(t), l(t), x(t)]$ is used to define the data collected by each mobile terminal in the time slot t, where $b(t)$ is the size of data, $l(t)$ is the elapsed time of data, and $x(t)$ is the index of mobile terminal. It is assumed that the collected data are randomly generated with probability p_n^g in each

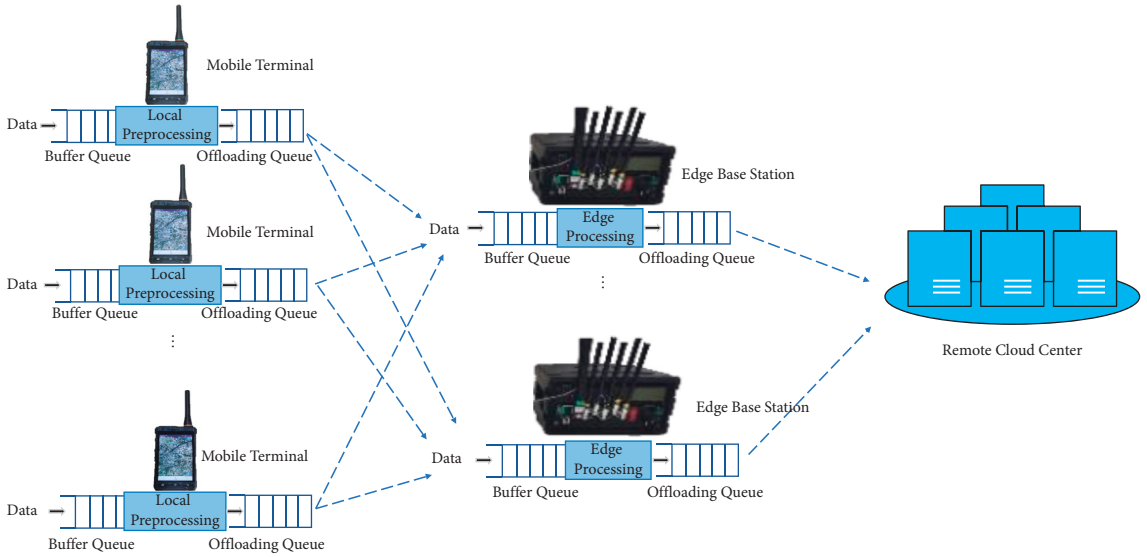

Figure 2: The dynamic data acquisition system.

time slot t, and data size follows a Poisson distribution with arrival rate λ_n. The process can be described as follows:

$$P\{b_n(t) = b\} = p_n^g \frac{e^{-\lambda_n}\lambda_n^b}{b!}. \tag{1}$$

The mobile terminal n has two data queues with capacity B_n^{cac} and B_n^{off}, respectively, as shown in Figure 3: the first one is the cache queue, which mainly stores the latest collected data, waiting for local preprocessing on the mobile terminal, and the other data queue is the offloading queue, which mainly stores processed data, waiting for the mobile terminal to offload to the edge base station. The two data queues adopt first-come first-service (FCFS) principle; that is, in each time slot, the mobile terminal can process only one packet, and only after processing one packet, the next one can be processed. The strategies of the data queue of the mobile terminal in each time slot are denoted as

$$e_n = \left[e_1(t), e_2(t), \ldots, e_{B_n^{cac}}(t)\right], \tag{2}$$

$$o_n = \left[o_1(t), o_2(t), \ldots, o_{B_n^{off}}(t)\right], \tag{3}$$

where $e_i(t) \in \{0,1\}$, $o_i(t) \in \{0,1\}$, and what they satisfy is as follows:

$$\sum_{i=1}^{B_n^{cac}} e_i(t) = 1, \text{for} n = 1, 2, \ldots, N, \tag{4}$$

$$\sum_{i=1}^{B_n^{off}} o_i(t) = 1, \quad \text{for } n = 1, 2, \ldots, N. \tag{5}$$

We can calculate the local processing time of mobile terminal n's data in the time slot t as follows:

$$\tau_n(t) = \frac{\sum b(t)}{f_n}, \tag{6}$$

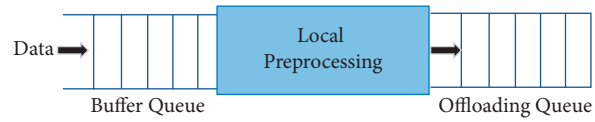

Figure 3: Data queue.

where f_n is the CPU cycle frequency of the mobile terminal, which is used to measure computation capacity.

3.2. Channel Model. Transmission channels between mobile terminals and edge base stations are likely to be obscured by some buildings, resulting in co-existence of line-of-sight (LoS) and nonline-of-sight (NLoS) communication. In this case, the line-of-sight channel model can no longer accurately describe the transmission channel between the mobile terminal and the edge base station, so we adopt the probabilistic line-of-sight channel model [20, 21]. Line-of-sight communication and nonline-of-sight communication have a probability of occurrence, respectively, which is a function of the environment, the density and height of the building, and the elevation angle between the mobile terminal and the edge base station. The probability of line-of-sight communication in the transmission channel can be denoted as

$$P_{LoS} = \frac{1}{1 + C \cdot \exp\left(-B[\theta - C]\right)}, \tag{7}$$

where C and B are constant values determined by the environment (such as the village, the town, the dense city, or other scenario), and θ is the elevation angle of the edge base station relative to the mobile terminal, so the probability of nonline-of-sight communication is $P_{NLoS} = 1 - P_{LoS}$ [22, 23]. Path loss between mobile terminal n and edge base station m can be denoted as

$$\overline{L}_{n,m}(t) = P_{LoS}(t) \cdot L_{n,m}^{LoS}(t) + P_{NLoS}(t) \cdot L_{n,m}^{NLoS}(t), \tag{8}$$

$L_{n,m}^{LoS}(t)$ and $L_{n,m}^{NLoS}(t)$ represent line-of-sight communication and nonline-of-sight communication model, respectively, and are denoted as $L(t) = (4\pi f_c/c)^2 \cdot d^2(t)$, where f_c is the carrier frequency, c is the speed of light, and $d(t)$ is the distance between the mobile terminal and the edge base station. The total channel transmission bandwidth is W, in the time slot t, the bandwidth allocation rate between the mobile terminal and the edge base station is $\varsigma_{n,m}(t)$, $w_{n,m}(t) = \varsigma_{n,m}(t) \cdot W$, and then the transmission rate is denoted as follows:

$$R_{n,m} = w_{n,m}(t)\log_2\left(1 + \frac{P_n(t)}{\bar{L}_{n,m}(t)\eta w_{n,m}(t)}\right), \tag{9}$$

where η denotes the noise power spectral density, $P_n(t)$ denotes the transmitted power, and what they satisfy is as follows:

$$0 \le P_n(t) \le P_n^{\max}(t) \tag{10}$$

3.3. Problem Formulation. The dynamic data acquisition system is very sensitive to timeliness. The traditional indicator of delay is defined as the difference between the receiving time and the sending time; it has a shortage that cannot measure the freshness of data. Data generated earlier may be sent later, and data generated later may be sent first due to channel quality, which is not appropriate for systems that require real-time status update. Thus, in order to optimize the timeliness of the system, we propose AoI to describe the freshness of the data received by the cloud center. Timeliness of the system can reflect the frequency of data collected, processed, and offloaded by mobile terminals in the system. Supposing that at the time slot t, the edge base station m receives the offloading data from the mobile terminal n and the generation time stamp of the data is $\delta(t)$, then we define the AoI as the difference between the two time slots, which can be denoted as

$$\Delta_n(t) = t - \delta(t). \tag{11}$$

According to the first-come first-service (FCFS) principle and assuming $\Delta_n(0) = 0$, the evolution of AoI over time can be derived, as shown in Figure 4, where the peak point of the curve is PAoI.

Our goal is to minimize the average AoI for all mobile terminals, which is denoted as

$$P_1: \min_{\{a_n(t),b(t)\}} \bar{\Delta}(t) = \frac{1}{N}\sum_{n=1}^{N}\Delta_n(t), \tag{12}$$

s.t. equations (1) to (10).

The real environment is complex, so the data in our system are generated randomly, and the transmission channel between the mobile terminal and the edge base station is also attenuated, which results in the strategies of the mobile terminals being different in each time slot. In addition, AoI of mobile terminals is not only related to the current state, but also related to the previous state, and even affects the subsequent state. Therefore, we model this optimization problem as a multi-agent MDP [24]. In the following part, we will discuss the combination of DRL and FL to solve this optimization problem.

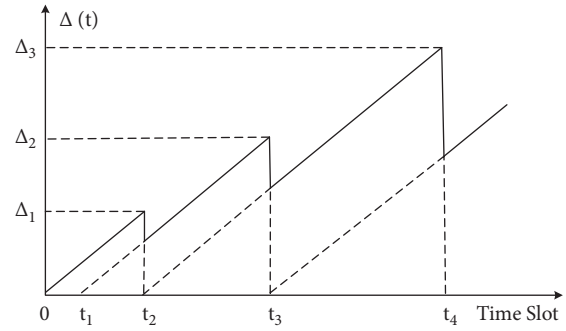

Figure 4: The evolution of AoI over time.

4. Multi-Agent Actor-Critic Algorithm Combined with an Attention Mechanism

In the optimization process of MADDPG algorithm mentioned above, each agent receives the state and action of other agents indiscriminately, resulting in the mixing of beneficial and profitless information, which will reduce the optimization effect. To solve the problem of inaccuracy optimization, we propose a multi-agent actor-critic algorithm with attention mechanism [25]. Combined with the attention mechanism, the agent can pay more attention to the beneficial information when learning the strategies of other agents, instead of using all the information, so as to achieve more accurate optimization.

4.1. MDP Formulation. We deploy models on all agents. First, we analyze their basic elements:

States: In our algorithm, the state S of the mobile terminal agent is denoted as $s = [l, I, B, w]$, where l is the location information of the mobile terminal and the edge base station, I is the data information to be offloaded, B is the state of its two data queues, w is the allocated bandwidth, and the state of the edge base station agent includes the state of all mobile terminals.

Actions: The mobile terminals collect the data, preprocess them locally, and offload them to the edge base station. Thus, its actions include preprocessing and offloading strategy, which can be denoted as $a_n(t) = [\mathrm{cac}_n(t), \mathrm{off}_n(t)]$; the edge base station allocates bandwidth for each mobile terminal, so the action of the edge base station is the set of bandwidth allocation ratio, which can be denoted as $a_m(t) = [\zeta_{1,m}(t), \zeta_{2,m}(t), \ldots, \zeta_{n,m}(t)]$.

Returns: The goal of this work is to minimize the average AoI of all mobile terminals, so the negative of $\bar{\Delta}(t)$ is regarded as the reward of the agent and can be denoted as

$$r_n = -\bar{\Delta}(t) \text{for} n = 1, 2, \ldots, N. \tag{13}$$

To study the global optimization problem of the system, consider a long-term return, denoted as

$$R_n = \sum_{i=0}^{T} \gamma^i r(t+i), \tag{14}$$

where γ is the reward discount factor and T is the length of time.

4.2. Deep Reinforcement Learning Combined with an Attention Mechanism. We hope that the agents can focus on the information beneficial to itself when learning, so the former actor-critic algorithm is improved by combining the attention mechanism, which is a mapping relationship [26, 27]. Agents can determine the importance of other agents to themselves by querying the state and actions of other agents and integrate this information into their own action value estimation function. Taking into account the information of other agents, the value function is denoted as

$$Q(S, A; \theta_n) = f\left[\phi_n(S_n, A_n))\varphi_n\right], \tag{15}$$

where f is the neural network, ϕ_n is an integration function, and φ_n is the coded value of the weight sum of other agents, namely, the contribution of other agents. The formula is described as follows:

$$\begin{aligned} \varphi_n &= \sum_{i \neq n}^{N} \beta_i V_i = \sum_{i \neq n}^{N} \beta_i \cdot L\left[Y \cdot \phi_i(S_i, A_i)\right], \\ \beta_i &\propto \exp\left(e_i^T M_k^T M_p e_n\right), i = \{1, 2, \ldots, N\}, \end{aligned} \tag{16}$$

where e_i is encoded by substituting the action and state of other agents into the integration function. Similarly, the action and state of the agent n can be encoded into e_n. After normalizing the similarity between e_n and e_i, weight β_i is generated through softmax layer. In this process, M_k converts e_i into "key" and M_p converts e_n into "query." The coding value V_i can be obtained by linear transformation of e_i and a shared matrix Y, and then substituting a nonlinear function L.

The actor-critic network is divided into two parts: the actor network is improved by policy gradient and can easily choose the appropriate action in the continuous action space, but because the actor network is round update, resulting in slow learning efficiency. Therefore, we add a value-based critic network to achieve single-step update using the TD algorithm and thus get the actor-critic algorithm. The actor network is the approximation of the policy function, according to the agent's current state to choose action, the critic network is the approximation of the value function, and then the actor network updates the probability of action according to the score, so that the actor network makes it easier to choose the better action.

The updating method of actor network is usually the policy gradient, and the policy gradient of continuous action space is denoted as

$$\frac{\partial}{\partial \theta} V(S; \theta) = E_{A \sim \pi(\cdot|S;\theta)}\left[\frac{\partial}{\partial \theta} \log \pi(A|S; \theta) \cdot Q_\pi(S, A)\right]. \tag{17}$$

But this expectation is difficult to calculate directly, so we use the Monte Carlo approximation to randomly choose an action according to the policy function

$$g(\hat{a}; \theta) = \frac{\partial}{\partial \theta} \log\, \pi(\hat{a}|s; \theta) \cdot q_\pi(s, \hat{a})\Big]. \tag{18}$$

Random selection guarantees unbiasedness. Input the current state of the agent and the randomly selected action into the critic network, and output the action value based on the attention mechanism; then, the actor network can be updated by random gradient:

$$\theta_{t+1} = \theta_t + \beta q(s_t, a_t; \theta_t), \tag{19}$$

where β is the learning rate of the actor network. The critic network is updated by minimizing the loss function of the mean square error (MSE). Firstly, the state and action of the agent are input into the attention mechanism to obtain the weighted sum of the encoded values, and the action value is calculated by (15)

$$l(w) = E\left[\left|q(s_t, a_t; w_t) - y_t\right|^2\right], \tag{20}$$

$$y_t = r_t + \gamma q(s_{t+1}, a_{t+1}; \widetilde{w}_t), \tag{21}$$

where $\widetilde{w}_t$ is the parameter of the target critic network, r_t is the agent's reward, and γ is the reward discount factor.

Both actor and critic networks have corresponding target networks which have the same structure and initialization way as the original network. Its function is mainly to estimate the next action a_{t+1} and score $q(s_{t+1}, a_{t+1}; \widetilde{w}_t)$ according to the next state s_{t+1}. Meanwhile, it improves the stability and convergence of training. It learns once every Tu round and updates through the original network parameters with fixed weight ξ.

As a common method to improve data utilization, experience replay is also used in this algorithm. A complete interaction between an agent and the environment is called transition, which is denoted as a tuple: $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$. All transitions are stored in the experience replay D as training samples. It has a limited capacity and when the capacity is full, new transition will replace the old transition, because the new transition is more valuable. Every time the model is trained, a certain sample is taken from it. To avoid missing the exploration of an action when an agent interacts with the environment, we will randomly select the action with probability ε.

Our framework is shown in Figure 5.

4.3. Train Agents in Federated Mode. We deploy agents on all mobile terminals and edge base stations, not just edge base stations or remote cloud centers, for the following reasons:

Data are generated mainly from the mobile terminal. If it is only deployed in the edge base station or cloud center, it needs to transmit a large amount of data, increasing the transmission pressure of the channel. Furthermore, when data are transmitted with a high secret level, such as location information of the target buildings in the urban environment, there is also the risk of leakage.

The deployment scheme we chose can solve the two problems, but there are also disadvantages: the training data of nonindependent and identical distribution (non-IDD)

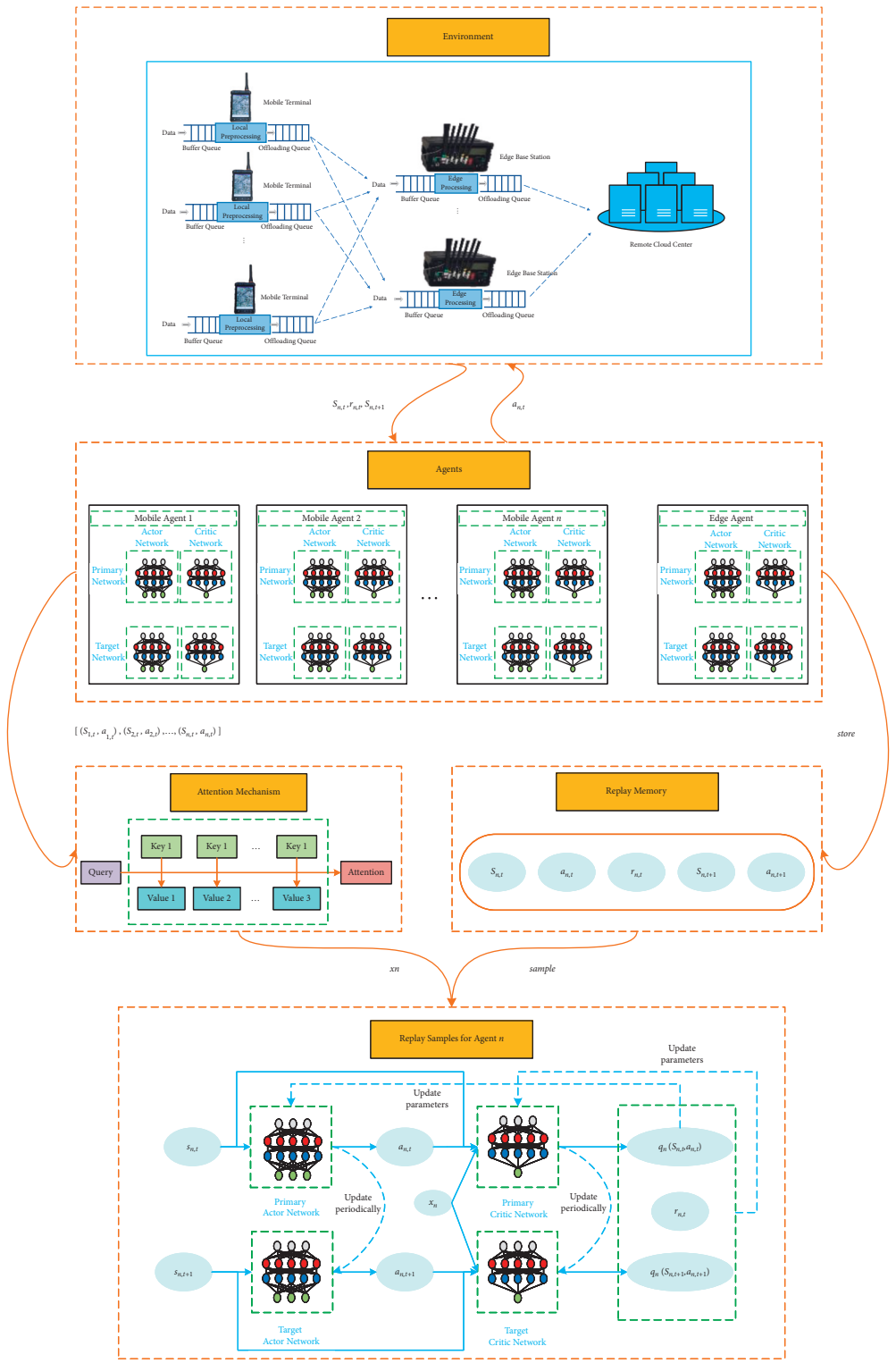

FIGURE 5: Deep reinforcement learning framework with an attention mechanism.

and unbalanced distribution, which is mainly due to training data that do not represent the characteristic of mobile terminals training data and the limited channel bandwidth makes it difficult to ensure that all equipment are online [28]. Therefore, we propose federated learning to train agents, which can avoid transmitting a large number of training data, but only need to share model parameters. It can not only solve the previous shortcomings, but also solve the problems of data privacy and security and reduce the impact of limited communication resources. After each Ef round of learning, all agents share their model parameters and update them. To be specific, the actor and the critic network of each

(1) Algorithm 1 Fed-DRL
(2) Initialization: Initialize system parameters and hyperparameters for learning.
(3) for t = 1,2, …, 5000 do
(4) Reset the environment for each agent, get local state s_t;
(5) Randomly generate $q \in [0, 1]$;
(6) for each agent n in 1,2, …, N do
(7) if $q < \varepsilon$ then
(8) Randomly choose action a_t;
(9) else
(10) Generate actions $a_t \sim \pi(A|S; \theta)$;
(11) end if
(12) end for
(13) The resulting action interacts with environment, generate s_{t+1} and r_t;
(14) Add $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$ of each agent into D
(15) for each agent n in 1,2, …, N do
(16) Sample $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$ from D;
(17) Calculate $q_t(a_t, s_t; w_t)$ using the critic network;
(18) Predict a_{t+1} using the target actor network;
(19) Calculate $q_{t+1}(a_{t+1}, s_{t+1}; w_t)$ using the target critic network;
(20) Update the actor network according to equation (20);
(21) Update the critic network according to equation (21);
(22) end for
(23) if t mod Tu == 1 then
(24) Update the target actor network and the target critic network with following method;
(25) $\theta_{t+1} = \xi\theta_{t+1} + (1 - \xi)\theta_t$
(26) $w_{t+1} = \xi w_{t+1} + (1 - \xi)w_t$
(27) end if
(28) if t mod Ef == 1 then
(29) Run edge-federated updating according to equations (23) and (24);
(30) end if
(31) end for

Algorithm 1: Fed-DRL algorithm.

mobile terminal agent retain parameters of weight σ, respectively, and update them in combination with parameters of other agents. The update mode is denoted as follows:

$$\theta_{t+1} = \theta_t \cdot \Omega, \tag{22}$$

$$w_{t+1} = w_t \cdot \Omega, \tag{23}$$

where θ_t and w_t, respectively, denote the parameters of the actor and the critic network of all mobile terminal agents at the time slot t and Ω is the federated update matrix.

$$\Omega = \begin{bmatrix} \sigma & \frac{1-\sigma}{N-1} & \cdots & \frac{1-\sigma}{N-1} \\ \frac{1-\sigma}{N-1} & \sigma & \cdots & \frac{1-\sigma}{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1-\sigma}{N-1} & \frac{1-\sigma}{N-1} & \cdots & \sigma \end{bmatrix}. \tag{24}$$

The Fed-DRL algorithm [29] proposed in this paper is shown in Algorithm 1:

Table 1: Some experimental parameters.

Parameter	Value	Parameter	Value
λ_n	1 Kb/slot	ε	0.3
B_n^{cac}, B_n^{off}	5	Ef, Tu	8
f_n	2×10^4	D	500
W	100 MHz	B	128
γ	0.8	$P_n^{\max}(t)$	0.2 W
ξ	0.8		

5. Performance Evaluation

In this section, we analyze the performance of the proposed algorithm through simulation experiments and compare it with other baseline algorithms based on deep reinforcement learning.

5.1. Experiment Settings. In this work, experiment simulation was carried out by Python. The experiment was run on a server configured with an Intel Core I7–9700H 3.6 GHz CPU and 8 GB memory. The virtual environment was TensorFlow GPU 2.x. Some experimental parameters are shown in Table 1.

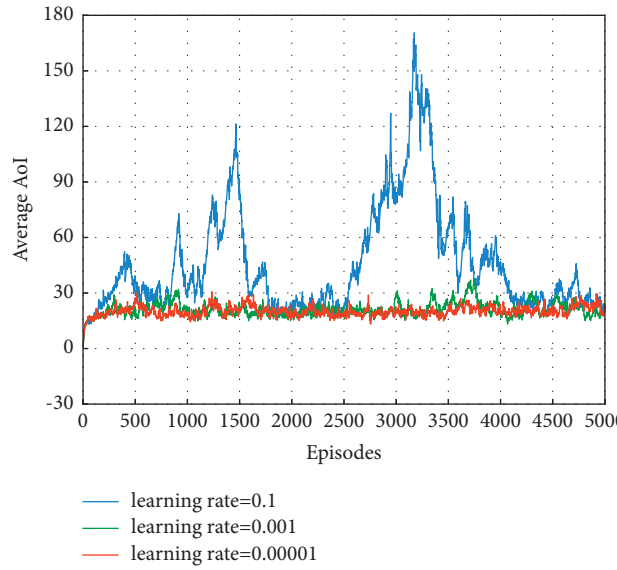

Figure 6: Convergence performance for different learning rates.

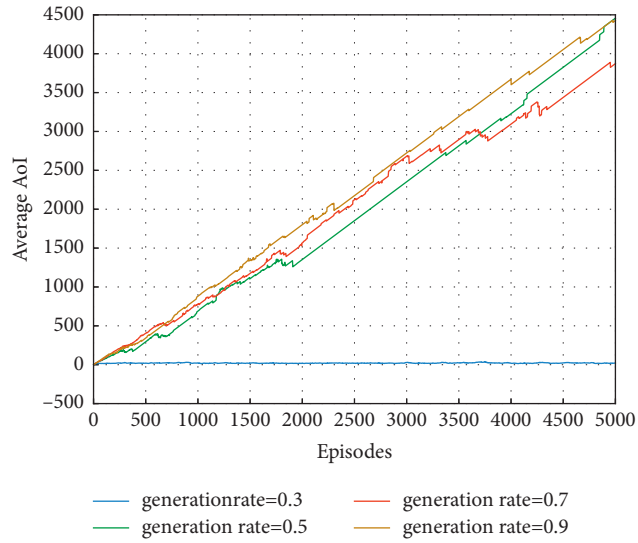

Figure 7: Convergence performance for different data generation rates.

5.2. Simulation Results and Discussion

5.2.1. Parameter Analysis. First, we set different parameters to compare the performance of the algorithm and get the optimal value. Figure 6 shows the convergence performance of the proposed algorithm for different learning rates. We assume that the learning rates of the actor and the critic networks are different. It can be seen that when $\alpha_{\text{Actor}} = 0.1, \alpha_{\text{Critic}} = 0.2$, the proposed algorithm vibrates greatly and is difficult to converge. The main reason is that the large learning rate makes both actor network and critic network take large update steps. When $\alpha_{\text{Actor}} = 0.001, \alpha_{\text{Critic}} = 0.002$ or $\alpha_{\text{Actor}} = 0.00001, \alpha_{\text{Critic}} = 0.00002$, the proposed algorithm can converge. However, when the learning rate is too small, the convergence speed will be very slow, requiring more iteration rounds to converge. Therefore, the best learning rate of the actor network and the critic network is $\alpha_{\text{Actor}} = 0.001$, $\alpha_{\text{Critic}} = 0.002$, respectively.

In Figure 7, we compare the influence of different data generation rates on the convergence performance of the algorithm. The results show that the algorithm performs the best when the data generation rate is 0.3. When it increases to 0.5, it will be difficult for the proposed algorithm to optimize the timeliness of the system, which may be due to the limitations of environmental resources and the system itself. Therefore, in the following experiment, we set the data generation rate to 0.3.

5.2.2. Performance Comparison. Figure 8 shows the comparison results between the proposed algorithm and the baseline algorithm under the same environment settings. In

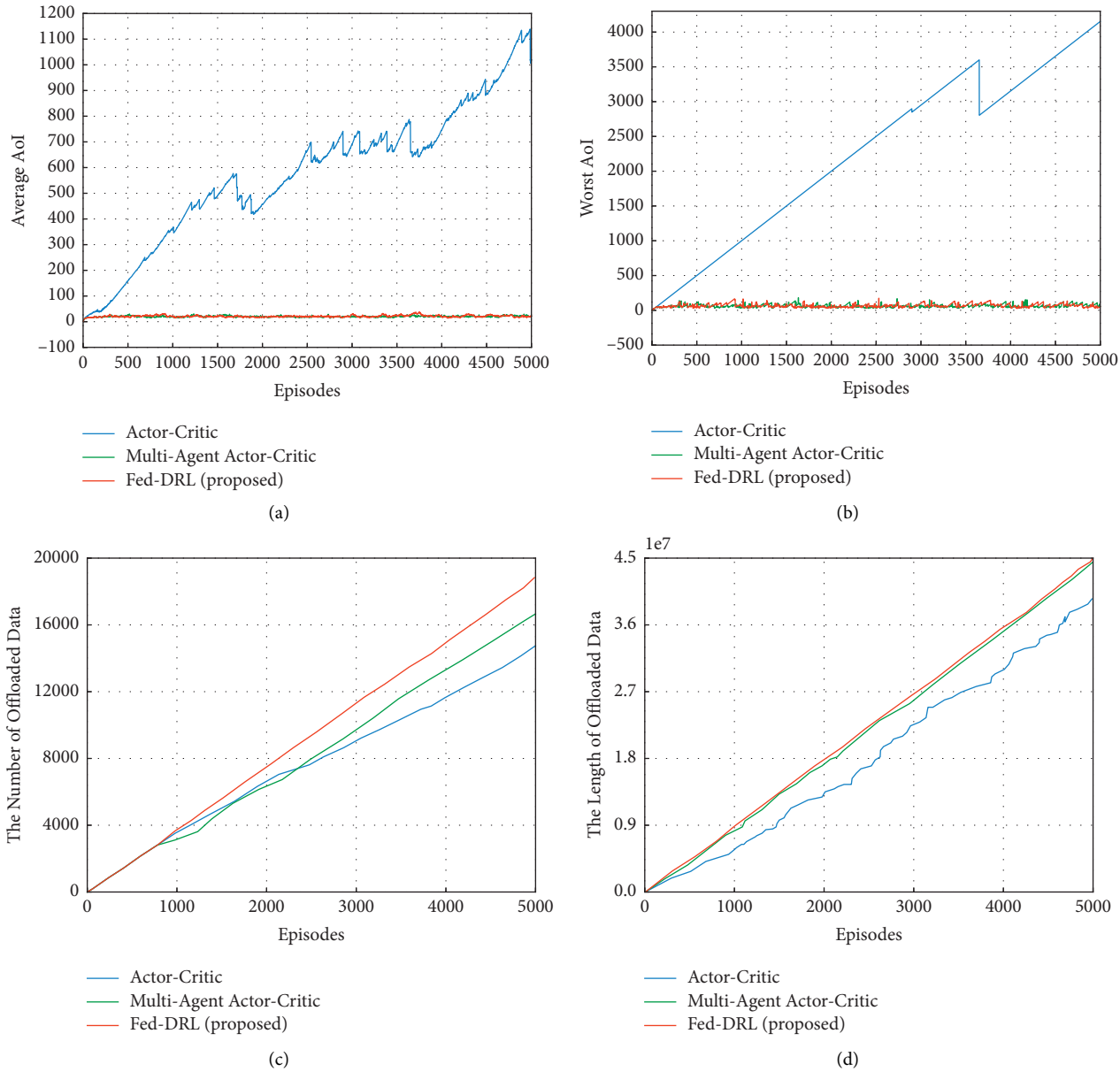

FIGURE 8: Comparison of indicators for different algorithms. (a) Average AoI comparison. (b) Worst AoI comparison. (c) The number of offloading data comparison. (d) The length of offloading data comparison.

Figure 8(a), the horizontal axis is the number of episodes and the vertical axis is the average AoI of the system. It can be seen that the average AoI for the actor-critic algorithm is significantly higher than the results of the other two algorithms, and it is difficult to converge. However, under the multi-agent actor-critic algorithm and Fed-DRL algorithm, the result is very stable and always keeps a low value.

Figure 8(b) shows the worst AoI of the three algorithms, and the Fed-DRL algorithm also performs best. Under the actor-critic algorithm, some resources are ignored for a long time, which may be because the algorithm requires complex neural network to extract the relationship between the global input state and the policy of each agent, which increases the difficulty of training. Furthermore, Figure 8(c) and 8(d), respectively, show the number and length of data offloaded received by the remote cloud center. The results show that, by the proposed algorithm, the data utilization rate is also improved, so that more collected data are uploaded to the cloud center, and the system status is updated in time, thus maintaining the timeliness of the system.

Then, we study the performance of the algorithm under different environment settings. Figure 9 shows the algorithm performance analysis for different numbers of edge base stations and mobile terminals. Figure 9(a) shows the average AoI of the system. It can be seen that the system can basically keep a low average AoI, which is consistent with common sense. In the box plots in Figure 9(b), when $M = 4, N = 60$, the performance is obviously the worst, which may be because the substantial increase in mobile terminals exceeds the service capacity of the edge base stations in a certain area.

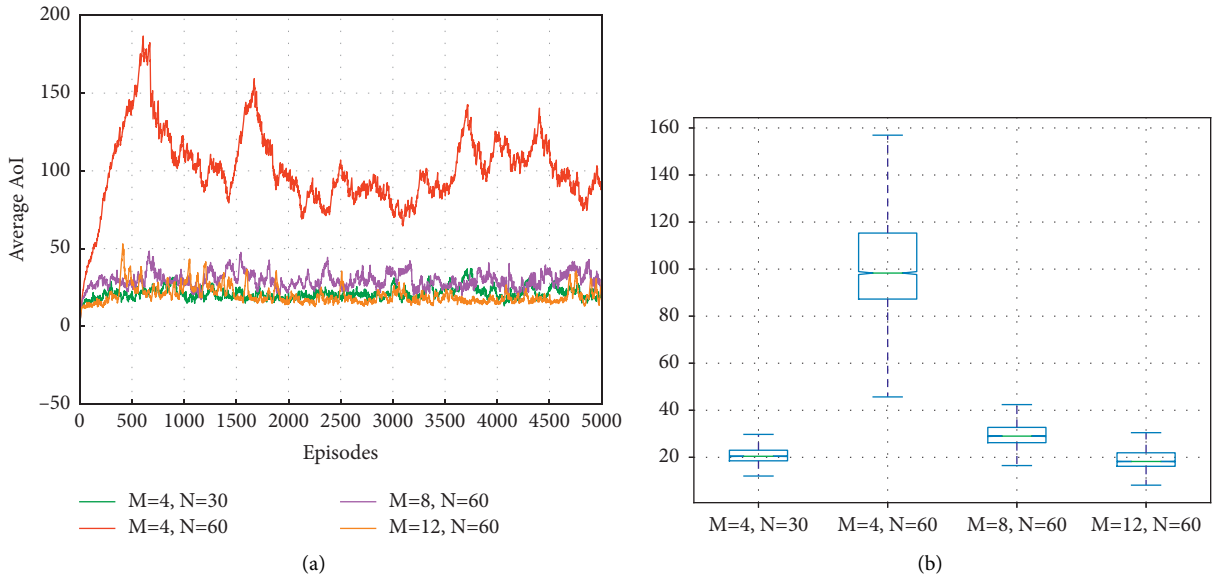

Figure 9: The performance of proposed algorithm for different numbers of mobile terminals and edge base stations. (a) Average AoI for different numbers of mobile terminals and edge base stations. (b) Box plots for different numbers of mobile terminals and edge base stations.

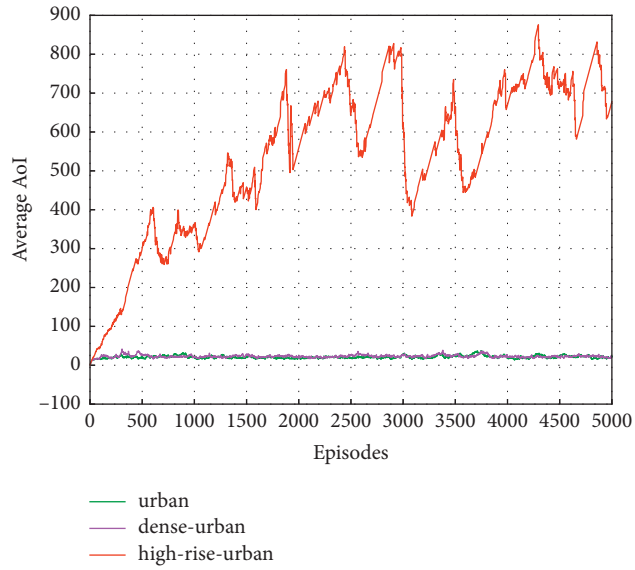

Figure 10: Average AoI for different urban environments.

When the number of edge base stations is increased to 8 and 12, the performance recovery can be even better. As shown in Figure 10, we test the performance of the proposed algorithm in the general urban environment, the dense urban environment, and the high-rise urban environment; the result of the high-rise city environment is not very ideal. The reason may be that the shielding of buildings interferes with the data transmission channel, and it performs well in the general urban environment and dense urban environment, indicating that the proposed algorithm can adapt to the general urban environment.

Finally, we compare the performance under different federated factors σ ($N = 4, M = 30$), as shown in Figure 11; we know that the federated factor measures the weight of each agent's own parameters when the parameters are updated, if σ is too small; intuitively, when $\sigma \geq 0.25 = 1/N$, the diagonal elements of the federated matrix will be smaller than the other elements, so that each agent will not learn much useful knowledge from their own observations, and agent will lose its own individuality. However, when $\sigma = 1$, the update matrix becomes an identity matrix, so that each agent learns the policy independently without parameter sharing, so we mainly study the cases of $\sigma \geq 0.25 = 1/N$, and we also study the case of $\sigma = 0.1$ for comparison. Figure 11(a) is a line plot, and Figure 11(b) is a box plot. It can be seen that when $\sigma = 0.1$, although the system tends to converge at last, its performance is unstable. When $\sigma = 1$, there is no federated update, the performance of the system

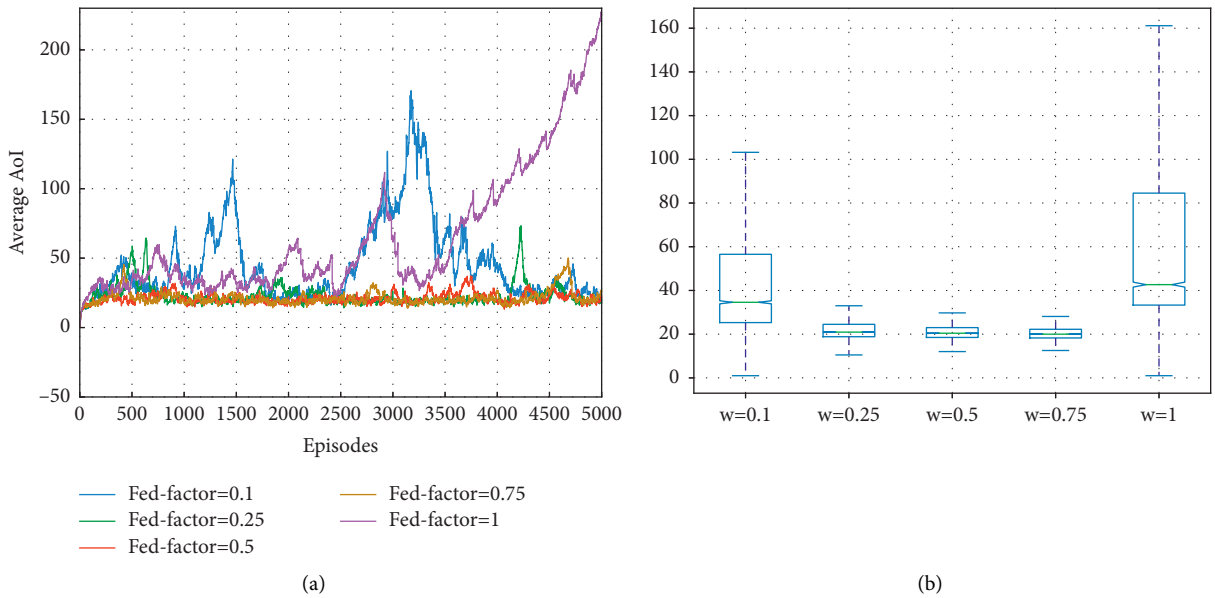

Figure 11: Algorithm performance for different federated factors. (a) Average AoI for different federated factors. (b) Box plots under different federated factors.

is poor and does not converge even after 5000 episodes. In addition, the deviation of the gradient is minimized when $\sigma = 0.25$.

6. Conclusion

The dynamic data acquisition system used for urban security is sensitive to timeliness. To solve this problem, AoI is proposed in this work to measure the freshness of data, so as to optimize the timeliness of system. A multi-agent deep reinforcement learning algorithm combined with attention mechanism is proposed; through the continuous interaction between the agents and the environment, federated learning is used to train the agent for computing offloading and resource allocation. The proposed algorithm is evaluated by simulation experiments. The results show that the proposed algorithm can achieve lower system average AoI and more stable training.

For further work, based on the proposed framework, more operations can be expanded, such as adaptability to the environment and mobility management. At the same time, the rewards in MDP can be flexibly defined, bringing more applications to the MEC systems.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported by the Youth Project of Natural Science Basic Research Program of Shaanxi Province (2021JQ-377) and Young Talent Support Project (2020-JCJQ-QT-050).

References

[1] H. J. Zheng, S. C. Yu, and X. L. Cui, "A survey on computing offloading strategy in edge computing," *Application of Computer Systems*, vol. 30, no. 12, pp. 28–36, 2021.

[2] R. D. Yates and S. Kaul, "Real-time status updating: multiple sources," in *Proceedings of the 2012 IEEE International Symposium on Information Theory Proceedings*, pp. 2666–2670, Cambridge, MA, USA, July 2012.

[3] A. Kosta, N. Pappas, and V. Angelakis, "Age of information: a new concept, metric, and tool," *Foundations and Trends in Networking*, vol. 12, no. 3, pp. 162–259, 2017.

[4] Y.-P. Hsu, E. Modiano, and L. Duan, "Age of information: design and analysis of optimal scheduling algorithms," in *Proceedings of the 2017 IEEE International Symposium on Information Theory (ISIT)*, pp. 561–565, Aachen, Germany, June 2017.

[5] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: how often should one update?" in *Proceedings of the 2012 Proceedings IEEE INFOCOM*, pp. 2731–2735, Orlando, FL, USA, March 2012.

[6] M. Costa, M. Ephremides, and A. Ephremides, "On the Age of information in status update systems with packet management," *IEEE Transactions on Information Theory*, vol. 62, no. 4, pp. 1897–1910, 2016.

[7] Q. Kuang, J. Gong, X. Chen, and X. Ma, "Age-of- information for computation-intensive messages in mobile edge computing," in *Proceedings of the 2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*, pp. 1–6, Xi'an, China, October 2019.

[8] I. Kadota, A. Modiano, and E. Modiano, "Scheduling algorithms for optimizing Age of information in wireless networks with throughput constraints," *IEEE/ACM Transactions on Networking*, vol. 27, no. 4, pp. 1359–1372, 2019.

[9] T. H. Tang, J. Wang, L. Song, and J. Song, "Minimizing Age of information with power constraints: multi-user opportunistic scheduling in multi-state time-varying channels," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 5, pp. 854–868, 2020.

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

[11] J. Pechac and P. Pechac, "Elevation dependent shadowing model for mobile communications via high altitude platforms in built-up areas," *IEEE Transactions on Antennas and Propagation*, vol. 56, no. 4, pp. 1078–1084, 2008.

[12] C. Bennis and M. Bennis, "Taming the tail of maximal information Age in wireless industrial networks," *IEEE Communications Letters*, vol. 23, no. 12, pp. 2442–2446, 2019.

[13] R. Talak, I. Kadota, and S. Karaman, "Scheduling policies for Age minimization in wireless networks with unknown channel state," in *Proceedings of the 2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 2564–2568, Vail, CO, USA, June 2018.

[14] H. Wang, S. X. Duan, and X. Xie, "Scheduling method for multi-channel wireless networks based on optimization of Age of information," *Journal of Electronics and Information Technology*, vol. 44, no. 2, pp. 702–709, 2022.

[15] L. Liu, K. Xiong, and Y. Lu, "Age-constrained energy minimization in UAV-assisted wireless powered sensor networks: a DQN-based approach," in *Proceedings of the Ieee Infocom 2021 - Ieee Conference On Computer Communications Workshops (Infocom Wkshps)*, pp. 1-2, Vancouver, BC, Canada, May 2021.

[16] R. D. Yates, Y. Sun, D. R. Kaul, E. Modiano, and S. Ulukus, "Age of information: an introduction and survey," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 5, pp. 1183–1210, 2021.

[17] L. P. Wan, X. G. Lan, and H. B. Zhang, "A survey on deep reinforcement learning theory and applications," *Pattern Recognition and Artificial Intelligence*, vol. 32, no. 1, pp. 67–81, 2019.

[18] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[19] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," *Machine Learning Proceedings 1994*, pp. 157–163, Morgan Kaufmann, 1994.

[20] V. Mnih and N. Heess, "Recurrent models of visual attention," pp. 2204–2212, 2014, https://arxiv.org/abs/1406.6247.

[21] X. L. Jiao, W. Lou, S. T. Guo, and L. Yang, X. Feng, X. Wang, G. Chen, Delay efficient scheduling algorithms for data aggregation in multi-channel asynchronous duty-cycled WSNs," *IEEE Transactions on Communications*, vol. 67, no. 9, pp. 6179–6192, 2019.

[22] Y. Zeng, J. Zhang, and R. Zhang, "Energy minimization for wireless communication with rotary-wing UAV," *IEEE Transactions on Wireless Communications*, vol. 18, no. 4, pp. 2329–2345, 2019.

[23] M. M. Azari, F. Rosas, and K. C. Chen, S. Pollin, Ultra reliable UAV communication using altitude and cooperation diversity," *IEEE Transactions on Communications*, vol. 66, no. 1, pp. 330–344, 2018.

[24] B. Gu, X. Yang, and Z. Lin, W. Hu, M. Alazab, R. Kharel, Multiagent actor-critic network-based incentive mechanism for mobile crowdsensing in industrial systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 9, pp. 6182–6191, 2021.

[25] C. Sun, X. Wu, X. Li, and Q. Fan, J. Wen, V. C. M. Leung, Cooperative computation offloading for multi-access edge computing in 6G mobile networks via soft actor critic," *IEEE Transactions on Network Science and Engineering*, p. 1, 2021.

[26] A. Vaswani, N. Shazeer, N. Parmar, and J. Uszkoreit, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[27] L. Lin, H. Luo, R. Ye, and M. Ye, "Recurrent models of visual Co-attention for person Re-identification," *IEEE Access*, vol. 7, pp. 8865–8875, 2019.

[28] N. L. S. X. L. Shan and Z. Q. Cui, "Gao FL": "DRL + FL": an intelligent resource allocation model based on deep reinforcement learning for Mobile Edge Computing," *Computer Communications*, vol. 160, pp. 14–24, 2020.

[29] Z. Q. Zhu, S. Wan, and P. Y. Fan, "An edge federated MARL approach for timeliness maintenance in MEC collaboration," in *Proceedings of the 2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, pp. 1–6, Montreal, QC, Canada, June 2021.