Age-of-Information-Based Computation Offloading and Transmission Scheduling in Mobile-Edge-Computing-Enabled IoT Networks

Yingying Jiang, Jia Liu, Iztok Humar, *Senior Member, IEEE*, Min Chen, *Fellow, IEEE*, Salman A. Alqahtani, and M. Shamim Hossain

***Abstract*—The emergence of mobile edge computing (MEC) technology has deployed edge clouds with strong computing capabilities closer to Internet of Thing (IoT) devices, which can effectively meet the demands for computing power and latency. However, in addition to the stringent latency requirements, more and more emerging IoT applications also have higher standards for the freshness and timeliness of collected information. In order to ensure the freshness and high-information value in IoT system, we propose an Age of Information (AoI)-based optimization strategy for computation offloading and transmission scheduling. The strategy considers the AoI during the transmission phase and the execution phase, respectively, under the constraints of delay and remaining energy. Then, a joint optimization model is established based on the comprehensive benefits of AoI and computation rate. To address the strong coupling between the offloading decision and the transmission decision, the original optimization problem is divided into two stages. By the use of the deep deterministic policy gradient (DDPG) algorithm and the dueling double deep *Q* network (D3QN) algorithm, the solution is obtained in terms of the offloading decision and transmission scheduling decision, respectively. The proposed joint optimization strategy considers the impact of the transmission decision on the offloading decision and is adaptable to the dynamic changes in the channel connection between the edge cloud and the user due to user mobility. Experimental results show that compared with other offloading and transmission strategies, the proposed approach has higher overall system revenue and lower AoI.**

***Index Terms*—Age of Information (AoI), dueling double deep *Q* network (D3QN), Internet of Things (IoT), transmission scheduling.**

Manuscript received 13 November 2022; revised 30 March 2023 and 23 May 2023; accepted 24 May 2023. Date of publication 19 June 2023; date of current version 7 November 2023. This work was supported by the Distinguished Scientist Fellowship Program, King Saud University, Riyadh, Saudi Arabia, under Grant DSFP2023. *(Corresponding author: Min Chen.)*

Yingying Jiang and Jia Liu are with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: yingyingjiang@hust.edu.cn).

Iztok Humar is with the Faculty of Electrical Engineering, University of Ljubljana, 1000 Ljubljana, Slovenia (e-mail: iztok.humar@fe.uni-lj.si).

Min Chen is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510640, China, and also with Pazhou Laboratory, Guangzhou 510640, China (e-mail: minchen@ieee.org).

Salman A. Alqahtani is with the Department of Computer Engineering, College of Computer and Information Sciences, King Saud University, Riyadh 11574, Saudi Arabia (e-mail: salmanq@ksu.edu.sa).

M. Shamim Hossain is with the Department of Software Engineering, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia (e-mail: mshossain@ksu.edu.sa).

Digital Object Identifier 10.1109/JIOT.2023.3283287

I. Introduction

With the rapid development of 5G and AI, intelligent Internet of Things (IoT) applications are continuously emerging, such as autonomous driving, smart healthcare, intelligent transportation, Industry 4.0, etc [1], [2], [3]. These applications often have computing requirements with ultralow latency and ultrahigh reliability [4]. Due to the limited computing power and battery capacity of IoT devices, edge servers with sufficient computing power are usually deployed closer to the terminal devices, and computation-intensive and latency-sensitive applications can be offloaded to the edge servers for execution [5]. In traditional 0-1 binary offloading strategies, IoT devices make offloading decisions based on local computing power and wireless channel connection quality, determining whether to execute computing tasks locally or at edge servers [6], [7], in order to minimize latency, energy consumption, or maximize system resource utilization [8]. However, performance metrics, such as latency and throughput cannot fully represent the requirements of latency-sensitive applications for data freshness and timeliness [9]. For example, in an IoT application of autonomous driving, transmitting and processing outdated environmental information can reduce the reliability of data analysis results, leading to erroneous decisions made by vehicles. Therefore, the requirements of data freshness in IoT applications should also be taken into account during offloading decisions.

The data freshness is measured by the Age of Information (AoI), which describes the time interval between the reception of the latest data packet by the information receiver and the time when the packet was generated by the IoT terminal [10]. The larger the AoI, the lower the freshness of the transmitted information in the system. Information with larger AoI can even be outdated and need to be obsoleted. Therefore, the reference value for making decisions in a mobile edge computing (MEC) system is also lower.

Existing work on AoI mainly focus on the timeliness of information during task transmission, while ignore the timeliness of information during task execution. In particular, there is insufficient research on the relevance of information freshness in edge computing environments [11]. In IoT systems with high-real-time requirements, the channel capacity of the edge cloud and bandwidth resource are limited, making it difficult to simultaneously transmit all data collected by IoT

devices to the edge cloud. The congestion of data packets in the transmission queue and the backlog of jobs in the edge cloud further hinder the timely delivery of data packets and their updating [12]. Therefore, an efficient transmission scheduling strategy is critical to ensure the real-time perception and transmission of fresh data in delay-sensitivity green IoT applications [13], [14]. On the other hand, the impact of the computation offloading strategy on the AoI cannot be ignored [15]. When the IoT device's offloading decision is to execute locally, the update of AoI is mainly related to the local computing time; when the offloading decision is to execute on the edge cloud, the update of AoI is related to the waiting time, transmission time, and computing time in the edge cloud. Therefore, the optimal offloading and transmission strategies need to be formulated based on the IoT device's computing capacity and channel connection status to minimize the AoI.

Due to the channel dynamics between end users and edge clouds, optimizing the computation offloading and transmission strategies to minimize the average AoI of the system in face of time varying information poses a great challenge [16]. IoT applications with multi users are particularly sensitive to latency, which has higher requirements for AoI. In addition, there are few studies on considering the mutual influence between transmission and offloading strategies, in order to minimize the AoI by jointly optimizing the offloading and link scheduling strategies. Therefore, we propose an AoI-based joint optimization strategy for transmission scheduling and computation offloading in edge networks. This strategy comprehensively considers the AoI in both the task transmission and execution phases [17], and establish the objective with the weighted gain of AoI and computation rate (i.e., the number of bits processed per unit time). The main contributions of our work are summarized as follows.

1) We propose an AoI-based joint optimization strategy for transmission scheduling and computation offloading in edge networks. An optimization model is established with the integrated benefit of AoI and computation rate as the objective under the constraints of delay and energy consumption, ensuring the freshness of data during task transmission and execution.
2) To address the strong coupling between transmission and offloading decisions, the optimization objective is modeled as a Markov decision process (MDP), and a solution for transmission scheduling and computation offloading is proposed based on the deep deterministic policy gradient (DDPG) algorithm and the dueling double deep Q network (D3QN).
3) Simulation experiments are conducted to validate that the proposed solution outperforms four baseline offloading schemes and three transmission schemes, while achieving the maximum overall system benefit and the minimum AoI.

The remaining structure of this article is organized as follows: Section II reviews the studies related to AoI and MEC. Section III establishes the joint optimization model of AoI and computation rate for computation offloading and transmission scheduling. Section III-B designs computation offloading and transmission scheduling strategies based on DDPG and D3QN, respectively. Section IV conducts multiple comparative experiments to demonstrate the effectiveness of the proposed approach. Finally, Section V concludes this article.

II. Related Work

In order to measure the freshness of the information in real-time IoT applications, the concept of AoI was first proposed by Kaul et al. [10]. The previous research on AoI mainly focus on the transmission scheduling problem in wireless networks, and only a few started to address the computation offloading problem based on AoI-awareness in last two years. In this section, the related work on AoI-aware transmission scheduling and computation offloading will be introduced, respectively.

AoI-Based Data Transmission: To ensure the full utilization of wireless network resources, many advances have been made in AoI-aware transmission scheduling strategies in MEC networks. Kadota et al. [19] proposed a discrete-time decision transmission scheduling strategy for unreliable channels and compare the following four scheme in terms of AoI: 1) greedy strategy; 2) random strategy; 3) maximum weight strategy; and 4) Whittle index strategy. Talak et al. [20] considered the problem of minimizing the average and peak AoI in wireless networks under interference constraints and analyzed the relationship between packet generation rate and link scheduling strategy. Jiang et al. [21] proposed a polling scheduling strategy to achieve a stable distribution of time-averaged AoI. Chen et al. [22] designed a static threshold-based AoI-dependent random access protocol, in which each IoT device accesses the channel with a certain probability only when its instantaneous AoI exceeds the threshold. Chen et al. [23] researched the AoI-aware radio resource management in a Manhattan grid vehicle-to-vehicle network to make band allocation and packet scheduling decision. Bhat et al. [24] assumed that the complete information about the channel state is known, and discuss which user should be selected to transmit data with a specific transmission power, so as to minimize the average AoI under time division multiple access and nonorthogonal multiple access strategies, respectively. Qian et al. [25] examined the transmission scheduling strategy for channel–user matching in a multiuser multichannel system to minimize the overall average AoI.

AoI-Based Computation Offloading: In addition, making computation offloading decisions based on AoI in MEC networks has attracted the attention of a small number of researchers. Kuang et al. [15] studied the AoI of computation-intensive tasks in MEC and derived closed-form AoI values for three modes: 1) local offloading; 2) full offloading; and 3) partial offloading, assuming an exponential distribution of infinite queues and transmission times. Li et al. [26], [27] optimized the offloading ratio and power in MEC systems with nonorthogonal multiple access to minimize the weighted sum of AoI and interference cost. Chen et al. [11] considered that IoT devices compete for shared spectrum and computational resources over time, and then define the AoI for task updates and model the computational offloading decision as a stochastic game model for solution. Li et al. [28] defined the Age of Processing (AoP) based on the AoI by jointly optimizing the

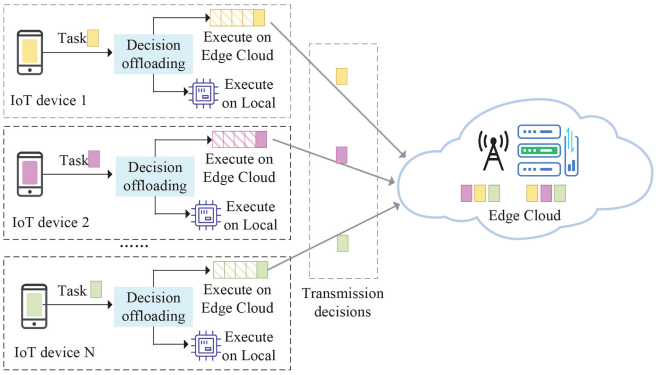

Fig. 1. System architecture of computation offloading and transmission scheduling.

state sampling frequency and computational offloading policy to minimize the average AoP in the long-term process, and formulated this problem as an infinite range constrained MDP (CMDP) with an average reward criterion, and transformed the CMDP problem into an unconstrained MDP using Lagrangian methods.

In general, most of above work is based on the AoI alone to develop the transmission scheduling policy or computation offloading policy, and rarely considers their joint optimization. On the one hand, the transmission scheduling policy of the edge cloud is affected by the offloading decision of IoT devices, and for the device whose offloading decision is locally executed, the edge cloud also does not need to schedule the device to transmit the task data. On the other hand, different transmission scheduling policies lead to different transmission AoI levels, which further affect the AoI level when the task is executed. Therefore, there is a strong coupling relationship between the offloading decision and the transmission decision, which is lack of consideration in aforementioned study.

III. System Model and Problem Formulation

A. System Architecture

The edge network system architecture for AoI-based computation offloading and transmission scheduling is shown in Fig. 1. This architecture includes a traditional fixed cellular base station and N IoT devices. The small edge server is deployed in the base station to provide computation offloading services to IoT devices with the capability of low-delay response. IoT devices have data acquisition units and processing units, continuously collecting real-time environmental data based on different application scenarios. These data need to be processed or transmitted in a timely manner to satisfy the demands of the users. As the wireless channel connection quality between IoT devices and the edge cloud changes dynamically, each time slot, the device needs to make offloading decisions based on the channel connection status, the computational amount, the latency requirements, and the remaining energy of the device. After receiving the offloading request from the terminal device, the edge server also needs to determine the scheduling of users for task data transmission based on the AoI requirements and channel status. Due to the limitation of wireless network channel capacity, only a limited number of tasks can be transmitted in each time slot. Therefore, for the tasks that are offloaded to the edge cloud and have not been transmitted, they need to wait in the transmission queue.

TABLE I
Table of Notations

Parameter	Description
N	The number of users
τ	Slot length
$S_i(t)$	Data size of task from user i in slot t
$x(t)$	Set of offloading decisions in slot t
$y(t)$	Set of transmission decisions in slot t
C_i	CPU cycles required for processing one bit of data
$f_{l,i}(t)$	Local computation capability of user device i
$D_{l,i}(t)$	Local computation delay of user i in slot t
$E_{l,i}(t)$	Local computation energy consumption of user i in slot t
$E_{max,i}(t)$	Maximum remaining energy of user device i in slot t
$r_{l,i}(t)$	Local computation rate of user i in slot t
$r_{o,i}(t)$	Transmission rate between user i and edge cloud in slot t
$D_{to,i}(t)$	Task transmission delay of user i in slot t
$D_{eo,i}(t)$	Task computation delay of user i at the edge cloud in slot t
$D_{o,i}(t)$	Offloading delay of user i in slot t
$E_{o,i}(t)$	Energy consumption of user i for task transmission in slot t
$A_{tr,i}(t)$	Task age during the transmission phase for user i in slot t
$A_{ex,i}(t)$	Task age during the execution phase for user i in slot t

Consider discrete time slots $t \in \{1, 2, 3, \ldots, T\}$, where the length of each time slot is denoted by τ, and assume that the execution time of each task cannot exceed one time slot [18]. The system consists of N IoT devices, corresponding to N computing tasks, and the set of tasks is denoted by $\mathcal{N} = \{1, 2, 3, \ldots, N\}$. At time slot t, the size of the computing task data generated by user $i \in \{1, 2, \ldots, N\}$ is denoted by $S_i(t)$. The set of each user's offloading decisions is denoted by $x(t) \triangleq (x_1(t), x_2(t), \ldots, x_N(t))$, where $x_i(t) = 1$ means that user i offloads its task to the edge server for computation, and $x_i(t) = 0$ means that user i performs the task locally. The transmission decision of the edge server is denoted by $y(t) \triangleq (y_1(t), y_2(t), \ldots, y_N(t))$, where $y_i(t) = 1$ means that user i is scheduled by the edge cloud to transmit the computing task in time slot t, and $y_i(t) = 0$ means that user i is not scheduled, and its computing task is waiting in the transmission queue. Each user needs to make an offloading decision first, and then the edge cloud makes a transmission scheduling decision based on the offloading requests of users who have decided to offload. Only users with $x_i(t) = 1$ can be scheduled by the edge cloud to transmit their computing tasks. The main symbols used in this chapter are summarized in Table I.

B. Network Model

1) Local Computing: When the wireless channel quality between the user and the edge cloud is poor, or the task computation requirement is low, the IoT device has certain computing capability. In this case, offloading the task to the edge cloud requires significant transmission delay and energy consumption. Therefore, the user is likely to make a decision to execute the task locally. For user i, the computation delay for the task to be computed locally on the device is as the following:

$$D_{l,i}(t) = \frac{C_i S_i(t)}{f_{l,i}(t)} \tag{1}$$

where C_i is the computational intensity of task i, which is the number of CPU cycles consumed per bit of data, and $f_{l,i}$ is the CPU frequency of device i. When the offloading decision is to execute the task locally, in order to ensure that the task is executed in a timely manner to avoid affecting the freshness of the data, it is assumed that the task needs to be completed within one time slot, subject to the following constraint:

$$D_{l,i}(t) \leq \tau. \tag{2}$$

The energy consumption $E_{l,i}(t)$ for task i when locally computed is shown as follows:

$$E_{l,i}(t) = k_i f_{l,i}(t)^3 D_{l,i}(t) = k_i f_{l,i}(t)^2 C_i S_i(t) \tag{3}$$

where k_i is the energy coefficient of device i, which is determined by the processor structure of the device. Since the battery capacity of terminal devices is limited, the energy consumed by tasks when executed locally cannot exceed the remaining battery energy $E_{\max,i}(t)$ is shown as follows:

$$E_{l,i}(t) \leq E_{\max,i}(t). \tag{4}$$

In addition to latency and energy consumption, the computation rate metric can also reflect the performance benefits of offloading decisions in the edge computing system. When the task is computed locally, the computation rate is the number of bits computed by the local CPU per unit time, and the local computation rate $r_{l,i}(t)$ is calculated as

$$r_{l,i}(t) = \frac{f_{l,i}(t)}{C_i}. \tag{5}$$

2) Edge Computing: Due to the limited local computing power of the terminal device, when the required computing workload of the task is large or the channel status is good, users usually choose to offload the task to the edge cloud for execution, exchanging the cost of transmission for the cost of local computation. When the offloading decision is made to execute the task at the edge cloud, the computation rate is equal to the data transmission rate $r_{o,i}(t)$, which is the number of bits transmitted per unit time. The transmission delay $D_{tr,i}(t)$ during transmission is given as

$$D_{to,i}(t) = \frac{S_i(t)}{r_{o,i}(t)}. \tag{6}$$

In addition to the transmission delay, the computation delay $D_{eo,i}(t)$ at the edge cloud is related to the computation resource allocation $f_{e,i}(t)$ assigned to task i by the edge cloud

$$D_{eo,i}(t) = \frac{S_i(t)}{f_{e,i}(t)}. \tag{7}$$

Since the computation result is small, the downlink transmission time of the edge cloud feedback to the user can be ignored. The overall delay of offloading task i to the edge cloud mainly includes the transmission delay and the computation delay in the edge cloud, that is, the offloading delay $D_{o,i}(t)$ is

$$D_{o,i}(t) = D_{to,i}(t) + D_{eo,i}(t). \tag{8}$$

The total offloading latency for each time slot should not exceed the length of the time slot τ, i.e.,

$$D_{o,i}(t) \leq \tau. \tag{9}$$

The total energy consumption of user device offloading mainly includes the transmission energy consumption $E_{o,i}(t)$

$$E_{o,i}(t) = e_i(t) D_{to,i}(t). \tag{10}$$

Similarly, the transmission energy cannot exceed the remaining energy of the device $E_{\max,i}(t)$, which is expressed as

$$E_{o,i}(t) \leq E_{\max,i}(t). \tag{11}$$

C. Age of Information Model

In each time slot, the user first makes an offloading decision $x(t)$, and then the edge cloud makes a transmission scheduling decision $y(t)$ based on the offloading decision. Since only the tasks that are offloaded to the edge cloud will generate transmission scheduling problems, the constraints that need to be satisfied between the offloading decision and the transmission decision are

$$y_i(t) \leq x_i(t) \quad \forall i \in \mathcal{N}. \tag{12}$$

For tasks offloaded to the edge cloud, to ensure the freshness of the data transmitted to the edge cloud, the AoI of the data packet during the transmission stage is used to represent its freshness, denoted as $A_{tr,i}(t)$. Since it is assumed that tasks are completed within one time slot, the AoI changes linearly. For users whose offloading decision is 1, if the edge cloud does not select them for data transmission, their tasks will wait in the transmission queue, and the AoI during the transmission phase will be increased by 1 based on the previous time slot. Conversely, if the user is scheduled by the edge cloud for data transmission, the AoI will be reset to 1, as the following:

$$A_{tr,i}(t) = \begin{cases} 1, & \text{if } y_i(t) = 1 \\ A_{tr,i}(t-1) + 1, & \text{else.} \end{cases} \tag{13}$$

In order to avoid situations where some users are never scheduled by the edge cloud for data transmission, which would result in an infinite increase in their AoI and therefore affect the correctness of user and edge cloud decisions, it is necessary to limit the AoI and provide a maximum threshold $A_{\max,i}$ for the AoI of each user.

In addition to the AoI during the transmission phase, we also considers the freshness of data during task execution, represented by $A_{ex,i}(t)$. Compared with the AoI during the transmission phase, the AoI during the execution phase $A_{ex,i}(t)$ also takes into account the task's computation delay [11], which means $A_{ex,i}(t)$ represents the number of time slots elapsed from the generation to the successful execution of user i's task. According to the user's offloading decision, the AoI during task execution can be divided into three cases. When the offloading decision is to execute locally, the task does not involve scheduling for transmission to the edge cloud. Therefore, if the task can be computed within one time slot and the local computation energy consumption satisfies the requirement of maximum remaining energy $E_{\max,i}(t)$, the task can be successfully executed, and $A_{ex,i}(t)$ will be reset to 1.

When the offloading decision is to offload to the edge cloud, it is also necessary to determine whether the task is selected for transmission based on the edge cloud's transmission scheduling decision. If the task data is successfully transmitted to the edge cloud and successfully executed, $A_{ex,i}(t)$ will also be reset to 1. Otherwise, if the user is not selected by the edge cloud, the task will continue to wait in the transmission queue, $A_{ex,i}(t)$ will be increased by 1 based on the previous moment, as follows:

$$A_{ex,i}(t) = \begin{cases} 1, & \text{if } x_i(t) = 0 \\ 1, & \text{if } x_i(t) = 1 \text{ and } y_i(t) = 1 \\ A_{ex,i}(t-1) + 1, & \text{else.} \end{cases} \quad (14)$$

Similarly, the execution-stage AoI $A_{ex,i}(t)$ is also subject to the maximum threshold limit $A_{\max,i}(t)$. If the AoI of user i reaches the maximum threshold limit, then the user's computation task will be immediately executed locally or transmitted to the edge cloud for execution.

D. Problem Formulation

Whether executed locally or on the edge cloud, the computation rate of user i, denoted by $r_i(t)$, can be expressed as

$$r_i(t) = (1 - x_i(t))r_{l,i}(t) + x_i(t)y_i(t)r_{o,i}(t). \quad (15)$$

Similarly, the total latency $D_i(t)$ and total energy consumption $E_i(t)$ can be represented as

$$D_i(t) = (1 - x_i(t))D_{l,i}(t) + x_i(t)y_i(t)D_{o,i}(t) \quad (16)$$

$$E_i(t) = (1 - x_i(t))E_{l,i}(t) + x_i(t)y_i(t)E_{o,i}(t). \quad (17)$$

To improve the freshness of data in MEC systems, the user's offloading decision needs to consider both the latency and energy constraints as well as the requirements for AoI. The offloading decision result further affects the formulation of edge cloud transmission scheduling decisions. In MEC systems with dynamically changing network states by considering quality of experience of multiusers and energy consumption, we jointly formulates offloading decision x and transmission decision y based on the current channel connection value quality, latency and energy constraints, and AoI requirements. In addition to minimizing the AoI of the MEC system, the optimization objective is to maximize the amount of bits processed per unit time, i.e., to maximize the system's computation rate. Therefore, the optimization problem is formulated as the weighted sum of task execution AoI and computation rate, is shown as follows:

$$\begin{aligned} P: \max_{x,y} \ & \lim_{T\to\infty} \frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T}\sum_{n=1}^{N} \eta r_i(t) - w_i A_{ex,i}(t)\right] \\ \text{s.t.} \ & C1: y_i(t) \le x_i(t) \quad \forall i \in \mathcal{N} \\ & C2: D_i(t) \le \tau \quad \forall i \in \mathcal{N} \\ & C3: E_i(t) \le E_{\max,i}(t) \quad \forall i \in \mathcal{N} \\ & C4: A_{tr,i}(t) \le A_{\max,i} \quad \forall i \in \mathcal{N} \\ & C5: A_{ex,i}(t) \le A_{\max,i} \quad \forall i \in \mathcal{N}. \end{aligned}$$

In optimization problem P, η is used to balance the importance of AoI and computation rate in the objective function. By adjusting the value of η, the contribution of computation rate to the overall system performance can be adjusted. When η is 0, the IoT system only considers AoI, and the objective of the unload and transmission decisions is to minimize the average AoI. As η increases, the weight of computation rate gradually increases, and the objective of the offloading and transmission decisions is to maximize the average weighted sum of AoI and computation rate. It should be noted that because smaller AoI values lead to higher system performance, the reward is calculated as a negative value of the AoI. w_i is the weight of AoI for each user's task execution process. Constraint C1 indicates that the transmission decision is made based on the unload decision, and only when the unload decision is to execute the task in the edge cloud will it be determined whether to schedule the task for transmission. Constraint C2 requires that the task execution delay cannot exceed the length of one time slot, C3 is the energy constraint, and C4 and C5 indicate that the AoI in both the transmission and execution phases of the task cannot exceed the maximum threshold $A_{\max,i}$.

IV. Joint Optimization Strategy for Computation Offloading and Transmission Scheduling

A. Overall Optimization Strategy Design

Our joint optimization strategy is denoted by $\pi = (\pi_{ex}, \pi_{tr})$, where π_{ex} is the offloading strategy, and π_{tr} is the transmission strategy. Different from the previous scheme which uses one deep reinforcement learning (DRL) algorithm to obtain multiple solution variables [29], due to the strong coupling between the two variables, the offloading decision needs to be given first, and then the transmission decision can select the user according to the place of the task execution. If one DRL algorithm is directly used to output the offloading decision and the transmission decision simultaneously, and the making of the two decisions is isolated, then the offloading decision cannot provide relevant information and guidance for the transmission decision. Therefore, this article uses a combination of two DRL algorithms to obtain the offloading strategy and the transmission strategy, respectively. Next, the design of the state space, action space, and reward function for the two strategies are presented.

State Space: The state $s_{ex}(t)$ of the offloading strategy is expressed as $s_{ex}(t) = \{s_{ex,1}(t), s_{ex,2}(t), \ldots, s_{ex,N}(t)\}$, where each component $s_{ex,i}(t)$ is composed of the transmission decision of the previous time slot, the current offloading AoI, the transmission AoI, computation rate, delay, and energy consumption, that is, $s_{ex,i}(t) \triangleq \{a_{tr}(t-1), A_{ex,i}(t), A_{tr,i}(t), r_i(t), D_i(t), E_i(t)\}$. The state $s_{tr}(t)$ of the transmission strategy is denoted as $s_{tr}(t) = \{s_{tr,1}(t), s_{tr,2}(t), \ldots, s_{tr,N}(t)\}$, where each component $s_{tr,i}(t)$ consists of the user's offloading decision in the current time slot, the data transmission rate of the uplink and the transmission AoI, that is, $s_{tr,i}(t) \triangleq \{x_i(t), r_{o,i}(t), A_{tr,i}(t)\}$.

Action Space: In the binary offloading mode, the offloading action $a_{ex}(t)$ consists of 0 and 1. Considering that when the offloading decision is executed by the edge cloud, the task is not scheduled for transmission but continues to wait, it is necessary to add an action dimension -1. In addition, when the

execution delay of the task exceeds the time slot length τ or the energy consumption exceeds $E_{\max,i}(t)$, the task is also in the state of waiting for execution, and the task is continued until the channel state is good or the delay and energy consumption meet the constraints. Therefore, each user's offloading action is $a_{ex,i}(t) = \{-1, 0, 1\}$. When the offloading action is -1, the corresponding computation rate, delay, and energy consumption are recorded as 0. At time slot t, the set of users whose offloading decision is executed by the edge cloud is recorded as $N'(t)$. Then the transmission decision is to select users in $N'(t)$ to offload the task to the edge cloud, so $a_{tr}(t) = \{1, 2, \ldots, N'(t)\}$. If $a_{tr}(t) = i$, then the transmission decision $y_i(t) = 1$, i.e., user i is selected by the edge cloud for task transmission, and vice versa, user i is not selected for transmission.

Reward Function: The goal of the offloading policy π_{ex} is to maximize the weighted gain of the computation rate and the AoI of the task execution, while also satisfying the latency and energy consumption requirements. In designing the reward function of the offloading strategy, the delay and energy consumption constraints are taken into account in the reward function $R_{ex}(t)$. Thus, at time slot t, the reward function $R_{ex}(t)$ is calculated as the following:

$$R_{ex}(t) = \sum_{i=1}^{N} \eta r_i(t) - w_i A_{ex,i} + \alpha(\tau - D_i(t)) + \beta\big(E_{\max,i}(t) - E_i(t)\big) \quad (18)$$

where α and β are the weighting factors of delay constraint and energy consumption constraint, respectively. When the delay or energy consumption of an offloading action does not meet the constraint requirements, the reward value corresponding to the action will decrease, and the strategy will reduce the execution probability of the action in the next action selection. The goal of the transmission strategy π_{tr} is to select the user for transmission from the offloading set $N'(t)$ when the offloading action is known, so as to maximize the weighted benefit of the AoI and data transmission rate in the transmission phase. The reward function $R_{tr}(t)$ of the transmission strategy is calculated according to the following:

$$R_{tr}(t) = \sum_{i=1}^{N'(t)} \eta y_i(t) r_{o,i}(t) - w_i A_{tr,i}(t). \quad (19)$$

At the beginning of each time slot, the state value $s_{ex}(t)$ is first input into the offloading strategy to obtain the offloading decision, which determines the execution position of each user. The reward value $R_{ex}(t)$ is not immediately available at this point, but rather, the offloading decision needs to be input into the transmission algorithm. The transmission strategy selects the user from the users whose offloading position is the edge cloud based on the offloading action, the data transmission rate and the transmission AoI, performs the transmission action, reaches the next state value $s_{tr}(t+1)$ and gets the transmission reward value $R_{tr}(t)$, and then sends the transmission decision and AoI value to the offloading strategy. According to the offloading and transmission action, the offloading strategy calculates the AoI, the computation rate, delay, and energy consumption of the offloading, reaches the next state $s_{ex}(t+1)$, and obtains the reward value $R_{ex}(t)$. Next, proceed to the next time slot $t+1$. It should be noted that, after the transmission strategy selects the transmission user, the unselected user tasks are still in the waiting state. Then, in the time slot $t+1$, for the task in the waiting state, the taken offloading action is -1, i.e., the task is not executed. In addition, for a task whose time delay and energy consumption exceed the constraint condition after the offloading action is performed, action -1 is also taken, that is, the task is not suitable to be executed currently, and is also in a waiting state until the condition is suitable for execution. When the offloading strategy executes action -1, the computation rate, latency, and energy consumption are all 0. However, if the AoI (including $A_{ex}(t)$ and $A_{tr}(t)$) in the waiting state exceeds the threshold limit, the action of executing the task locally or transmitting the task to the edge cloud is directly taken.

B. DDPG-Based Computation Offloading Strategy

The DDPG algorithm combines the advantages of the deep Q network (DQN) algorithm and actor–critic algorithm [30], solves the problem that the sample data cannot be independently and identically distributed through the mechanism of experience replay and the mechanism of coexistence of online network and target network [31]. The actor network and critical network are used to formulate and evaluate strategies, respectively. Fig. 2 is the diagram of the DDPG algorithm. The DDPG algorithm contains a total of four networks, namely, the online actor network, the online critic network, the target actor network, and the target critic network. The online actor network parameters are denoted as θ^u, the online critic network parameters are denoted as θ^Q, the target actor network parameters are denoted as $\theta^{u'}$, and the target critic network parameters are denoted as $\theta^{Q'}$.

The online actor network is used to interact with the environment to generate sample data. The user state $s_{ex}(t)$ is input to the online actor network, and the Gumbel-Softmax in [32] is used to obtain the discrete offloading action $a_{ex}(t)$. The edge cloud selects the user for transmission according to the offloading strategy and sends the transmission strategy to the user, who gets the reward value $R_{ex}(t)$ according to the offloading and transmission decision, and arrives at the next state $s_{ex}(t+1)$. The data $[s_{ex}(t), a_{ex}(t), R_{ex}(t), s_{ex}(t+1)]$ generated in the interaction process is stored in the experience playback unit, and when there are enough samples in the experience playback unit, a batch of sample data is taken out for updating the four networks. Record the sample data taken as $[s_{ex}^m, a_{ex}^m, R_{ex}^m, s_{ex}^{m'}]$. When updating the online critical network, s_{ex}^m and a_{ex}^m are input to the online critical network to obtain the state–action evaluation value Q. To get the action corresponding to $s_{ex}^{m'}$, $s_{ex}^{m'}$ is fed into the target actor network, and then the corresponding action is evaluated by the target critic network to get the value $'$. The loss function of the online critic network is as follows:

$$L = \frac{1}{M}\sum_{m=1}^{M}\Big[R_{ex}^m + \gamma_{ex} Q'\Big(s_{ex}^{m'}, u'\Big(s_{ex}^{m'}\Big)\Big) - Q\big(s_{ex}^m, a_{ex}^m\big)\Big]^2 \quad (20)$$

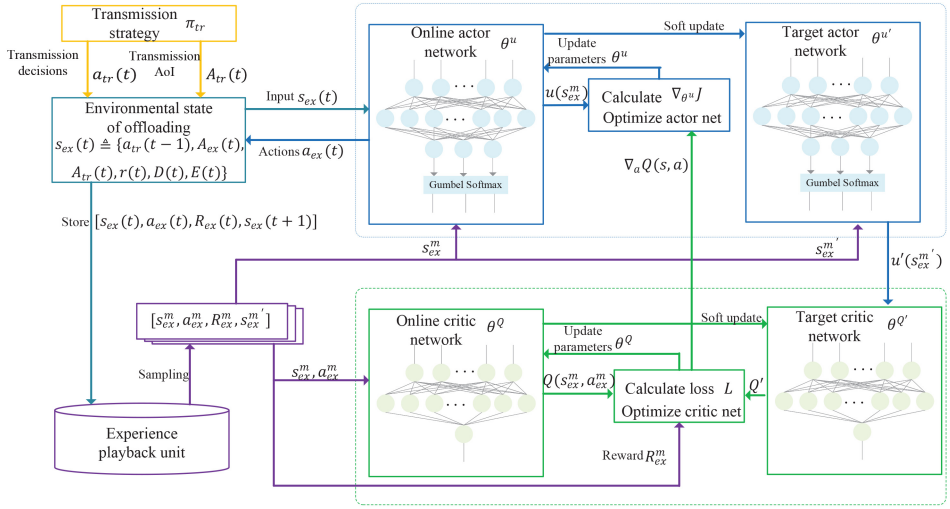

Fig. 2. DDPG-based computation offloading strategy.

where M is the number of taken samples, and γ_{ex} is the discount factor in the offloading strategy. We use the gradient descent method to update the online critical network parameter θ^Q. Next, in order to update the online actor network, it is necessary to input the Q value obtained from the online critical network into the online actor network, as shown in the following:

$$\nabla_\theta uJ = \frac{1}{M}\sum_{m=1}^{M} \nabla_a Q(s,a)\Big|_{s=s_{ex}^m, a=u(s_{ex}^m)} \nabla_\theta uu\left(s_{ex}^m\right). \quad (21)$$

After the online network is updated, the soft update method is used to update the two target networks, as shown

$$\begin{cases} \theta^{u'} = \delta\theta^u + (1-\delta)\theta^{u'} \\ \theta^{Q'} = \delta\theta^Q + (1-\delta)\theta^{Q'} \end{cases} \quad (22)$$

where δ is a constant that is much less than 1. Algorithm 1 summarizes the detailed process of solving the unloading strategy based on DDPG.

C. D3QN-Based Transmission Scheduling Strategy

When solving the transmission strategy, it is mainly to select tasks for transmission from the users whose execution location is the edge cloud. Compared with the DDPG algorithm, the DQN algorithm is simpler and suitable for the user selection problem in the transmission scheduling scenario, but the DQN algorithm also has some defects. For the state s'_{tr} of the next time node, DQN estimates the action selection a'_{tr} and value function Q of s'_{tr} all in the target network. However, the action a'_{tr} with the largest target Q value is not necessarily the best action for the state s'_{tr}, which can easily lead to the estimated Q value of the target network being larger than the real value function, and the problem of overestimation occurs. Overestimation spreads wrong information, resulting in the trained parameter θ not being able to fit the state value well, and as the number of training increases, the error will gradually increase.

Algorithm 1 DDPG-Based Offloading Strategy

Input:
Time slot τ, the weight η of computation rate, weights α and β of delay constraint and energy consumption constraint;

Output:
The optimal task offloading strategy π_{ex}^*;

1: Initialize state $s_{ex}(0)$ and online network parameters θ^u and θ^Q;
2: **for** $t = 1, 2, \ldots, T$ **do**
3: Input the state $s_{ex}(t)$ into the online actor network to obtain the action value $a_{ex}(t)$;
4: Obtain the transmission strategy π_{tr} and the transmission AoI according to the Algorithm 2;
5: Users perform the offloading action, calculating the latency, energy consumption, computation rate and the offloading AoI;
6: Calculate the reward value $R_{ex}(t+1)$ according to the equation (18) and update the state $s_{ex}(t+1)$;
7: Store the data $[s_{ex}(t), a_{ex}(t), R_{ex}(t), s_{ex}(t+1)]$ in the experience playback unit;
8: Take out a batch of data from that experience playback unit for training;
9: Update the online critic and actor network according to the equation (20) and (21), respectively;
10: Update the target actor and critic network according to the equation (22);
11: **end for**

To solve the problem of overestimation in DQN, we introduce the double DQN algorithm, which decouples the estimation of the action value function from the action selection of s'_{tr} [33]. The action selection for the next time node state s'_{tr} is performed in the online network, that is, the action a'_{tr} corresponding to the maximum $Q(s'_{tr}, a'_{tr}; \theta)$ is selected, where θ is the online network parameter. The value function estimation of a'_{tr} is performed in the target network, that is, the value

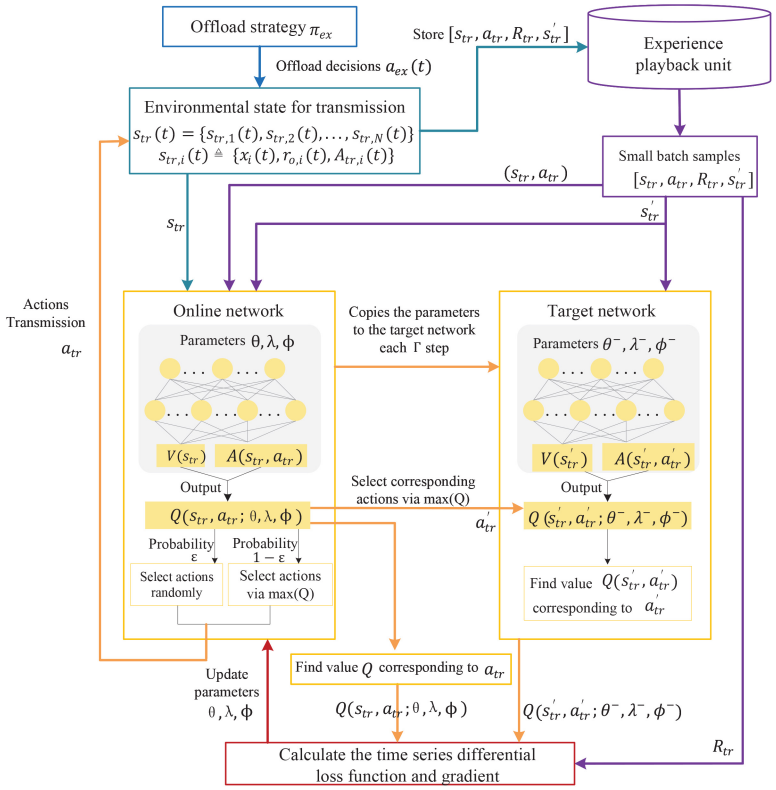

Fig. 3. D3QN-based transmission scheduling strategy.

function of a'_{tr} is estimated as $Q(s_{tr}^{-})$, where θ^{-} is the target network parameter. The loss function $L^{\mathrm{DDQN}}(\theta)$ of double DQN is as the following:

$$\mathcal{L}^{\mathrm{DDQN}}(\theta) = \mathbb{E}\left[\left(R_{tx} + \gamma_{tr} Q\left(s'_{tr}, \arg\max_{a'_{tr}} Q\left(s'_{tr}, a'_{tr}; \theta\right); \theta^{-}\right) - Q(s_{tr}, a_{tr}; \theta)\right)^{2}\right]. \tag{23}$$

However, in some cases, different actions taken in the same state have the same impact on the environment, which can lead to the same reward value. At this time, it is not necessary to evaluate the value of each action choice. However, different states still have different values. One problem with DQN or double DQN is that it is impossible to distinguish the value of states. In order to evaluate the advantages and disadvantages of state value functions, we also introduced the dueling DQN algorithm, which decomposes the action value function $Q(s_{tr}, a_{tr})$ into a state value function $V(s_{tr})$ and an action advantage function $A(s_{tr}, a_{tr})$, that is, $Q(s_{tr}, a_{tr}) = V(s_{tr}) + A(s_{tr}, a_{tr})$ [34]. This algorithm adjusts the structure of the deep neural network in DQN. The first few layers of the network are the same as the structure in DQN, and the parameter is θ. In the previous layer of the output layer, the network is divided into two parallel structures, one network is used to calculate the state value function $V(s_{tr})$, the parameter is λ; the other network is used to calculate the action advantage function $A(s_{tr}, a_{tr})$, the parameter is ϕ. Finally, there is an aggregation layer that adds the state value function $V(s_{tr})$ and the action advantage function $A(s_{tr}, a_{tr})$ to obtain the final action value function $Q(s_{tr}, a_{tr})$. However, in dueling DQN, if the state value function and the action advantage function are directly added, once the final output Q value is given, it will cause the lack of recognizability of V and A. To solve this problem, the advantage function needs to subtract the mean for decentralization. The action value function $Q(s_{tr}, a_{tr}; \theta, \lambda, \phi)$ in dueling DQN can be expressed as the following formula:

$$Q(s_{tr}, a_{tr}; \theta, \lambda, \phi) = V(s_{tr}; \theta, \lambda) + \left[A(s_{tr}, a_{tr}; \theta, \phi) - \frac{1}{|N'(t)|} \sum_{a'_{tr} \in N'(t)} A\left(s_{tr}, a'_{tr}; \theta, \phi\right)\right]. \tag{24}$$

Combining the advantages of double DQN and dueling DQN, we fuse the two DRL algorithms to obtain dueling double DQN (D3QN), and then use the D3QN algorithm to solve the optimal transmission policy for edge clouds, as shown in Fig. 3. The temporal differential error L of L^{D3QN} is calculated according to the following:

$$\mathcal{L}^{D3QN}(\theta, \lambda, \phi) = \mathbb{E}\left[\left(R_{tr} + \gamma_{tr} Q\left(s'_{tr}, \arg\max_{a'_{tr}} Q\left(s'_{tr}, a'_{tr}; \theta, \lambda, \phi\right); \theta^{-}, \lambda^{-}, \phi^{-}\right) - Q(s_{tr}, a_{tr}; \theta, \lambda, \phi)\right)^{2}\right] \tag{25}$$

where θ, λ, and ϕ are the online network parameters and θ^{-}, λ^{-}, and ϕ^{-} are the target network parameters.

The process of solving the transmission strategy based on D3QN is shown in Algorithm 2. At each moment, the transmission state s_{tr} is first input into the online network to

Algorithm 2 D3QN-Based Transmission Scheduling Strategy

Input:

The weight η of computation rate, the weight w_i of AoI;

Output:

The optimal transmission strategy π_{tr}^{*};

Initialize the state $s_{tr}(0)$ and the network parameters θ, λ, ϕ;
Initialize $\epsilon = 1$, minimum value $\epsilon_{\min}$;
for $t = 1, 2, \ldots, T$ **do**
  Obtain the offloading policy π_{ex} according to Algorithm 1;
  Generate the transmission decision $a_{tr}(t)$ via the ϵ-greedy policy;
  Set the selected user's transmission AoI to 1, and the unselected user plus 1;
  if the transmission AoI of non-selected users $\geq A_{\max,i}$ **then**
    set the AoI of this user to $A_{\max,i}$;
    Increase the probability of the user being selected;
  end if
  Get the next state s'_{tr} according to the data transmission rate and AoI;
  Calculate the reward function R_{tr} according to the equation (19);
  Store the sequence pair $[s_{tr}, a_{tr}, R_{tr}, s'_{tr}]$ into the experience playback unit;
  Randomly select a batch of data sets from the experience playback unit for training;
  Calculate the loss function L^{D3QN} according to the equation (25);
  Update θ, λ, ϕ through gradient descent method;
  if training steps % Γ == 0 **then**
    Copy θ, λ, ϕ to $\theta^{-}, \lambda^{-}, \phi^{-}$;
  end if
end for

obtain the state value function $V(s_{tr})$ and the action advantage function $A(s_{tr}, a_{tr})$, and further aggregated to obtain the state–action value function $Q(s_{tr}, a_{tr})$, and then the ϵ-greedy strategy of exploration rate decay is adopted. At the initial training stage of the algorithm, because the edge cloud has not yet learned a better transmission strategy, it randomly selects an action with a large probability value ϵ for exploration, and selects the action a_{tr} corresponding to the largest Q value with a small utilization probability value $1-\epsilon$, so as to avoid falling into a local optimal situation. The increment of each decay of ϵ is 0.001, and the minimum value of decay $\epsilon_{\min}$ is 0.1. After the algorithm has been trained for a period of time, the edge cloud has learned a better strategy, so the probability of utilization is gradually increased, and the probability of exploration is reduced until the probability of exploration decays to the minimum value of 0.1. Afterward, when the DRL agent of the edge cloud interacts with the environment, the user is randomly selected for data transmission with a probability of 0.1, and the action corresponding to the maximum Q is selected with a probability of 0.9.

After the edge cloud selects the transmission user, the status of transmission AoI and computation rate shifts to s'_{tr}. When the user's AoI reaches the maximum threshold $A_{\max,i}$ the probability of the user being selected is increased in the next transmission scheduling, and the action selection of the edge cloud at the next moment will be more inclined toward the user. The interaction data $[s_{tr}, a_{tr}, R_{tr}, s'_{tr}]$ is stored in the experience playback unit M. When the number of interactions reaches a predetermined t_{pre}, a batch of data is randomly selected from M to update the evaluation network. After each training Γ steps of the evaluation network, the parameters are copied to θ^{-}, λ^{-}, and ϕ^{-}. As the evaluation network continues to train and learn, the algorithm eventually reaches a stable state, and the edge cloud obtains an approximately optimal transmission strategy π_{tr}^{*}.

D. Complexity Analysis

The complexity of our proposed joint optimization algorithm consists of two parts, namely, the computation offloading solution algorithm of Algorithm 1 and the transmission scheduling solution algorithm of Algorithm 2. Since the user's offloading decision is formulated in the online execution phase of the DDPG algorithm, the main focus is on the complexity of the online execution of the algorithm. The complexity of the online execution phase of the DDPG algorithm is related to the input state dimension of the actor network, the output action dimension, the number of network layers, and the number of neurons.The input state dimension of the actor network is $|s_{ex}|$, the output action dimension is $|a_{ex}|$, the number of hidden layers is L_{ex} and the number of hidden units in the l_{ex}th layer is $K_{l_{ex}}$. The complexity of a single user making an uninstall decision is $O(G1) = O(|s_{ex}|K_1 + \sum_{l_{ex}=1}^{L_{ex}^{-1}} K_{l_{ex}} K_{l_{ex+1}} + K_{L_{ex}}|a_{ex}|)$ and the overall complexity of N users making an uninstall decision is $O(NG1)$. When the edge cloud makes transmission scheduling decisions, it also focuses on the online execution complexity of the D3QN algorithm. The input state dimension of the D3QN algorithm online network is $|s_{tr}|$, the output action dimension is $|a_{tr}|$, the number of hidden layers is L_{tr}, and the number of hidden units in the l_{tr}th layer is $K_{l_{tr}}$, then the complexity of the transmission scheduling algorithm is $O(G2) = O(|s_{tr}|K_1 + \sum_{l_{tr}=1}^{L_{tr}-1} K_{ltr} K_{l_{tr+1}} + K_{L_{tr}}|a_{tr}|)$. Based on the above analysis, the overall complexity of the joint optimization algorithm for computing offloading and transmission scheduling is $O(NG1 + G2)$. Since the number of network layers and hidden units in the DDPG algorithm and D3QN algorithm are fixed, the complexity of the algorithm is mainly related to the input state dimension, output action dimension, and the number of users.

V. Experiment and Performance Analysis

A. Parameters Setting

According to the work of Chen et al. [35], the wireless channel maintains the original state with a probability of 0.8 and is in another channel state with a probability of 0.2 at each moment. The initial data transmission rates are (1.5, 2.25, 1.25, 1.5) MB/s, and the changed data transmission rates are (0.768,

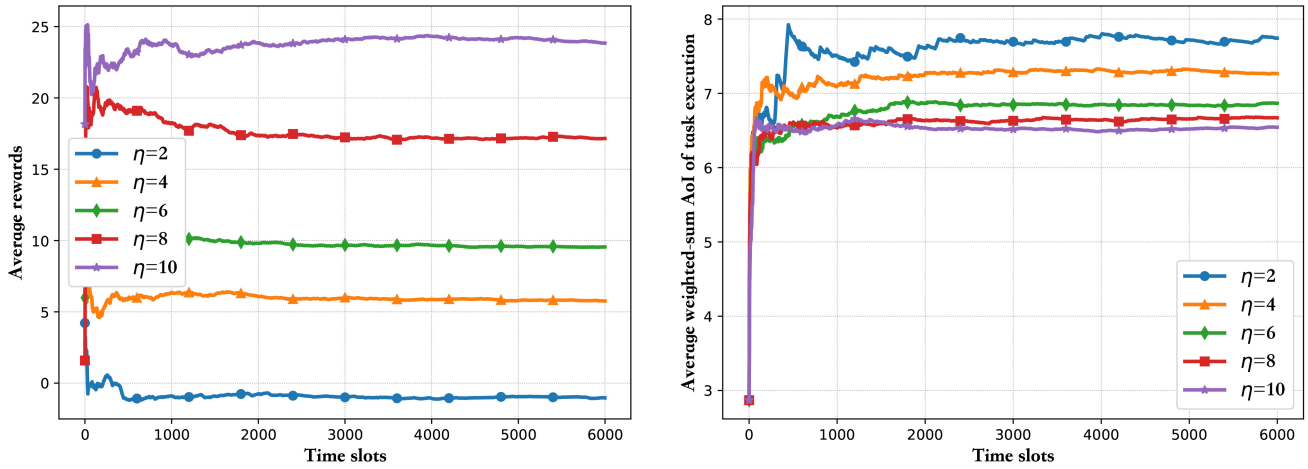

Fig. 4. Convergence under different η. (a) Effect of η on average reward. (b) Effect of η on average weighted-sum AoI.

1.0, 0.384, 1.12) MB/s. The data size $S_i(t)$ of computing tasks in each time slot is evenly distributed between [1,2] Mbits, the residual energy $E_{\max,i}(t)$ of the IoT device is evenly distributed between [0.5,1.5] J, and the time slot length τ is 1 s. The energy consumption coefficient of the IoT device is 10^{-28} and the calculation intensity C_i is 1000 cycles/bit. The edge cloud has 50 GHz of computing power, and computing resources are evenly distributed to users whose offloading decisions are performed for the edge cloud. The local computing power $f_{l,i}$ is evenly distributed between [0.5, 1] GHz. The maximum values $A_{\max,i}$ of the AoI in the offloading stage and the transmission stage are both 10, and the initial AoI are set to 0. Unless otherwise specified, the AoI weight of each user is 1 in the offloading stage, the computation rate weight η is 10, and the delay and energy consumption constraint parameters α and β are 1.

In the offloading strategy based on the DDPG algorithm, two hidden layers are set, with 64 hidden units per layer. The capacity of the experience playback unit is 2000, the batch data size is set to 64, the learning rate of both the actor and critical networks is 0.001, and the soft update factor δ is 0.01. Each hidden layer is output using the ReLU activation function, and parameters are updated using the Adam optimizer. The discount factor γ_{ex} of the reward function is 0.95. In the D3QN-based transmission strategy, three hidden layers are set, with 250 hidden units per layer. The size of the batch is 32, and the capacity of the experience playback unit is also 2000. After 500 online network executions, network training, and updates begin. After every 200 training sessions of the online network, the parameters θ, λ, and ϕ are copied to θ^-, λ^-, ϕ^-. The discount factor γ_{tr} of the reward function is 0.9, and the learning rate is also 0.001. Like the DDPG algorithm, the ReLU activation function and the Adam optimizer are used.

In order to verify the scheme proposed in this article, we compare the proposed scheme with four other offloading schemes. The four compared offloading schemes are: 1) computation rate-based offloading scheme, i.e., the optimization objective is to maximize the computation rate without considering the requirement of information freshness; 2) random offloading scheme, i.e., each time slot user randomly chooses to execute locally or in the edge cloud; 3) local execution scheme, i.e., the tasks are all executed locally; and 4) all offloading scheme, i.e., all tasks are offloaded to the edge cloud for execution. In addition, to verify the performance of the transmission scheduling strategy, we also compare the D3QN-based transmission scheme with the DQN-based, the policy gradient-based, and the random transmission policy.

B. Performance Analysis

Fig. 4 shows the influence of different η on the average rewards of the system and the average weighted-sum AoI of task execution. It can be clearly seen that they all converge in 6000 iterations under five different values of η. In addition, as can be seen from Fig. 4(a), as η increases, the average rewards of the system also increases. This is because the system revenue is calculated as the weighted sum of the negative value of the AoI and the computation rate, then plus the impact of delay and energy constraints. If η is larger, the impact of computation rate on overall benefits will also be greater, which leads to users being more inclined to make actions that increase computation rate (including local computation rate and data transmission rate) when making offloading decisions. As can be seen from Fig. 4(b), the average weighted-sum AoI decreases as η increases, which reflects the higher freshness of information during task execution. The increase in η skews the user's offloading decision toward actions with larger computation rates. If the local computation rate is greater, tasks are executed more quickly locally, making the AoI decrease. If the data transmission rate is greater, then tasks are also transmitted more quickly to the edge cloud, so the AoI also decreases.

As shown in Fig. 5, with the increase of the local computing power of the IoT device, the average rewards of the proposed scheme and the other three schemes increases except for the all offloading scheme. This is because the local computation rate increases, and therefore the average revenue value of the system also increases. In the all offloading scheme, tasks are

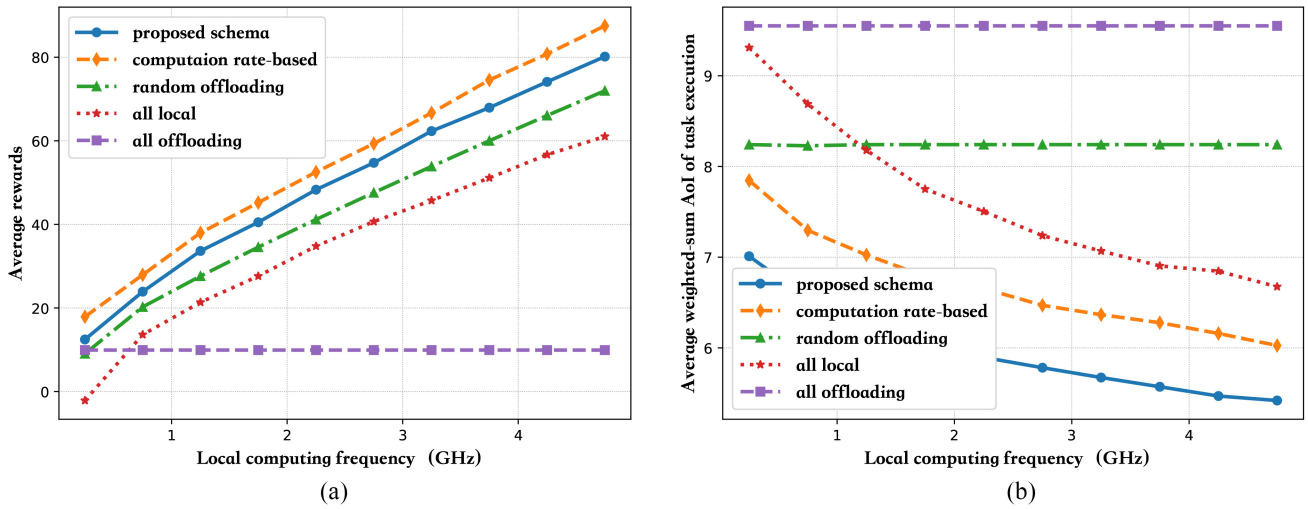

Fig. 5. Comparison of the impact of local computing power. (a) Comparison of average reward. (b) Comparison of average weighted-sum AoI.

all offloaded to the edge cloud for execution, and the channel state between the user and the edge cloud is independent of the local computing capacity, so the average revenue value of all offloading schemes is not affected by the local computing capacity of the device. Moreover, as can be seen from Fig. 5(a), the average benefits of the proposed scheme is larger than that of the other three schemes except for the computation rate-based offloading scheme. This is because, in the computation rate-based offloading scheme, the system revenue is aimed at maximizing the computation rate, without considering the AoI. However, this does not mean that this scheme is superior to the proposed scheme. From Fig. 5(b), it can be seen that this scheme is at the expense of AoI. The proposed scheme takes into account both the computation rate and the AoI, while the random offloading scheme takes neither the computation rate nor the AoI into account, and the local execution and all offloading schemes have fixed actions and cannot adapt to the dynamic changes of the channel, so the system benefit is smaller than the proposed scheme.

In the transmission experiment, the transmission AoI weights of the four users are set to (4, 3, 2, 1), respectively. Fig. 6(a) shows the change of the average weighted-sum AoI in transmission phase with the number of iterations. Obviously, the AoI of the four transmission schemes converge in 6000 iterations. The D3QN-based transmission scheme has the smallest AoI among the four schemes, followed by the DQN algorithm, and the random transmission scheme has the largest AoI. This is because the D3QN-base strategy combines the advantages of dueling DQN and double DQN, solves the overestimation problem in the DQN algorithm, and introduces state value function and action advantage function, which is better than the unimproved DQN algorithm. Policy gradient algorithm has poor stability because of the correlation between the previous and subsequent states in the data sequence used in parameters update, but its effect is better than the random algorithm. The random transmission scheme does not consider the influence of the AoI, and the user is randomly selected for data transmission each time, so the AoI is the largest.

Fig. 6(b) shows the proportion of each user being successfully transmitted under the four transmission schemes. The AoI weight and network environment state of different users are considered in the D3QN-based transmission scheduling strategy. Since user2 has a better network environment (higher data transmission rate) and the AoI weight is second only to user1, the proportion of successfully transmitted packets is the highest. User1 has the highest AoI weight, but due to its poor network environment, the edge cloud is affected by the computation rate when making transmission policies, so the percentage of successfully transmitted packets for user1 is slightly lower than that for user2. However, due to the low-AoI weight, the proportion of successfully transmitted data packets of user3 and user4 is lower than that of user1 and user2. This is consistent with the transmission scheduling strategy to be implemented in the actual situation, that is, user data with a higher AoI weight level is transmitted as far as possible, but the network environment of different users cannot be ignored. The problem with the DQN-based strategy is that the transmission ratio of user2 is too high compared to user1, which does not meet the requirement that user1 has the highest AoI weight. In the policy gradient-based scheme, user4, which has the lowest AoI weight, is completely ignored, resulting in the packets in user4 not being transmitted all the time. In the random transmission scheme, users are randomly selected to transmit each time, and the AoI weight characteristics are not considered at all, and the influence of the network environment is also not taken into account.

Fig. 6(c) shows the change of the average data transmission rate of four users based on D3QN transmission strategy. Obviously, the average data transmission rate of user 2 is higher because it is scheduled by the edge cloud more times. The second is user 1. Because its AoI weight level is the highest, then the AoI of this user has the greatest impact on the overall income among the four users. Although the network environment is poor, it is also selected by the edge cloud for many times, so the average data transmission rate is still high. User 3 and user 4 are selected less times due to lower AoI

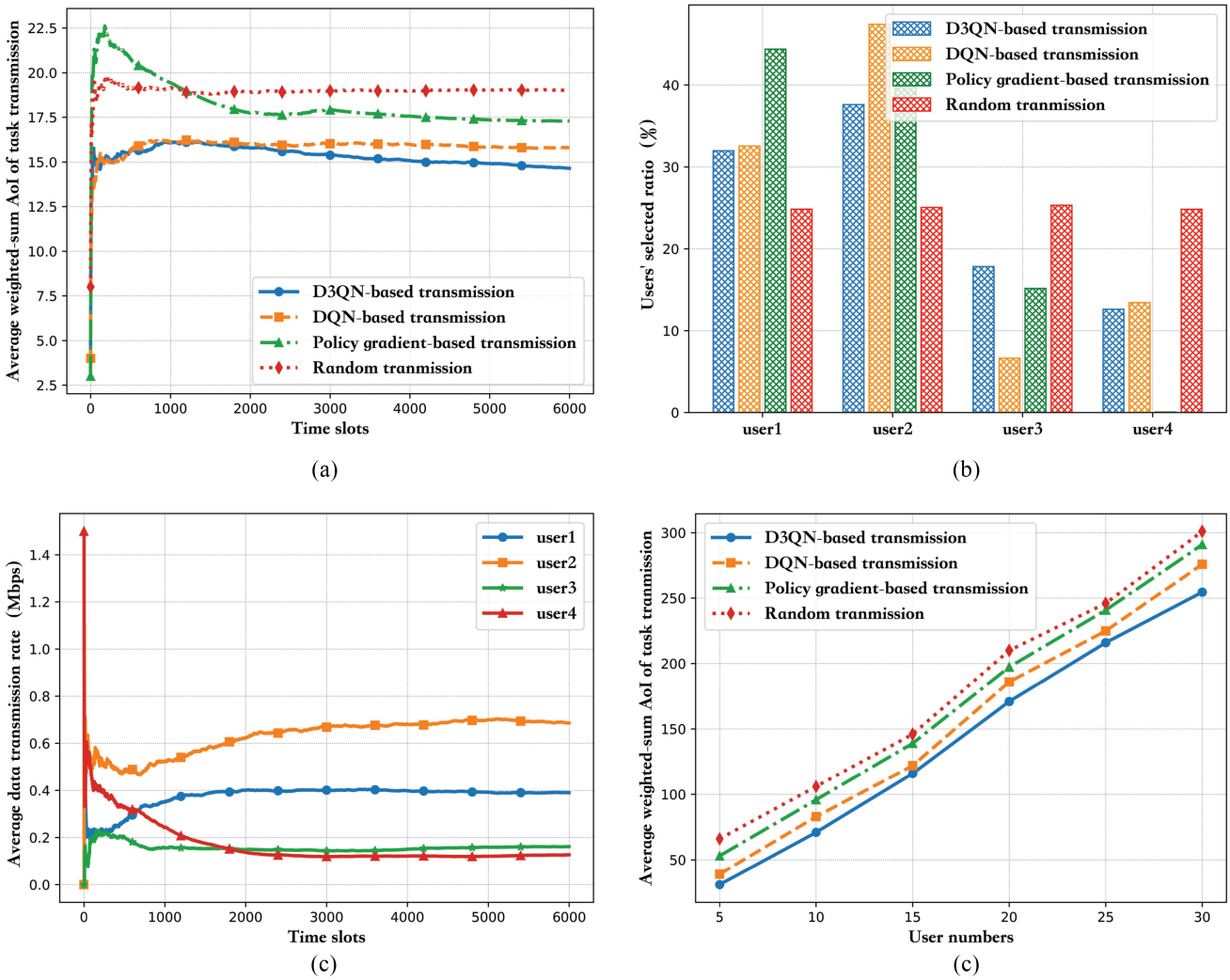

Fig. 6. Performance comparison of transmission strategies. (a) Convergence of transmission AoI. (b) Percentage of packets successfully transmitted. (c) Average data transmission rate. (d) Effect of the number of users on AoI.

weight level, and the average data transmission rate is lower than that of the first two users.

Fig. 6(d) shows the variation of the average weighted-sum AoI with the number of users for the four transmission strategies. As the number of users increases, the AoI performance is decreasing. However, the D3QN-based strategy still has the smallest AoI. This shows that the D3QN-based transmission scheduling strategy can provide a good perception of the overall gain in AoI and computation rate, so that the data received in the edge cloud is as fresh as possible, thus giving more reliable analysis results.

VI. Conclusion

In order to guarantee the timeliness and freshness of data in real-time IoT application systems, this article proposes an AoI-aware joint optimization strategy for computation offloading and transmission scheduling. Considering the constraints of delay and energy consumption, our model aims at the comprehensive benefit of AoI and computation rate in task execution and transmission phase. Then, a joint optimization strategy of computation offloading and transmission scheduling based on DDPG and D3QN is proposed. Experimental results show that the proposed scheme has greater overall system revenue and lower AoI than the other four offloading schemes and three transmission schemes.

References

[1] X. Diao, Y. Cai, B. Yu, and Q. Wu, "Location and complex status update strategy optimization in UAV-assisted IoT," *IEEE Internet Things J.*, early access, Feb. 22, 2023, doi: 10.1109/JIOT.2023.3244541.

[2] C. Guo, X. Wang, L. Liang, and G. Y. Li, "Age of information, latency, and reliability in intelligent vehicular networks," *IEEE Netw.*, early access, Sep. 26, 2022, doi: 10.1109/MNET.124.2200132.

[3] Y. Hao, Y. Miao, M. Chen, H. Gharavi, and V. C. M. Leung, "6G cognitive information theory: A mailbox perspective," *Big Data Cogn. Comput.*, vol. 5, no. 4, p. 56, 2021.

[4] A. Muhammad, I. Sorkhoh, M. Samir, D. Ebrahimi, and C. Assi, "Minimizing age of information in multiaccess-edge-computing-assisted IoT networks," *IEEE Internet Things J.*, vol. 9, no. 15, pp. 13052–13066, Aug. 2022.

[5] F. Chai, Q. Zhang, H. Yao, X. Xin, R. Gao, and M. Guizani, "Joint multi-task offloading and resource allocation for mobile edge computing systems in satellite IoT," *IEEE Trans. Veh. Technol.*, early access, Jan. 23, 2023, doi: 10.1109/TVT.2023.3238771.

[6] Y. Hao, M. Chen, H. Gharavi, Y. Zhang, and K. Hwang, "Deep reinforcement learning for edge service placement in softwarized industrial cyber-physical system," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5552–5561, Aug. 2021.

[7] L. Liu, X. Qin, Y. Tao, and Z. Zhang, "Timely updates in MEC-assisted status update systems: Joint task generation and computation offloading scheme," *China Commun.*, vol. 17, no. 8, pp. 168–186, Aug. 2020.

[8] X. He, S. Wang, X. Wang, S. Xu, and J. Ren, "Age-based scheduling for monitoring and control applications in mobile edge computing systems," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2022, pp. 1009–1018.

[9] Z. Fang, J. Wang, Y. Ren, Z. Han, H. V. Poor, and L. Hanzo, "Age of information in energy harvesting aided massive multiple access networks," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 5, pp. 1441–1456, May 2022.

[10] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *Proc. 31st Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Mar. 2012, pp. 2731–2735.

[11] X. Chen et al., "Information freshness-aware task offloading in air-ground integrated edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 1, pp. 243–258, Jan. 2022.

[12] H. He, H. Shan, A. Huang, Q. Ye, and W. Zhuang, "Edge-aided computing and transmission scheduling for LTE-U-enabled IoT," *IEEE Trans. Wireless Commun.*, vol. 19, no. 12, pp. 7881–7896, Dec. 2020.

[13] C. Wang, X. Yu, L. Xu, and W. Wang, "Energy-efficient task scheduling based on traffic mapping in heterogeneous mobile-edge computing: A green IoT perspective," *IEEE Trans. Green Commun. Netw.*, vol. 7, no. 2, pp. 972–982, Jun. 2023, doi: 10.1109/TGCN.2022.3186314.

[14] Y. Hao, L. Hu, and M. Chen, "Joint sensing adaptation and model placement in 6G fabric computing," *IEEE J. Selected Areas Commun.*, early access, Jun. 2023, doi: 10.1109/JSAC.2023.3280968.

[15] Q. Kuang, J. Gong, X. Chen, and X. Ma, "Analysis on computation-intensive status update in mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4353–4366, Apr. 2020.

[16] X. Ling, J. Gong, R. Li, S. Yu, Q. Ma, and X. Chen, "Dynamic age minimization with real-time information preprocessing for edge-assisted IoT devices with energy harvesting," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 3, pp. 2288–2300, Jul.–Sep. 2021.

[17] L. Liu, X. Qin, X. Xu, H. Li, F. R. Yu, and P. Zhang, "Optimizing information freshness in MEC-assisted status update systems with heterogeneous energy harvesting devices," *IEEE Internet Things J.*, vol. 8, no. 23, pp. 17057–17070, Dec. 2021.

[18] H. Chen, X. Qin, Y. Li, and N. Ma, "OEnergy-aware path planning for obtaining fresh updates in UAV-IoT MEC systems," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2022, pp. 1791–1796, doi: 10.1109/WCNC51071.2022.9771867.

[19] I. Kadota and E. Modiano, "Minimizing the age of information in wireless networks with stochastic arrivals," *IEEE Trans. Mobile Comput.*, vol. 20, no. 3, pp. 1173–1185, Mar. 2021.

[20] R. Talak, S. Karaman, and E. Modiano, "Optimizing information freshness in wireless networks under general interference constraints," *IEEE/ACM Trans. Netw.*, vol. 28, no. 1, pp. 15–28, Feb. 2020.

[21] Z. Jiang, B. Krishnamachari, X. Zheng, S. Zhou, and Z. Niu, "Timely status update in wireless uplinks: Analytical solutions with asymptotic optimality," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3885–3898, Apr. 2019.

[22] H. Chen, Y. Gu, and S.-C. Liew, "Age-of-information dependent random access for massive IoT networks," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Jul. 2020, pp. 930–935.

[23] X. Chen et al., "Age of information aware radio resource management in vehicular networks: A proactive deep reinforcement learning perspective," *IEEE Trans. Wireless Commun.*, vol. 19, no. 4, pp. 2268–2281, Apr. 2020, doi: 10.1109/TWC.2019.2963667.

[24] R. V. Bhat, R. Vaze, and M. Motani, "Minimization of age of information in fading multiple access channels," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 5, pp. 1471–1484, May 2021.

[25] Z. Qian, F. Wu, J. Pan, K. Srinivasan, and N. B. Shroff, "Minimizing age of information in multi-channel time-sensitive information update systems," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Jul. 2020, pp. 446–455.

[26] B. Li, T. Shi, J. Chen, S. Li, Y. Wang, and T. Zhang, "Age of information updates in non-orthogonal multiple access-mobile edge computing system based on reinforcement learning," *J. Electron. Inf. Technol.*, vol. 44, no. 12, pp. 4238–4245, 2022.

[27] B. Li, W. Wu, X. Duan, and Y. Qi, "Research on the minimizing of information age based on NOMA-MEC system," *J. Commun.*, vol. 42, no. 10, pp. 222–232, 2021.

[28] R. Li, Q. Ma, J. Gong, Z. Zhou, and X. Chen, "Age of processing: Age-driven status sampling and processing offloading for edge-computing-enabled real-time IoT applications," *IEEE Internet Things J.*, vol. 8, no. 19, pp. 14471–14484, Oct. 2021.

[29] X. Xie, H. Wang, and M. Weng, "A reinforcement learning approach for optimizing the age-of-computing-enabled IoT," *IEEE Internet Things J.*, vol. 9, no. 4, pp. 2778–2786, Feb. 2022.

[30] R. V. Bhat, R. Vaze, and M. Motani, "Minimization of age of information in fading multiple access channels," in *Proc. 4th Int. Conf. Learn. Rep. (ICLR)*, May 2016, pp. 1–4.

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

[32] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with Gumbel–Softmax," in *Proc. 5th Int. Conf. Learn. Rep. (ICLR)*, Apr. 2017, pp. 1–13.

[33] V. Hasselt, Hado, A. Guez, and D. Silver, "Deep reinforcement learning with double *Q*-learning," in *Proc. 30th AAAI Conf. Artif. Intell. (AAAI)*, vol. 30, 2016, Art. no. 10295.

[34] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling network architectures for deep reinforcement learning," in *Proc. 33rd Int. Conf. Mach. Learn. (PMLR)*, Jun. 2016, pp. 1995–2003.

[35] J. Chen, Y. Wang, and T. Lan, "Bringing fairness to actor–critic reinforcement learning for network utility optimization," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2021, pp. 1–10.