IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 4, 2021, accepted August 28, 2021, date of publication August 30, 2021, date of current version September 9, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3109132

Deep Reinforcement Learning and Game Theory for Computation Offloading in Dynamic Edge Computing Markets

SHUYANG LI, XIAOHUI HU, AND YONGWEN DU
School of Electronic and Information Engineering, Lanzhou Jiaotong University, Lanzhou 730070, China
Corresponding author: Xiaohui Hu (0619680@mail.lzjtu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 11461038, and in part by the Innovation Foundation of Colleges and Universities in Gansu Province under Grant 2020A-033.

ABSTRACT As a promising paradigm, computation offloading technology can offload computing tasks to multi-access edge computing (MEC) servers, which is an appealing choice for resource-constrained terminal devices to reduce their computational effort. However, due to limited resources, one crucial research challenge for computation offloading is to design the appropriate offloading policy to determine which tasks should be offloaded in some complex circumstances. In this paper, we study the offloading decision problem in a software-defined networking (SDN) driven MEC environment with multiple users and multiple servers. To ensure that end-users do not abuse the computing resources in the MEC system, we formulate the profit of MEC servers as our optimization objective. We jointly optimize the selection of MEC servers, the size of offloading data, and the price of MEC computing service to maximize the profit of MEC servers. However, considering the dynamic and stochastic of end-users, it is challenging to obtain the optimal policy in such a MEC environment. We apply deep reinforcement learning (DRL) and Game theory to our proposed approach. Specifically, we propose a proximal policy optimization (PPO) reinforcement learning framework to tackle the selection of MEC servers. Secondly, a two-step optimization problem is formulated to determine the size of offloading data and the pricing of computing services. The optimal values of those two were determined by achieving the Nash equilibrium of the strategy game between end-users. Extensive simulation results prove that our proposal has a better performance than existing solutions in convergence time and stability.

INDEX TERMS Edge computing, offloading decision, Markov decision process, deep reinforcement learning, game theory, Nash equilibrium.

I. INTRODUCTION

Driven by the Internet of Things (IoT) and 5G networks, mobile computing has seen a paradigm shift recently, from centralized cloud computing to multi-access edge computing (MEC) [1]. Because of the advantages of low latency, mobile energy saving, context awareness, privacy/security enhancement, etc., edge computing has stimulated extensive efforts in academia and industry to develop this technology [2]. The rapid development of MEC technology has paved the way for integrating MEC servers as intelligent entities into IoT and 5G networks [3]. The architecture of the multi-access edge computing network is depicted in Figure 1.

Although a series of proposed MEC offloading approaches can significantly enhance user's computing power, developing an efficient and reliable edge computing system remains challenging. Those challenges include architecture design, mobility management, security, QoS and QoE compliant services, content caching, and computation offloading [4]. Due to some growing problems, including but not limited to limitations of resources, dynamic and stochastic of the end-users, scalability, reliability, security, and privacy, the offloading decision is considered one of the most challenging problems of MEC. Therefore, designing an efficient offloading approach is necessary to determine which tasks should be

The associate editor coordinating the review of this manuscript and approving it for publication was Chakchai So-In.

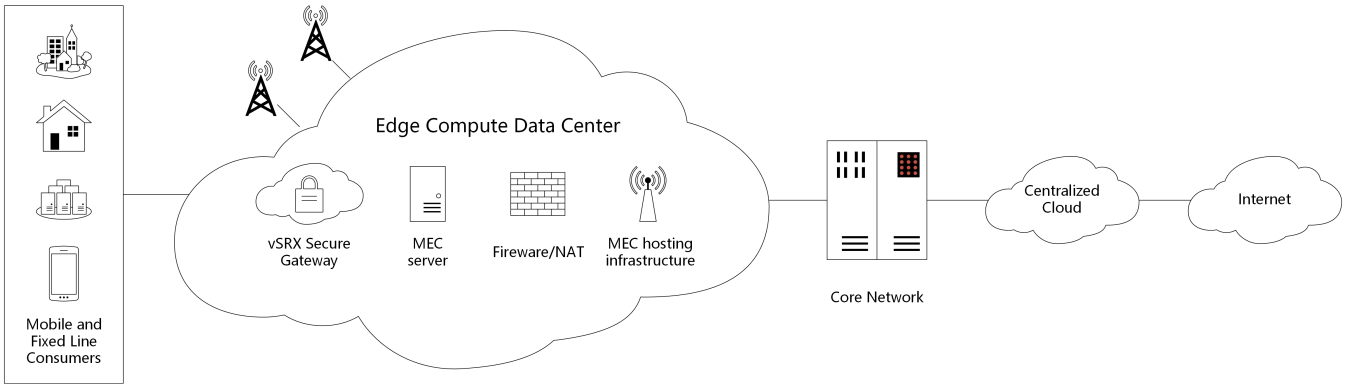


FIGURE 1. The architecture of multi-access edge computing network.

offloaded to the MEC servers. However, most of the existing work could only achieve the entire operation of the system by centralized control [5], [6].

At present, more and more literature [7]–[9] shows that researchers apply reinforcement learning (RL) to the offloading decision in the MEC environment, which is suitable for the dynamic and stochastic of mobile devices. However, most RL-based offloading approaches are designed by Q-learning and Deep Q-Network algorithms, which do not have good performances in a complex MEC offloading environment. In RL-based approaches, it takes time for the agent to explore and learn the policy to take appropriate action to get higher cumulative rewards, depending on the state-space of the problems. If the dimensions of states and actions increase, exploration and learning will become considerably time-intensive [10].

Game theory is an effective theoretical framework to analyze the interaction between different participants acting on their benefits. Since the game theory approaches explore the best policy in the current situation, which seems to be more suitable for the dynamic behavior of mobile equipment, any participant has no incentive to deviate unilaterally. Due to its broad prospects, recent research applies game theory to edge computing offloading decisions in the MEC environment. Game theory provides end-users with enhanced flexibility and allows users to make their computing offloading decisions autonomously in a distributed manner [11], making a decentralized computing offloading approach possible. In addition, in the game theory approach, it is assumed that end-users can effectively make decisions based on local observations without obtaining global information with low complexity for optimization purposes. Moreover, game theory approaches can be easily analyzed by ubiquitous mathematical tools [12]. The main advantage of game theory lies in its distributed nature, which can match multi-user offloading scenarios. However, game theory can only guarantee the Nash equilibrium, which may not be the global optimal solution.

Edge computing is a physical infrastructure that can bring computing, storage, and energy resources closer to the end-user, device, and source of data which significantly reduces latency and the computational effort of end-devices. However, we have found that deploying computing infrastructure near end-users cannot solve all technical challenges well. In recent years, software-defined networking (SDN), as another significant trend in networking, is taken into account by tons of researchers [13]. SDN is by far the most promising proposal for programmable networks. It separates the control plane from the data plane and enables programmable control mechanisms [14]. The control mechanism provided by SDN can reduce the complexity of network architecture, which is suitable for the MEC environment. In [15], the capabilities of SDN are efficiently and effectively used to address computation offloading problems such as the selection of MEC server, the routing of offloading data, announcing pricing, and controlling the smooth completion of the MEC system operations in the MEC environment. In [16], a fog computing architecture based on SDN was proposed to select a computing mode for each end user's computing task, determining where the tasks are executed.

This paper fills the research as mentioned above gaps by proposing an approach based on Game theory and deep reinforcement learning (DRL). However, challenges arise due to partial offloading in factual circumstances. Specifically, we consider that task is partible when we model the offloading problem, which is different from the simple binary offloading mode considered in existing work [17]. It is noted that our study refers to the model proposed by existing research [18].

The main contributions of our proposal that differentiate it from the existing works are summarized as follows.

1) Our proposal studies the offloading decision-making problem in the SDN-driven MEC environment with multi-user and multi-server. We formulate maximizing the MEC servers' profit as the optimization objective.
2) To maximize the profit of the MEC server, we jointly optimize the selection of the MEC server, the size of offloading data, and the price of the MEC computing service. We introduce the PPO-based dynamic computation offloading algorithm (PDCO) to optimize the offloading policy.

3) Specifically, we propose a PPO-based RL framework to tackle the selection of MEC servers. The problem is further formulated as a Markov decision process (MDP), with well-designed expressions of state, action, and reward to represent the environment features. Secondly, a two-step optimization problem is formulated to determine the size of offloading data and the pricing of computing services. We solve the two-step optimization problem by formulating a non-cooperative game between end-users.
4) A series of detailed simulation results prove our proposal's advantage in convergence time and stability by comparing the PDCO algorithm with other RL-based approaches.

The remainder of this paper is organized as follows. Section 2 briefly reviews the related literature. Section 3 provides the system model and problem formulation. Section 4 gives the algorithm design and provides detailed work of the algorithm. Section 5 evaluates the proposed approach performance by a series of simulations. Section 6 concludes the paper.

II. RELATED WORK

As a promising paradigm, MEC brings computing, storage, and energy resources closer to the network's edge. The concept of MEC was first proposed by the ETSI in 2014, which aims at providing ultra-low latency and ultra-high bandwidth for latency-sensitive applications [19], [20]. The emergence of new technologies, especially 5G technology, has promoted the continuous prosperity and development of MEC technology. As mentioned above, 5G technology defines three typical usage scenarios: enhanced mobile broadband, ultra-reliable low-latency communication, and massive machine-type communication [21]. These scenarios are consistent with edge computing, and these features also provide significant advantages for computing offloading.

The offloading decision is one of the most challenging problems in the field of computation offloading. Different approaches have been proposed recently to help optimize offloading policy regarding whether, where, and how much to offload to improve the efficiency of the offloading process, which is crucial to computation offloading. Shakarami *et al.* [10], [22], [23] have systematically reviewed the computation offloading approaches from the perspectives of machine learning, Game theory, and stochastic, respectively. From the perspective of single users, many existing works have studied computation offloading in the early days. Computation offloading can save energy was proven by Redenko *et al.* [24] according to their experimental results. A hybrid method for computation offloading was proposed to improve energy saving and minimize service latency by Xian *et al.* [25]. More research currently has focused on the MEC offloading environment with multiple users and multiple servers. Moreover, the objective of MEC is usually to optimize system delay [26]–[28], energy consumption [29]–[31], and the weighted sum of delay and energy consumption [32]–[34].

Although all the above works have shown the benefits and potential of the MEC technology, they assume that the MEC server provides free computing and communication services for the users in the MEC environment, which is usually unrealistic. In the multi-user multi-server MEC environment, there will be a situation where a large number of users compete for a limited quantity of servers, which will cause the problem of uneven distribution of computing resources. Some recent works [35]–[37], by setting prices for MEC computing service and formulating a distributed offloading game between end-users to solve offloading decision problems. Nevertheless, they mainly focus on offloading problems in a relatively static MEC environment. Those proposed approaches may not be suitable for real network environments affected by various factors.

As mentioned above, RL or game theory alone may not obtain the globally optimal policy. Therefore, some existing studies combine RL and game theory to solve the problems related to offloading decision-making in the MEC environment. In [38], the problem of MEC server activation was formulated as a minority game, and RL was used to solve the Nash equilibrium of non-cooperative games. [39] studied the problem of security in the process of offloading decisions based on RL and game theory, which derive the optimal offloading rates and reduce the attack rate of smart attackers at IoT devices. In [40], Zhan *et al.* studied the computation offloading game in incomplete information sharing based on the PPO algorithm. Unlike previous studies, our proposal formally solves the problem of uneven allocation of computing resources, the offloading decision for complex state space problems, and the challenge of achieving Nash equilibrium in the dynamic MEC environment. This joint optimization problem is non-trivial, and we describe our proposed approach in detail in the following sections.

III. SYSTEM MODEL AND PROBLEM FORMULATION

A. SYSTEM MODEL

The SDN-driven MEC system architecture with multiple servers and multiple users is shown in Figure 2. Without loss of generality, we consider a group of end-users u, where $u \in \mathcal{U}, \mathcal{U} = [1, \ldots, u, \ldots, U]$ and a group of MEC servers s, where $s \in \mathcal{S}, \mathcal{S} = [1, \ldots, s, \ldots, S]$. We divide the continuous decision period into several time slots t, where $t \in \mathcal{T}, \mathcal{T} = [1, \ldots, t, \ldots, T]$ denotes the corresponding set. Each end-user needs to execute computation-intensive and delay-sensitive tasks periodically at each time slot. In general, end-users will instinctively select MEC servers with strong computing performance to improve their own QoE. Since a large number of users compete for a limited quantity of servers in the multi-user and multi-server MEC environment, we establish a computing market model based on SDN-driven multi-server competition. In this model, the MEC server provides computing services to end-users and charges users a

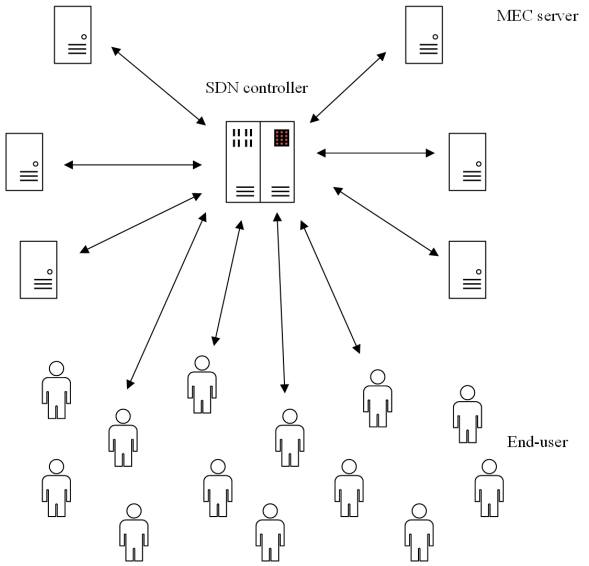


FIGURE 2. The SDN-driven MEC system architecture with multiple servers and multiple users.

certain form of fee. The SDN controller acts as an intermediary between servers and end-users to process information and makes decisions. The SDN controller executes our proposed PDCO algorithm at each time slot, which is equipped with a strong computing power CPU, and advanced computing power makes fast decision-making possible. Unless otherwise stated, Table 1 summarizes the important notations used in this paper.

Following the model in [18], the whole execution of the SDN-driven MEC system is divided into several time slots. During each time slot, the selection of the MEC server, the size of offloading data, and the price of the MEC computing service are determined by the SDN controller. Initially, each MEC server sends the price information of the computing service to SDN to announce its price $p_s^{(t)}[\$/bit]$ to end-users. Meanwhile, each end-user receives information through the SDN controller and offloads its data $b_{u,s}^{(t)}[bit]$ to the selected MEC server. In each time slot t, each end-user has a total of $I_u^{(t)}$ computing tasks to execute. Due to partial offloading mode, the part of computing tasks is executed on the specified MEC server (i.e., $b_{u,s}^{(t)} \in \left[0, I_u^{(t)}\right]$), while the rest of the tasks are executed locally(i.e., $I_u^{(t)} - b_{u,s}^{(t)}$).

As mentioned above, we assume that a total of $I_u^{(t)}$ computing tasks need to be executed by each end-user u at the time slot t. Consider the reasonable allocation of MEC server computing resources, and the SDN controller needs to determine the optimal size of offloaded data $b_{u,s}^{(t)}$ for the end-user u. The size of data offloaded by other users except u is denoted by $b_{-u}^{(t)}$ and the relative size of offloading data is

$$r_u^{(t)} = \frac{b_{u,s}^{(t)}}{b_{-u}^{(t)}} \tag{1}$$

Here, we use a logarithmic function on the relative offload $r_u^{(t)}$ of the end-user u to capture the perceived satisfaction (i.e., QoE) of the end-user u, which is denoted as

$$s_u^{(t)}\left(b_{u,s'}^{(t)} b_{-u}^{(t)}\right) = \alpha_u \log\left(1 + \beta_u r_u^{(t)}\right) \tag{2}$$

where parameters $\alpha_u, \beta_u \in R^+$ are adjustable parameters which determine the slope of $s_u^{(t)}$ in a personalized manner. The offloading cost of end-user u according to the relative size of offloading data $r_u^{(t)}$ is denoted as

$$c_u^{(t)}\left(b_{u,s}^{(t)}, b_{-u}^{(t)}\right) = d_u^{(t)} p_s^{(t)} r_u^{(t)} \tag{3}$$

where $d_u^{(t)} \in R^+$ denotes dynamic behavior of end-user u and $p_s^{(t)}$ denotes the computing service price of MEC server s. According to the mentioned above, the utility function of each end-user u is denoted as

$$\begin{aligned} U_u^{(t)}\left(b_{u,s}^{(t)}, b_{-u}^{(t)}, p^{(t)}\right) &= s_u^{(t)}\left(b_{u,s}^{(t)}, b_{-u}^{(t)}\right) - c_u^{(t)}\left(b_{u,s}^{(t)}, b_{-u}^{(t)}\right) \\ &= \alpha_u \log\left(1 + \beta_u r_u^{(t)}\right) - d_u^{(t)} p_s^{(t)} r_u^{(t)} \end{aligned} \tag{4}$$

We need to define a parameter $R_s^{(t)}$ in the MEC server system to represent the reputation of the MEC server s, which changes dynamically at each time slot. This parameter is used to update the probability that the user selects a specific server. We consider the following aspects.

First of all, MEC servers provide some discount $f_s^{(t)}$ to encourage end-users to give preference to select a specific MEC server. Those discounts are indicated as a percentage of the original price announced by the MEC servers. In addition, the MEC server needs to consider the computation cost $c_s^{(t)}[\$/bit]$ of the MEC server when processing computing tasks it receives.

Secondly, If the MEC servers can well meet the computing needs of end-users, those servers will win a good reputation among end-users. Specifically, we define the penetration rate of the MEC server as the total size of computing data processed by the MEC server that accounts for the total size of computing data processed in the SDN-driven MEC system.

Furthermore, we define a parameter to show the current congestion level of the server, which is denoted by $CONG_s$. We assume that the maximum size of offloading each MEC server can handle that is characterized by B_s^{max}. Therefore, the congestion level can be expressed as the percentage of the size of offloaded data by the end-user to the maximum capacity of the MEC server.

Following all the above analysis, the reputation score of MEC server $R_s^{(t)}$ could be denoted as

$$\begin{aligned} R_s^{(t)} = w_1 \frac{\frac{\sum_{k \neq s}\left[\left(1 - f_k^{(t)}\right)\right] p_k^{(t)}}{K}}{\left(1 - f_s^{(t)}\right) p_s^{(t)}} + w_2 \frac{1}{(1 + \mathrm{CONG}_s)^3} \\ + w_3 \frac{\sum_{t \in \{1,\ldots,T\}} \sum_{u \in U} b_{u,s}^{(t)}}{\sum_{s \in S} \sum_{t \in \{1,\ldots,T\}} \sum_{u \in U} b_{u,s}^{(t)}} \end{aligned} \tag{5}$$

The total revenue of each MEC server s is determined by the service price $p_s^{(t)}$, the corresponding discount $f_s^{(t)}$, and the

TABLE 1. List of notations.

Notations	Definitions
$\mathcal{U}$	The set of end-user u
$\mathcal{S}$	The set of MEC server s
$\mathcal{T}$	The set of time slot t
$p_s^{(t)}$	The service's price of server s at time slot t
$I_u^{(t)}$	The maximum size of data that each user u needs to process at time slot t
$b_{u,s}^{(t)}$	The size of data offloaded by user u to server s at time slot t
$b_{-u}^{(t)}$	The size of data offloaded by other users except u at time slot t
$r_u^{(t)}$	The relative size of offloaded by end-user u at time slot t
$s_u^{(t)}$	Quality of Experience of end-user u at time slot t
$c_u^{(t)}$	Cost of end-user u at time slot t
$U_u^{(t)}$	Utility function of end-user u at time slot t
$R_s^{(t)}$	Reputation of server s at time slot t
$REV_s^{(t)}$	Revenue of server s at time slot t
$C_s^{(t)}$	Computation cost of server s at time slot t
$P_s^{(t)}$	Profit of server s at time slot t
$f_s^{(t)}$	Discount of server s at time slot t
$A_u^{(t)}$	The strategy space of user u at time slot t

size of offloading data $b_{u,s}^{(t)}$, which is denoted as

$$REV_s^{(t)}\left(b^{(t)}, p^{(t)}\right) = \left(1 - f_s^{(t)}\right) p_s^{(t)} \sum_{u \in U} b_{u,s}^{(t)} \tag{6}$$

Moreover, we should also consider the computing cost of the server, which is denoted as

$$C_s^{(t)}\left(b^{(t)}\right) = c_s^{(t)} \sum_{u \in U} b_{u,s}^{(t)} \tag{7}$$

According to the above analysis, the total profit of each MEC server s is denoted as

$$\begin{aligned} P_s^{(t)}\left(b^{(t)}, p^{(t)}\right) &= REV_s^{(t)}\left(b^{(t)}, p^{(t)}\right) - C_s^{(t)}\left(b^{(t)}\right) \\ &= \left(1 - f_s^{(t)}\right) p_s^{(t)} \sum_{u \in U} b_{u,s}^{(t)} - c_s^{(t)} \sum_{u \in U} b_{u,s}^{(t)} \end{aligned} \tag{8}$$

B. PROBLEM FORMULATION

Through the PPO-based MEC server selection algorithm, each end-user is given the selection of the MEC server. From the end-user's perspective, the objective of each end-user is to maximize its utility by offloading the optimal data size to the selected MEC server. Similarly, each MEC server aims to maximize its profit by executing the offloaded tasks. Therefore, a two-step optimization is formulated as

$$b^{(t)*} = \text{argmax}_{b_{u,s}^{(t)}}\, U_u^{(t)}\left(b_{u,s}^{(t)}, b_{-u}^{(t)}, p^{(t)}\right) \tag{9}$$

$$p^{(t)*} = \text{argmax}_{p^{(t)}}\, P_s^{(t)}\left(b^{(t)}, p^{(t)}\right) \tag{10}$$

From the above equations, the optimal price $p^{(t)*}$ and the optimal size of offloading data $b^{(t)*}$ are interdependent. We apply DRL and Game theory to solve the two-step optimization problem, described in detail in the following sections.

IV. PPO-BASED DYNAMIC COMPUTATION OFFLOADING ALGORITHM

A. MARKOV DECISION PROCESS FORMULATION

In our proposal, the process of offloading decision is formulated as a Markov decision process (MDP), and we consider the standard RL framework [41], in which the agent interacts with the environment. In general, a MDP can be described as a quintuple $M = <\text{S,A,R,P},\gamma>$. State, action, and reward at each time slot t are denoted as $s_t \in S$, $a_t \in A$, and $r_t \in \text{R}$, respectively. The dynamics of the environment are represented by the state transition probability $p_{ss'}^a = \Pr\left\{s_{t+1} = s' \mid s_t = s, a_t = a\right\}$ and expected reward $R_a\left(s, s'\right) = \mathbb{E}\left\{r_{t+1} \mid s_t = s, a_t = a\right\}$, where $\forall s, s' \in S, a \in A$. We introduce how we define state space, action space, and reward as follows.

- State space: We take the probability of selecting the MEC server in each time slot as the current state. In the initial state, the end-users have the same probability of selecting each MEC server, determined by the number of MEC servers. Since the state space, action space, and reward are too simple, no longer denoted by notations here.
- Action space: The behavior of selecting MEC server as the action space of the DRL algorithm. Therefore, the size of the action space also depends on the number of MEC servers.
- Reward: The MEC server selected by the end-user will be compared with the MEC server with the highest reputation score. If the end-user chooses the MEC server with the highest reputation score, rewards will be given. Otherwise, it will be punished.

B. MEC SERVER SELECT ALGORITHM BASED ON PPO

To optimize offloading policy by policy gradient DRL algorithms, we parameterize the computation offloading policy θ_u of user u as π_{θ_u}. The policy gradient algorithms aim at finding a set of optimal parameters θ_u^*, which maximizes the expectation of cumulative rewards. Therefore, the optimization objective of the policy gradient is

$$\begin{aligned} \theta_u^* &= \arg\max_{\theta_u} L_u\left(\pi_{\theta_u}\right) \\ &= \arg\max_{\theta_u} \text{E}_{\theta_u}\left(V_{\pi_{\theta_u}}(s)\right) \\ &= \arg\max_{\theta_u} \text{E}_{\theta_u}\left(Q_{\pi_{\theta_u}}(s,a)\right) \end{aligned} \tag{11}$$

where

$$V_{\pi_{\theta_u}}(s) = \mathbb{E}_{\pi_{\theta_u}}\left[R_u^t \mid S_u^t = s_u\right] \tag{12}$$

$$Q_{\pi_{\theta_u}}(s,a) = \mathbb{E}_{\pi_{\theta_u}}\left[R_u^t \mid S_u^t = s_u, A_u^t = a_u\right] \tag{13}$$

$$R_u^t = \sum_{t=0}^{T} \gamma * r_u^t \tag{14}$$

In the above equation, the set of all users' policies is denoted by $\Pi = \left\{\pi_{\theta_u}\right\}_{u \in U}$, state-value function is denoted by $V_{\pi_{\theta_u}}(s)$, action-value function is denoted by $Q_{\pi_{\theta_u}}(s,a)$,

cumulative rewards of user u at time slot t is denoted by R_u^t and $\gamma \in [0, 1]$ is discount factor.

According to the [41] and [42], the policy gradient could be calculated as

$$\begin{aligned}\nabla_{\theta_u} L\left(\pi_{\theta_u}\right) &= \mathbb{E}_{\pi_{\theta_u}}\left[\nabla_{\theta_u} \log_{\pi_{\theta_u}} Q^{\pi_{\theta u}}(s,a)\right] \\ &= \mathbb{E}_{\pi_{\theta_u}}\left[\nabla_{\theta_u} \log_{\pi_{\theta_u}} A^{\pi_{\theta u}}(s,a)\right] \\ &\approx \mathbb{E}_{\pi_{\theta_u}}\left[f_u \nabla_{\theta_u} \log_{\pi_{\theta_u}} A^{\pi_{\theta u}}(s,a)\right] \end{aligned} \tag{15}$$

where $f_u = \frac{\pi_{\theta_u}}{\pi_{\hat{\theta}_u}}$, $A_{\pi_{\theta u}}(s,a) = Q_{\pi_{\theta_u}}(s,a) - V_{\pi_{\theta_u}}(s)$ is the advantage function for action and state, $\hat{\theta}_u$ represents the parameter of policy for sampling. We adapt the proximal policy optimization [43] to accelerate the convergence, which clips the policy gradient as

$$\nabla_{\theta_u} L\left(\pi_{\theta_u}\right) \approx \mathbb{E}_{\pi_{\hat{\theta}_u}}\left[\nabla_{\theta_u} \log_{\pi_{\theta_u}} C(s,a)\right] \tag{16}$$

where

$$C(s,a) = \min\left[f_u A^{\pi_{\theta u}}(s,a), \eta\left(f_u\right) A^{\pi_{\theta u}}(s,a)\right] \tag{17}$$

$$\eta(x) = \begin{cases} 1+\varepsilon, & x > 1+\varepsilon \\ x, & 1-\varepsilon \le x \le 1+\varepsilon \\ 1-\varepsilon, & x < 1-\varepsilon \end{cases} \tag{18}$$

where the parameter ε is set to 0.2 in our proposal.

In this model, we design an actor and critic network for each end-user. We design the actor network for approximating the policy and the critic network for approximating the value function, which is denoted by π_{θ_u} and V_{ω_u}, where θ_u and ω_u represent the parameters of actor network and critic network, respectively.

We define the loss function for updating the critic network V_{ω_u} as

$$J_u\left(\omega_u\right) = \mathbb{E}_{\pi_{\omega_u}}\left[-V_{\omega_u}(s) + \mathbb{E}_{\pi_{\omega_u}}\left[r + V_{\omega_u}\left(\mathrm{s}'\right)\right]\right]^2 \tag{19}$$

During the training process, the estimated gradient about ω_u is calculated as

$$\nabla_{\omega_u} \hat{J}_{\mathrm{u}} = \frac{1}{D} \sum_{k=0}^{D-1} \left[V_{\omega_u}(s) - Y_{\mathrm{u}}^k\right] \frac{dV_{\omega_u}(s)}{d\omega_u} \tag{20}$$

where

$$Y_u^k = R_u^k - \gamma^{D-k} R_u(D) + \gamma^{D-k} V_{\omega_u}(D) \tag{21}$$

And D is the size of a mini-batch for updating. Similarly, the estimated gradient about θ_u is calculated as

$$\nabla_{\theta_u} \hat{L}_u = \frac{1}{D} \sum_{t=0}^{D-1} \nabla_{\theta_u} \log_{\pi_{\theta_u}} C\left(s^t, a^t\right) \tag{22}$$

We update π_{θ_u} by mini-batch stochastic gradient ascent and update V_{ω_u} by mini-batch stochastic gradient descent.

$$\theta_u \leftarrow \theta_u + l_{u,2} \nabla_{\theta_u} \hat{L}_u \tag{23}$$

$$\omega_u \leftarrow \omega_u - l_{u,1} \nabla_{\omega_u} \hat{J}_u \tag{24}$$

where $l_{u,2}$ is the learning rate of actor network and $l_{u,1}$ is the learning rate of critic network.

The pseudocode of the MEC server select algorithm based on PPO is given in Algorithm 1.

Algorithm 1 MEC Server Select Algorithm Based on PPO

```
for each user u ∈ U do
    Initialize γ, l_{u,1}, l_{u,2}, ω_u, and θ_u
end for
for each time slot t ∈ T do
    for each user u ∈ U do
        Run policy π^t_{θ_u} for time slot t, collect s_t, a_t, r_t
        Estimate advantages A^t_{π_{θ_u}}(s_t, a_t)
        π^t_{old_{θ_u}} ← π^t_{θ_u}
    end for
    if t%D == 0 then
        for each user u ∈ U do
            Calculate ∇_{ω_u} Ĵ_u and ∇_{θ_u} L̂_u via (20) and (22)
            Update θ_u and ω_u via (23) and (24)
        end for
    end if
end for
```

C. DECENTRALIZED COMPUTATION OFFLOADING GAME

In this section, we formulate a strategy game for solving the two-step optimization problem to obtain the optimal size of offloading data and the optimal price. As a robust framework for computation offloading problems, game theory can analyze the interaction between game players acting in their interests, thereby optimizing offloading strategy so that no end-user has the purpose to deviate unilaterally.

1) GAME FORMULATION

Let $b_{-u}^{(t)} = \left(b_1^{(t)}, \ldots, b_{u-1}^{(t)}, b_{u+1}^{(t)}, \ldots, b_U^{(t)}\right)$ be the size of offloading data by all other users except end-user u, where $t \in \mathcal{T}, \mathcal{T} = (1, \ldots, t, \ldots, T)$ denotes the corresponding set of time slots. Given other end-users' offloading decision $b_{-u}^{(t)}$, end-user u would like to make appropriate decision $b_{u,s}^{(t)}$ to maximize QoE (capture by personal utility) while considering the given constraints, i.e.,

$$\max_{b_{u,s}^{(t)}} U_u^{(t)}\left(b_{u,s}^{(t)}, b_{-u}^{(t)}, p^{(t)}\right) \tag{25}$$

$$\text{s.t. } 0 \le b_{u,s}^{(t)} \le I_u^{(t)} \tag{26}$$

We then formulate the problem above as a strategy game $G = \left[\mathcal{U}, \left\{A_u^{(t)}\right\}, \left\{U_u^{(t)}\right\}\right]$ to solve the optimal size of offloading data $b_{u,s}^{(t)}$, where $\mathcal{U}$ denotes the corresponding set of the end-users participant in the game G, $A_u^{(t)}$ denotes the strategy space of end-user u as well as $U_u^{(t)}$ denotes utility function of end-user u. The strategy game G is a non-cooperative game, our objective aims to design an algorithm to achieve the Nash equilibrium [44] of the non-cooperative game G, and

the definition of Nash equilibrium is given in the following details.

Definition 1: (Nash equilibrium) If each end-user $u \in U$ satisfies

$$U_u^{(t)}\left(b_{u,s}^{(t)*}, b_{-u}^{(t)*}\right) \geq U_u^{(t)}\left(b_{u,s}^{(t)}, b_{-u}^{(t)*}\right), \forall b_{u,s}^{(t)} \in A_u^{(t)} \quad (27)$$

The set of computation offloading strategy $b_u^{(t)*} = \left[b_{1,s}^{(t)*}, \ldots, b_{u,s}^{(t)*}, \ldots, b_{U,s}^{(t)*}\right], \forall t \in \mathcal{T}, u \in \mathcal{U}, s \in \mathcal{S}$ *is the Nash equilibrium point of game* $G = \left[U, \left\{A_u^{(t)}\right\}, \left\{U_u^{(t)}\right\}\right]$. We then study the existence of Nash equilibrium of the strategy game $G = \left[U, \left\{A_u^{(t)}\right\}, \left\{U_u^{(t)}\right\}\right]$. And to proceed, we introduce an important concept as follows, which is called Best Response [44].

Definition 2: (Best Response) At time slot t, given other end-users' strategies $b_{-u}^{(t)}$*, end-user u's strategy* $b_u^{(t)*}$ *is the Best Response, if*

$$U_u\left(b_u^{(t)*}, b_{-u}^{(t)}\right) \geq U_u\left(b_u^{(t)}, b_{-u}^{(t)}\right) \quad (28)$$

We analyze the existence of Nash equilibrium, and the proof of the existence of Nash equilibrium is given in APPENDIX.

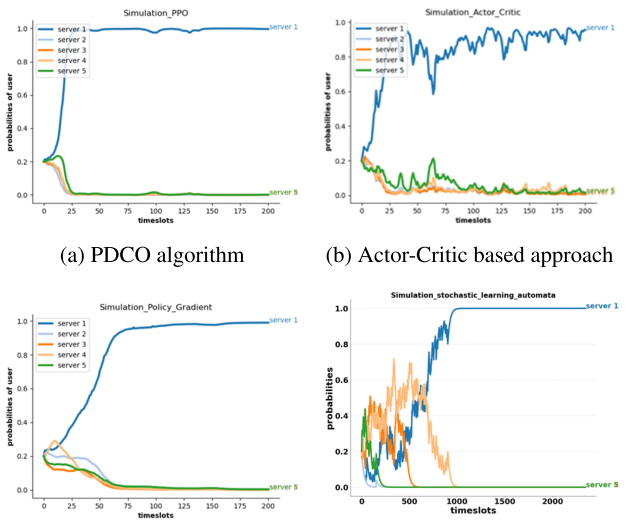
2) PPO-BASED DYNAMIC COMPUTATION OFFLOADING ALGORITHM

According to (27) and (28), all the end-users play the best response strategies towards each other at the Nash equilibrium. We design a low complexity PPO-based dynamic computation offloading algorithm to achieve the Nash equilibrium of the strategy game G, and the pseudocode of the PDCO algorithm is given in Algorithm 2.

Algorithm 2 PPO-based dynamic computation offloading algorithm (PDCO)

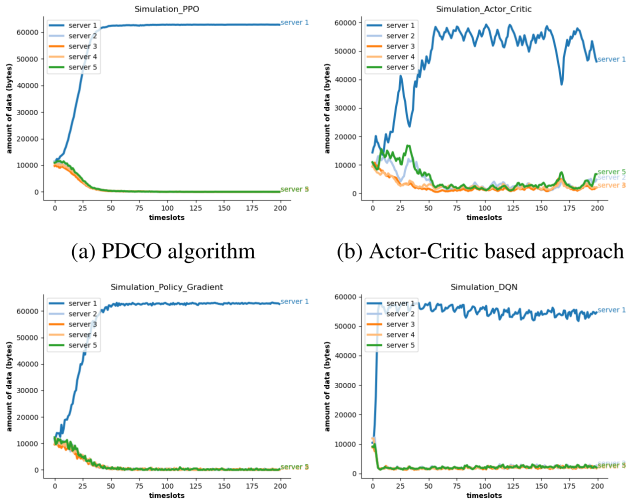
1: **for** each user $u \in U$ **do**
2: Initialize probability and $p_s^{(t)}$.
3: **end for**
4: **for** each time slot $t \in T$ **do**
5: **for** each user $u \in U$ **do**
6: Each user u choose one server s via Algorithm 1.
7: Record $b_{u,s}^{(t)}$, $b_{-u}^{(t)}$, and selection for each user u.
8: **end for**
9: **while** not converged **do**
10: Calculate $r_u^{(t)}$, $s_u^{(t)}$, $R_s^{(t)}$, $REV_s^{(t)}$, and $P_s^{(t)}$ via (1), (2), (6), (7), and (8), respectively.
11: Input $b_{u,s}^{(t)}$, $b_{-u}^{(t)}$, $p^{(t)}$, and calculated $b^{(t)}$ via (9).
12: Input $b_u^{(t)}$, $p_s^{(t)}$, and calculated $p^{(t)}$ via (10).
13: Judge whether it converges to NE point via (27).
14: **end while**
15: Update policy parameter θ_u via Algorithm 1.
16: **end for**

We analyze the time complexity and convergence of our proposed PDCO algorithm in the following. During the execution of the PDCO algorithm, each end-user update policy based on actor network π_{θ_u} and critic network V_{ω_u}, so the computational complexity is only based on a fully connected deep neural network. According to [45], the time complexity can be described as $O\left(\sum_{f=1}^{F} \epsilon_f \cdot \epsilon_{f-1}\right)$, where ϵ_f denotes the number of neural units in the fully connected layer f. The convergence of our proposed PDCO algorithm is mainly determined by the convergence of the Nash equilibrium of the strategy game. As mention above, we provide detailed proof of the existence of the Nash equilibrium in APPENDIX.



(a) PDCO algorithm (b) Actor-Critic based approach
(c) Policy gradient based approach (d) Stochastic learning automata

FIGURE 3. The probabilities of users in different approaches.

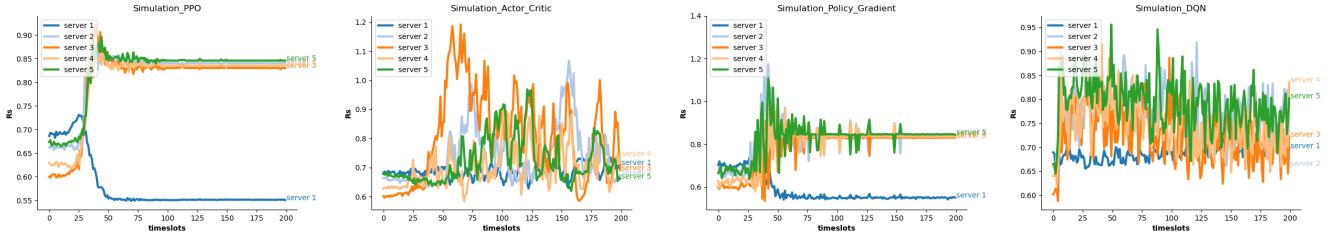


(a) PDCO algorithm (b) Actor-Critic based approach
(c) Policy gradient based approach (d) DQN based approach

FIGURE 4. The size of data processed by each server in different approaches.

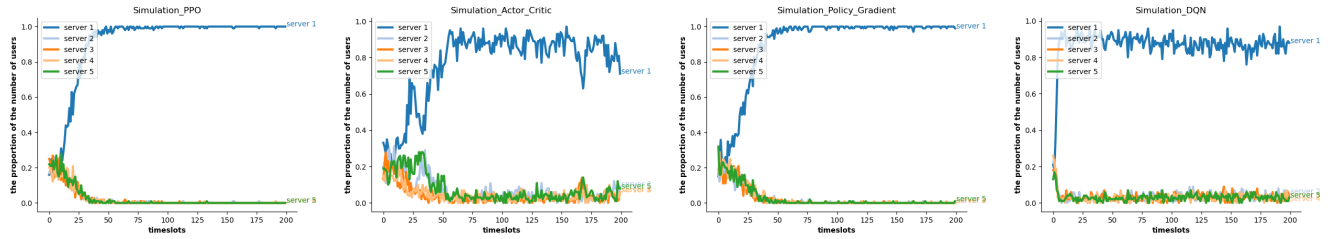
V. PERFORMANCE EVALUATION

We provide some numerical results to explain the superiority of the PDCO algorithm compared to other RL-based



(a) PDCO algorithm (b) Actor-Critic based algorithm (c) Policy gradient based algorithm (d) DQN based algorithm

FIGURE 5. **The reputation score of each server in different approaches.**



(a) PDCO algorithm (b) Actor-Critic based algorithm (c) Policy gradient based algorithm (d) DQN based algorithm

FIGURE 6. **The number of user per server in different approaches.**

approaches in this section. All algorithms and the corresponding simulation are implemented based on Python and executed on a desktop computer with Intel Core i7-8700 6 cores CPU and 32GB RAM.

A. SIMULATION SETTINGS

To prove the superiority of the PPO-based DRL framework, we conduct several detailed simulations in the MEC environment. We use different RL (or DRL) algorithms as baseline approaches to compare the decision-making process of MEC server selection. Specifically, we compare our proposed approach with the stochastic learning automata approach [46] used in [18], the policy gradient approach, the Deep Q-Network approach, and the Actor-Critic approach. Detailed data and charts are given in the following. Unless otherwise stated, the simulation parameters are summarized in Table 2.

However, it should be noted that this paper and the corresponding simulation results focus on the process of offloading decisions rather than the actual process of offloading and computing itself. To simplify the model, we assume that communication between end-users and MEC servers does not interfere with each other. Moreover, we do not deal with the transmission power control problem, assuming that the user transmits at a fixed power. Different from some existing studies [32], [47], [48], the impact of delay and energy consumption on offloading decisions is not considered.

B. SIMULATION RESULT

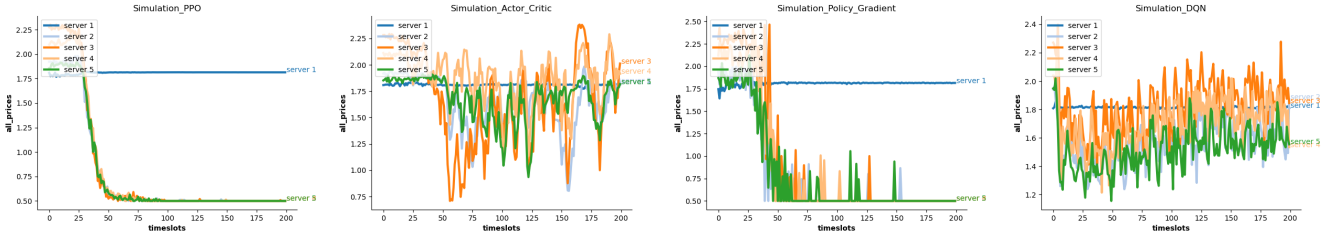
First of all, we study the convergence of the end-users selection probability for each server based on different DRL-based approaches. In this simulation, we set the number

TABLE 2. **Simulation parameters.**

Parameters	Values
Number of end-users	100
Number of MEC servers	5
The maximum size of offloading data	1000
The maximum episode	200
$w1$	1/3
$w2$	1/3
$w3$	1/3
The cost of server 1	0.12
The cost of server 2	0.14
The cost of server 3	0.20
The cost of server 4	0.17
The cost of server 5	0.13
The discount of server 1	0.05
The discount of server 2	0.04
The discount of server 3	0.02
The discount of server 4	0.03
The discount of server 5	0.05

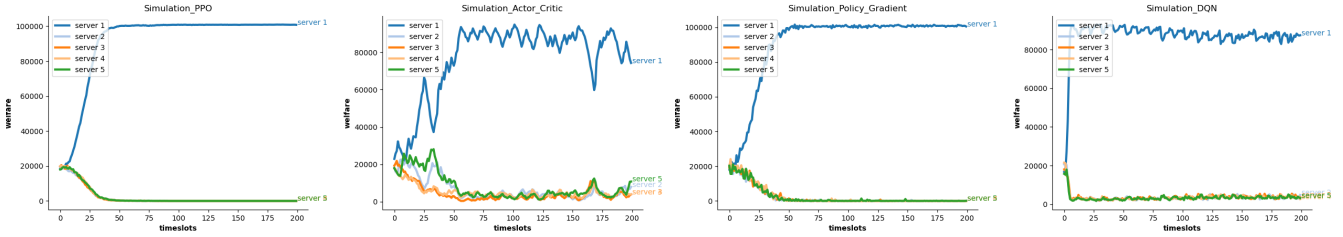
of end-users to 100 and the number of MEC servers to 5. We consider that the maximum size of offloading data that each end-user needs to execute is the same (i.e., $I_u^{(t)} = 1000\ bit$).

As shown in Figure 3, given a total number of episodes to 200, our proposed PDCO algorithm converges to Nash equilibrium in approximately 25 episodes, which is significantly better than other approaches in terms of convergence time and stability. The stochastic learning automata approach [46] proposed in [18] converges to Nash equilibrium in approximately 1100 episodes in the simulation with 2000 episodes. Since the DQN algorithm is a value-based DRL algorithm, the state transition probability is not involved. It should be noted that since various DRL-based approaches have more tremendous advantages than stochastic learning automata, the results of stochastic learning automata will no longer be shown in the following experimental results.



(a) PDCO algorithm (b) Actor-Critic based algorithm (c) Policy gradient based algorithm (d) DQN based algorithm

FIGURE 7. The service price of each server in different approaches.



(a) PDCO algorithm (b) Actor-Critic based algorithm (c) Policy gradient based algorithm (d) DQN based algorithm

FIGURE 8. The profit of each server in different approaches.

According to the actual situation, the more favorable the price of MEC computing service, the more attractive it is to end-users. Thereby server 1 accumulates most end-users. We notice that compared with other MEC servers, server 1 has the least cost and the greatest discount, set in Table 2. We count the total size of offloading data processed by each server in each time slot in the same simulation parameters. We find that our proposed PDCO algorithm also has good performance than other approaches. The DQN approach and the Actor-Critic approach do not converge well to the Nash equilibrium. The specific results are shown in Figure 4.

Subsequently, we observe the reputation score R_s of each server among different algorithms. As known from previous analysis, the MEC server's reputation R_s depends on the relative price, congestion level, and penetration rate. R_s essentially controls the probability that each end-user offloads its data. To this end, we calculate the reputation of the server and the number of users of the server at each time slot (represented by the percentage of the number of end-users accommodated by each MEC server to the total number of end-users). The specific results are shown in Figure 5 and Figure 6.

Then, we collect statistics on the pricing of computing services based on Game theory. We find that in different DRL-based approaches, the computing price of server 1 has almost stabilized, but the computing prices of other servers are fluctuating. The specific results are shown in Figure 7.

Finally, we compare the profit of MEC servers in different DRL-based approaches. As shown in Figure 8, we find that the trend of this parameter is similar to the total size of data processed by each server. As with the previous comparison results, we find that our proposed PDCO algorithm achieves Nash equilibrium faster than other DRL-based approaches.

VI. CONCLUSION

This paper studies and jointly optimizes the selection of MEC servers, the size of offloading data, and the price of the computing service. The flexibility and programmability provided by SDN technology make the actual implementation of the proposed framework possible. In particular, the proximal policy optimization DRL algorithm used in the MEC server select algorithm shows the power capacity and potential of DRL in some specific scenarios. The optimal size of offloading data and the optimal pricing of the computing service are formulated as a two-step optimization problem, which is solved by achieving the Nash equilibrium of the strategy game. By comparing different RL-based approaches in the specific MEC environment, a series of detailed simulation results demonstrate the performance and advantages of the PPO algorithm used in our proposal.

It should be noted that we assume that communication between end-users and servers does not interfere with each other. We do not deal with the transmission power control problem that we believe that the user transmits at a fixed power in our current work. In the next step, we will extend the existing research model and include communication interference and transmission power control as part of the joint optimization problem. In addition, as the MEC server is the public resource, end-users should consider the risk of the MEC server when selecting and using it. Our future work will focus on studying users' decision-making problems when facing threats of public resources, which is based on the tragedy of the commons [49] and prospect theory [50]. Finally, it should be pointed out that challenges arise due to ultra-low latency and energy efficiency requirements for some applications. In some existing work [51], the issues of latency and energy

have been directly considered part of optimization problems, and those factors are also taken into account in our future work.

APPENDIX

We provide a detailed proof of the existence of the Nash equilibrium in this section. According to [18] and [44], the necessary and sufficient conditions of the existence of the Nash equilibrium are as follows.

1) the strategy space $A_u^{(t)}, \forall u \in \mathcal{U}, t \in \mathcal{T}$ should be non-empty, convex, and compact subset of an Euclidean space $\mathbb{R}^U$.
2) the utility function $U_u^{(t)}\left(b_{u,s}^{(t)}, b_{-u}^{(t)}, p^{(t)}\right)$ is continuous in $b_u^{(t)}$ and quasi-concave in $b_{u,s}^{(t)}$.

We have the following analysis.

1) The strategy space $A_u^{(t)}$ denotes the size of offloading data that end-users can offload to MEC servers. Thus, by its range $0 \leq A_u^{(t)} \leq I_u^{(t)}$ we can know that the strategy space is non-empty, convex, and compact subset of an Euclidean space $\mathbb{R}^U$.
2) According to (4), the utility function $U_u^{(t)}\left(b_{u,s}^{(t)}, b_{-u}^{(t)}, p^{(t)}\right)$ is continuous in $b_u^{(t)}$.
3) We derive the second derivative of the utility function (4), which is given as follows.

$$\frac{\partial^2 U_u^{(t)}\left(b_{u,s}^{(t)}\right)}{\partial b_{u,s}^{(t)2}} = -\frac{\alpha_u \beta_u^2}{B_{-u}^{(t)2}} \cdot \frac{1}{\left[\beta_u + \frac{\beta_u b_{u,s}^{(t)}}{B_{-u}^{(t)}}\right]^2} < 0$$

According to $\frac{\partial^2 U_u^{(t)}\left(b_{u,s}^{(t)}\right)}{\partial b_{u,s}^{(t)2}} < 0$, we know that the utility function (4) is concave in $b_{u,s}^{(t)}$, which is also quasi-concave in $b_{u,s}^{(t)}$.

From the analysis above, we can verify the existence of Nash equilibrium of the strategy game $G = \left[U, \left\{A_u^{(t)}\right\}, \left\{U_u^{(t)}\right\}\right]$.

REFERENCES

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[2] M. García-Valls, A. Dubey, and V. Botti, "Introducing the new paradigm of social dispersed computing: Applications, technologies and challenges," *J. Syst. Archit.*, vol. 91, pp. 83–102, Nov. 2018.

[3] G. Fragkos, S. Lebien, and E. E. Tsiropoulou, "Artificial intelligent multi-access edge computing servers management," *IEEE Access*, vol. 8, pp. 171292–171304, 2020.

[4] H. Lin, S. Zeadally, Z. Chen, H. Labiod, and L. Wang, "A survey on computation offloading modeling for edge computing," *J. Netw. Comput. Appl.*, vol. 169, Nov. 2020, Art. no. 102781.

[5] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid method for minimizing service delay in edge cloud computing through VM migration and transmission power control," *IEEE Trans. Comput.*, vol. 66, no. 5, pp. 810–819, May 1, 2017.

[6] T. G. Rodrigues, K. Suto, H. Nishiyama, N. Kato, and K. Temma, "Cloudlets activation scheme for scalable mobile edge computing with transmission power control and virtual machine migration," *IEEE Trans. Comput.*, vol. 67, no. 9, pp. 1287–1300, Sep. 2018.

[7] L. Huang, S. Bi, and Y.-J.-A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 11, pp. 2581–2593, Nov. 2020.

[8] L. Huang, X. Feng, A. Feng, Y. Huang, and L. P. Qian, "Distributed deep learning-based offloading for mobile edge computing networks," *Mobile Netw. Appl.*, pp. 1–8, 2018.

[9] T. Yang, Y. Hu, M. C. Gursoy, A. Schmeink, and R. Mathar, "Deep reinforcement learning based resource allocation in low latency edge computing networks," in *Proc. 15th Int. Symp. Wireless Commun. Syst. (ISWCS)*, Aug. 2018, pp. 1–5.

[10] A. Shakarami, M. Ghobaei-Arani, and A. Shahidinejad, "A survey on the computation offloading approaches in mobile edge computing: A machine learning-based perspective," *Comput. Netw.*, vol. 182, Dec. 2020, Art. no. 107496.

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

[12] Z. Han, D. Niyato, W. Saad, T. Başar, and A. Hjørungnes, *Game Theory in Wireless and Communication Networks: Theory, Models, and Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2012.

[13] A. C. Baktir, A. Ozgovde, and C. Ersoy, "How can edge computing benefit from software-defined networking: A survey, use cases, and future directions," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2359–2391, 4th Quart., 2017.

[14] M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li, "Software defined networking: State of the art and research challenges," *Comput. Netw. J.*, vol. 72, pp. 74–98, Oct. 2014.

[15] Y. Jararweh, A. Doulat, A. Darabseh, M. Alsmirat, M. Al-Ayyoub, and E. Benkhelifa, "SDMEC: Software defined system for mobile edge computing," in *Proc. IEEE Int. Conf. Cloud Eng. Workshop (IC2EW)*, Apr. 2016, pp. 88–93.

[16] J. Wang and D. Li, "Adaptive computing optimization in software-defined network-based industrial Internet of Things with fog computing," *Sensors*, vol. 18, no. 8, p. 2509, Aug. 2018.

[17] J. Zheng, Y. Cai, Y. Wu, and X. Shen, "Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach," *IEEE Trans. Mobile Comput.*, vol. 18, no. 4, pp. 771–786, Apr. 2019.

[18] G. Mitsis, P. A. Apostolopoulos, E. E. Tsiropoulou, and S. Papavassiliou, "Intelligent dynamic data offloading in a competitive mobile edge computing market," *Future Internet*, vol. 11, no. 5, p. 118, May 2019.

[19] M. Patel *et al.*, "Mobile-edge computing introductory technical white paper," Mobile-Edge Comput. (MEC) Ind. Initiative, White Paper, 2014, pp. 854–864, vol. 29.

[20] L. Lin, X. Liao, H. Jin, and P. Li, "Computation offloading toward edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1584–1607, Aug. 2019.

[21] P. Popovski, K. F. Trillingsgaard, O. Simeone, and G. Durisi, "5G wireless network slicing for eMBB, URLLC, and mMTC: A communication-theoretic view," *IEEE Access*, vol. 6, pp. 55765–55779, 2018.

[22] A. Shakarami, M. Ghobaei-Arani, M. Masdari, and M. Hosseinzadeh, "A survey on the computation offloading approaches in mobile edge/cloud computing environment: A stochastic-based perspective," *J. Grid Comput.*, vol. 18, no. 4, pp. 639–671, 2020.

[23] A. Shakarami, A. Shahidinejad, and M. Ghobaei-Arani, "A review on the computation offloading approaches in mobile edge computing: A game-theoretic perspective," *Softw., Pract. Exper.*, vol. 50, no. 9, pp. 1719–1759, Sep. 2020.

[24] A. Rudenko, P. Reiher, G. J. Popek, and G. H. Kuenning, "Saving portable computer battery power through remote process execution," *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 2, no. 1, pp. 19–26, 1998.

[25] C. Xian, Y.-H. Lu, and Z. Li, "Adaptive computation offloading for energy conservation on battery-powered systems," in *Proc. Int. Conf. Parallel Distrib. Syst.*, Dec. 2007, pp. 1–8.

[26] I. Alghamdi, C. Anagnostopoulos, and D. P. Pezaros, "Delay-tolerant sequential decision making for task offloading in mobile edge computing environments," *Information*, vol. 10, no. 10, p. 312, Oct. 2019.

[27] K. Zhang, Y. Mao, S. Leng, S. Maharjan, and Y. Zhang, "Optimal delay constrained offloading for vehicular edge computing networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[28] Y. Wu, L. P. Qian, K. Ni, C. Zhang, and X. Shen, "Delay-minimization nonorthogonal multiple access enabled multi-user mobile edge computation offloading," *IEEE J. Sel. Topics Signal Process*, vol. 13, no. 3, pp. 392–407, Jun. 2019.

[29] C. You, Y. Zeng, R. Zhang, and K. Huang, "Asynchronous mobile-edge computation offloading: Energy-efficient resource management," *IEEE Trans. Wireless Commun.*, vol. 17, no. 11, pp. 7590–7605, Nov. 2018.
[30] Y. Pan, M. Chen, Z. Yang, N. Huang, and M. Shikh-Bahaei, "Energy-efficient NOMA-based mobile edge computing offloading," *IEEE Commun. Lett.*, vol. 23, no. 2, pp. 310–313, Feb. 2019.
[31] X. Xu, Y. Li, T. Huang, Y. Xue, K. Peng, L. Qi, and W. Dou, "An energy-aware computation offloading method for smart edge computing in wireless metropolitan area networks," *J. Netw. Comput. Appl.*, vol. 133, pp. 75–85, May 2019.
[32] G. Zhang, W. Zhang, Y. Cao, D. Li, and L. Wang, "Energy-delay tradeoff for dynamic offloading in mobile-edge computing system with energy harvesting devices," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4642–4655, Oct. 2018.
[33] W. Zhang, Z. Zhang, S. Zeadally, H.-C. Chao, and V. C. M. Leung, "MASM: A multiple-algorithm service model for energy-delay optimization in edge artificial intelligence," *IEEE Trans. Ind. Informat.*, vol. 15, no. 7, pp. 4216–4224, Jul. 2019.
[34] T. T. Vu, N. V. Huynh, D. T. Hoang, D. N. Nguyen, and E. Dutkiewicz, "Offloading energy efficiency with delay constraint for cooperative mobile edge computing networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–6.
[35] M. Liu and Y. Liu, "Price-based distributed offloading for mobile-edge computing with computation capacity constraints," *IEEE Wireless Commun. Lett.*, vol. 7, no. 3, pp. 420–423, Jun. 2018.
[36] S.-H. Kim, S. Park, M. Chen, and C.-H. Youn, "An optimal pricing scheme for the energy-efficient mobile edge computation offloading with OFDMA," *IEEE Commun. Lett.*, vol. 22, no. 9, pp. 1922–1925, Sep. 2018.
[37] H. Shah-Mansouri, V. W. S. Wong, and J. Huang, "An incentive framework for mobile data offloading market under price competition," *IEEE Trans. Mobile Comput.*, vol. 16, no. 11, pp. 2983–2999, Nov. 2017.
[38] S. Ranadheera, S. Maghsudi, and E. Hossain, "Mobile edge computation offloading using game theory and reinforcement learning," 2017, *arXiv:1711.09012*. [Online]. Available: http://arxiv.org/abs/1711.09012
[39] L. Xiao, C. Xie, T. Chen, H. Dai, and H. V. Poor, "A mobile offloading game against smart attacks," *IEEE Access*, vol. 4, pp. 2281–2291, 2016.
[40] Y. Zhan, S. Guo, P. Li, and J. Zhang, "A deep reinforcement learning based offloading game in edge computing," *IEEE Trans. Comput.*, vol. 69, no. 6, pp. 883–893, Jun. 2020.
[41] R. S. Sutton, A. G. Barto, *Introduction to Reinforcement Learning*, vol. 135. Cambridge, MA, USA: MIT Press, 1998.
[42] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.
[43] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*. [Online]. Available: http://arxiv.org/abs/1707.06347
[44] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*. Cambridge, MA, USA: MIT Press, 1994.
[45] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.
[46] A. G. Barto and P. Anandan, "Pattern-recognizing stochastic learning automata," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, no. 3, pp. 360–375, May 1985.
[47] T. Alfakih, M. M. Hassan, A. Gumaei, C. Savaglio, and G. Fortino, "Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on SARSA," *IEEE Access*, vol. 8, pp. 54074–54084, 2020.
[48] F. Guo, H. Zhang, H. Ji, X. Li, and V. C. M. Leung, "An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2651–2664, Dec. 2018.
[49] G. Hardin, "The tragedy of the commons," *J. Natural Resour. Policy Res.*, vol. 1, no. 3, pp. 243–253, 2009.
[50] J. S. Levy, "An introduction to prospect theory," *Political Psychol.*, pp. 171–186, 1992.
[51] P. A. Apostolopoulos, E. E. Tsiropoulou, and S. Papavassiliou, "Cognitive data offloading in mobile edge computing for Internet of Things," *IEEE Access*, vol. 8, pp. 55736–55749, 2020.

SHUYANG LI was born in Xuancheng, Anhui, China, in 1997. He received the B.E. degree from the Industrial and Commercial College, Anhui University of Technology, in 2019. He is currently pursuing the master's degree with Lanzhou Jiaotong University. His research interests include mobile edge computing and machine learning.

XIAOHUI HU was born in 1963. He received the Ph.D. degree from Northwestern Polytechnical University. He is currently working as a Professor and a Master Supervisor with Lanzhou Jiaotong University. His main research interests include distributed computing and mobile edge computing.

YONGWEN DU was born in 1974. He is currently working as an Associate Professor with Lanzhou Jiaotong University. His main research interests include wireless sensor networks and game theory.