COVER PAPER

Learning-Based Joint Service Caching and Load Balancing for MEC Blockchain Networks

Wenqian Zhang[1,2], Wenya Fan[1,2], Guanglin Zhang[1,2,*], Shiwen Mao[3]

[1] College of Information Science and Technology, Donghua University, Shanghai 201620, China
[2] Engineering Research Center of Digitized Textile and Apparel Technology, Ministry of Education, Shanghai 201620, China
[3] Department of Electrical and Computer Engineering, Auburn University, Auburn, AL 36849-5201, USA
[*] The corresponding author, email: glzhang@dhu.edu.cn, smao@ieee.org

Cite as: W. Zhang, W. Fan, *et al*., "Learning-based joint service caching and load balancing for mec blockchain networks," *China Communications*, vol. 20, no. 1, pp. 125-139, 2023. **DOI**: 10.23919/JCC.2023.01.011

Abstract: Integrating the blockchain technology into mobile-edge computing (MEC) networks with multiple cooperative MEC servers (MECS) providing a promising solution to improving resource utilization, and helping establish a secure reward mechanism that can facilitate load balancing among MECS. In addition, intelligent management of service caching and load balancing can improve the network utility in MEC blockchain networks with multiple types of workloads. In this paper, we investigate a learning-based joint service caching and load balancing policy for optimizing the communication and computation resources allocation, so as to improve the resource utilization of MEC blockchain networks. We formulate the problem as a challenging long-term network revenue maximization Markov decision process (MDP) problem. To address the highly dynamic and high dimension of system states, we design a joint service caching and load balancing algorithm based on the double-dueling Deep Q network (DQN) approach. The simulation results validate the feasibility and superior performance of our proposed algorithm over several baseline schemes.

Keywords: cooperative mobile-edge computing; blockchain; workload offloading; service caching; load balancing; deep reinforcement learning (DRL)

Received: May 26, 2022
Revised: Jul. 01, 2022
Editor: Haifeng Zheng

I. INTRODUCTION

Mobile edge computing (MEC), deployed in proximity to mobile devices (MD), is a promising technology to deal with latency-critical and computing-intensive workloads in the prospective Internet of Things (IoT) [1]. Establishing trust among multiple parties (e.g., edge/cloud providers) in MEC networks utilizing multiple servers (MECS) is a challenge because these parties often have conflicts of interest [2]. Blockchain, as an emerging decentralized security system [3, 4] and a public ledger of various types of transactions [5], has been incorporated in numerous applications, e.g., bitcoin, IoT, and smart grid, etc. [6]. Integrating the blockchain technology, with their advantages of decentralization, trust, and anonymity, into MEC systems has attracted great interest [7]. Compared with the traditional cooperative MEC system with a single central authority, the MEC system empowered by blockchain can enable decentralized, secure communications among cooperative MECS [8]. Because the MECS have the reputation records in the MEC Blockchain network, which motivates the MECS to process more workloads while meeting the requirements of MDs. This promotes load balancing among multiple MECS and full utilization of network computing resources.

In order to satisfy the service requests for delay-sensitive workloads and achieve high utilization of resources in MEC blockchain networks, edge caching [9, 10] and load balancing among cooperative MECS [11] were proposed. Edge caching can prestore the necessary application data at MECS for computing service, which can reduce the backhaul transmission delay to the core network and better utilize the service capability of MECS [12, 13]. In addition, MEC blockchain networks usually carry highly dynamic, diverse, and computation intensive workloads, which are difficult for a single MEC server to process [14]. Load balancing can reshape the workload distribution in MEC blockchain networks and facilitate the appropriate use of their limited computing resources [11]. In addition, the cooperative MEC networks empowered by blockchain can establish a secure reward mechanism to facilitate load balancing among MECS.

Most existing works are focused on secure workload offloading schedules [15, 16], credible data transmission schemes [17], the cooperation among MECS [18], and allocation of the limited communication and computation resources [19–21] in MEC blockchain networks, which attempted to improve the service capabilities or maximize the long-term system profits. Due to the complex process of solving these problems, it usually takes a long time for the iterative procedure to converge to the optimal solution [22]. In addition, the basis for the blockchain mechanism is a computing process called mining. Nevertheless, the mining process (e.g., performing Delegated Proof of Stake (DPoS)) [23] and workload computing in MEC systems are generally complicated and require considerable storage and computing resources [24]. Therefore, developing an intelligent and self-organizing resource allocation scheme is critical in MEC blockchain networks with limited service capabilities. To this end, deep reinforcement learning (DRL) was introduced to obtain optimal strategies and maximize long-term rewards [25, 26]. In [27], the DRL was introduced to optimize the energy allocation and minimize the system cost under highly dynamic and high-dimensional system states. The recent work in [28] performed task scheduling to maximize the long-term mining reward with the minimum cost on resources by leveraging DRL.

In this paper, we investigate the problem of joint service caching and load balancing for blockchain-authorized MEC networks with multiple cooperative MECS and multiple types of workloads. We aim to establish a secure load balancing mechanism to maximize the utilization of service resources in the MECS, and to jointly optimize service caching, workloads offloading, and service resources allocation strategies to achieve a high network revenue as well as meet the workload requirements. In particular, we present the main contributions of this work as follows. Firstly, we consider an MEC blockchain network with multiple cooperative MECS and MDs, as well as multiple types of workloads. We establish a secure load balancing mechanism based on blockchain to improve the service capability, and maximize the utilization of service resources of the network by optimizing the allocation of communication and computation resources. Secondly, we formulate the long-term network revenue maximization in MEC blockchain networks as an MDP problem. We then design a double-dueling DQN based joint service caching and load balancing algorithm to solve the formulated problem, which is characterized by the highly dynamic and high dimensional system states. Lastly, we analyze the convergence and performance of the proposed scheme through extensive simulations. Compared with several benchmark algorithms, the proposed algorithm achieves a greater network revenue while better satisfying the requirements of workloads.

The remainder of this work is organized as follows. In Sections II and III, we introduce the system model and problem formulation, respectively. In Section IV, we present the double-dueling DQN based joint service caching and load balancing algorithm. In Section V, we discuss the simulation results and performance analysis. We conclude the paper in Section VI.

II. SYSTEM MODEL

2.1 MEC Blockchain Networks

As depicted in Figure 1, we propose a blockchain-enabled mobile edge computing network with multiple cooperative MECS and MDs, which consists of an MEC system and a blockchain system. We consider that the MEC blockchain network has M MEC servers denoted by a set $\mathcal{M} \triangleq \{1, 2, ..., M\}$, and N MDs denoted by a set $\mathcal{N} \triangleq \{1, 2, ..., N\}$. The data traffic between MDs and MECS is transmitted through wire-

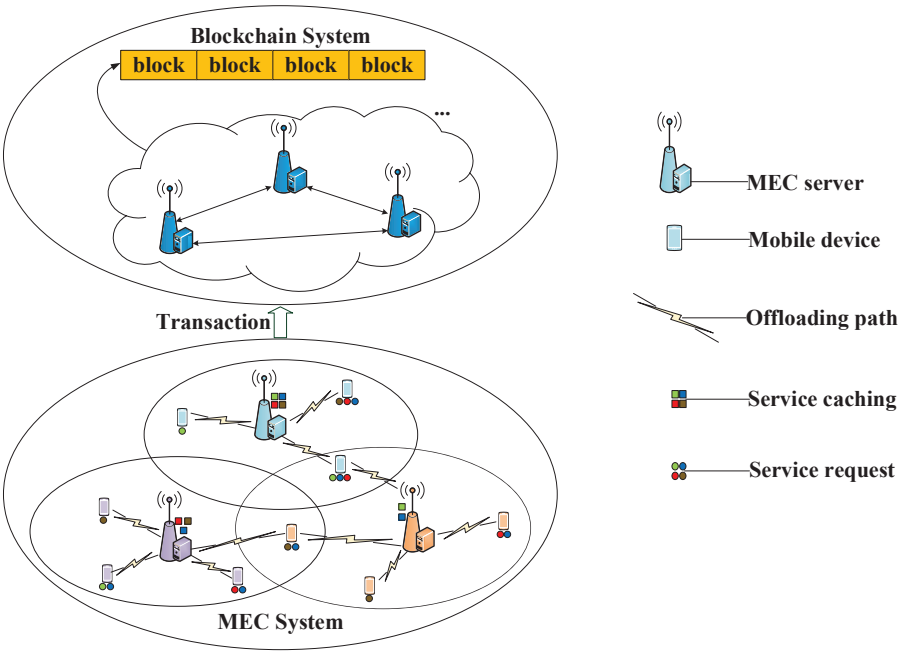

Figure 1. *Architecture of the MEC blockchain network considered in this paper.*

less channels, and its transmission mechanism is based on Orthogonal Frequency-Division Multiple Access (OFDMA) [29]. The cooperative MECS communicate over a wireline Local Area Network (LAN).

In the MEC system, we denote $\mathcal{N}_m \subseteq \mathcal{N}$ as the subset of MDs associated with MEC server m (e.g., MD n is the subscribe of the MEC server m, which is termed the "associated relationship" between MD n and MEC server m in this paper), and the MEC server m provides computation services for the MDs in $\mathcal{N}_m$ to obtain payoffs from the system. We assume that the MDs in the overlapping coverage area of multiple MECS can transmit workloads directly to the corresponding MEC server. After the computation results are returned, each MD will provide the corresponding MEC server a service evaluation score, which is related to the reputation value of the MEC server.

To ensure the security and privacy of the MEC system, we introduce the blockchain technology into the MEC network. The blockchain system can collect and store information from the MEC system, such as workload offloading records and the MEC server's reputation value. Such information will be grouped into data blocks and recorded on the blockchain after consensus is reached (e.g., the Nakamoto consensus agreement). The M MEC servers act as miners in the blockchain system, where the first miner to solve the consensus problem will obtain the mining reward and broadcast the verified transaction to other blockchain nodes in a safe and immutable manner [6].

2.2 Workload Arrival and System Service Capability

The proposed system operates over discrete time periods $\mathcal{T} \triangleq \{0, 1, ..., T\}$. In each time slot t, the workloads generated by each MD will be offloaded to one of the associated MEC server for execution. For the MD n, the types of generated workloads in time slot t can be modeled as a set $\mathcal{K} = \{1, 2, ..., K\}$. Without loss of generality, we assume that the workloads from MD n arrive at MEC m follow a Poisson distribution with rate $\pi_{n,m}(t)$ in time slot t [30]. We denote $\beta^k_{n,m}(t) \in [0, 1]$ as the proportion of type k workloads to the total workloads generated by MD n, and $\boldsymbol{\beta}_{n,m}(t) = \{\beta^k_{n,m}(t)\}_{k\in\mathcal{K}}$ is the set of workload percentages. The execution requirements for the type k workloads generated by MD n associated with MEC server m are modeled as a vector of four tuples, which is denoted by $I^k_{n,m}(t) \triangleq \{a_k, d_k, h_k, \tau_k\}$. For the type k workloads, a_k (in GB) indicates the required storage capacity, d_k (in Mb/workload) is the data size of each workload, h_k (in CPU cycles/Mb) denotes the required CPU cycles for workloads execution, and τ_k (in sec) is

the maximum execution delay deadline.

We consider the case that the MECS have limited service capabilities (e.g., computation capability, storage capacity, and communication capability), and the MEC with a heavier loads can transfer some workloads to the MECS that have lighter loads to achieve load balancing through the LAN. We denote R_m and F_m as the overall storage capacity and computation capability of MEC server m, respectively. Since the different execution requirements of each type of workloads, only the MECS that have cached the related applications data are eligible to provide services for the corresponding types of workloads.

2.3 Service Caching and Load Balancing

MD n sends service requests to the connected MECS at each time slot t. The service requests from MD n for type k workloads can be processed only when MEC m has cached the corresponding application data and has sufficient service resources. Let $\boldsymbol{x}_m(t) = \{x_m^k(t) \in \{0,1\} | m \in \mathcal{M}, k \in \mathcal{K}\}$ be the set of service caching decisions of MEC server m at time slot t, which is used to indicate whether the application data for type k workloads is cached at MEC server m (when $x_m^k(t) = 1$) or not (when $x_m^k(t) = 0$) at time slot t. Note that the service caching decisions are constrained by the overall storage capacity of MEC server m, i.e.,

$$\sum_{k \in \mathcal{K}} a_k x_m^k(t) \leq R_m. \tag{1}$$

When the service requests of MDs in $\mathcal{N}_m$ arrive at the associated MEC server m at each time slot, the load balancing among the cooperative MECS will be implemented by transmitting the redundant workloads to nearby MECS with low loads. Denote $\boldsymbol{z}_m(t) = \{z_{m,l}^k(t) | l \in \mathcal{M}, k \in \mathcal{K}\}$ as the set of load balancing decisions among MECS for MEC server m at time slot t, where $z_{m,l}^k(t) \in [0,1]$ is the proportion of the k-type of workloads transmitted from MEC server m to MEC server l.

Let $\mathcal{N}_{ml} \subseteq \mathcal{N}_m$ be the set of MDs associated with MEC server m in the overlapping area of MEC server m and MEC server l. Note that $\{\mathcal{N}_{ml}\}_{l \in \mathcal{M}, l \neq m} \bigcup \mathcal{N}_{mm} = \mathcal{N}_m$, where $\mathcal{N}_{mm}$ indicates the set of MDs associated with MEC server m only within the coverage area of MEC server m. We denote $\boldsymbol{y}_m(t) = \{y_{n,m}^k(t) \in \{0,1\} \mid n \in \mathcal{N}_m, k \in \mathcal{K}\}$ as the workload offloading decisions for MD n associated with MEC server m, where $y_{n,m}^k(t) = 1$ means that the workloads generated by MD n are offloaded to MEC server m at time slot t. Similarly, $y_{n,l}^k(t) = 1$ indicates that the workloads are transmitted to MEC server l from MD n associated with MEC server m directly. Note that if and only if $n \in \mathcal{N}_{ml}$, we have $y_{n,l}^k(t) \geq 0$, otherwise $y_{n,l}^k(t) = 0$. In addition, the workloads can only be processed on the MEC server m that caches the application data for type k workloads. Thus we have

$$\begin{cases} y_{n,l}^k(t) = 0, z_{m,l}^k(t) = 0, & \text{if } x_l^k(t) = 0 \\ y_{n,l}^k(t) \geq 0, z_{m,l}^k(t) \geq 0, & \text{if } x_l^k(t) \neq 0, \end{cases} \tag{2}$$

where $\{x_l^k(t) \in \{0,1\} | l \in \mathcal{M}, k \in \mathcal{K}\}$ is the service caching decision of MEC server l at time slot t.

2.4 System Cost

In the cooperative MEC system, we mainly consider the cost related to energy consumption and execution delay, which is determined by the following processes: (i) workload offloading to MECS; (ii) load balancing among MECS; and (iii) workload execution at MECS.

2.4.1 Workload Offloading to MECS

In view of the OFDMA transmission mechanism, interference between multiple MDs is ignored due to different MDs occupy non-overlapping subcarrier sets. We assume that there are $|S|$ subcarriers available for data wireless transmission among MEC server m and multiple MDs in its service area, which is denoted by $\mathcal{S} = \{1, 2, ..., s, ...|S|\}$ [29]. And $w_{n,m}(t)$ is the bandwidth of one of the subcarrier for the uplink data transmissions from MD n to MEC server m. The sum of occupied bandwidth resource of all MDs in the coverage area of MEC server m can not exceed the whole bandwidth resource of MEC server m, i.e.,

$$\sum_{n \in \mathcal{N}} w_{n,m}(t) \leq W_m, \tag{3}$$

where W_m is the overall available bandwidth resource of MEC server m.

In each time slot t, the workloads generated by the MDs associated with MEC server m can be offloaded

to MEC server m for execution, and then the processing results will be returned to MDs. Without loss of generality, we focus on the energy consumption of the uplink data transmission and execution delay. According to Shannon's theorem, the uplink data transmission rate between MD n and MEC server m is given by

$$r^u_{n,m}(t) = w_{n,m}(t) \log\left(1 + \frac{P^u_{n,m}(t) H_{n,m}(t)}{\sigma^2}\right), \tag{4}$$

where $P^u_{n,m}(t)$ is the transmit power for the uplink data transmissions from MD n to MEC server m; $H_{n,m}(t)$ is the channel gain; and σ^2 is the additive white Gaussian noise power. The uplink data transmission delay from MD n to MEC server m for unit workload of type k can be written as $T^k_{n,m}(t) = d_k / r^u_{n,m}(t)$, while $E^k_{n,m}(t) = P^u_{n,m}(t) d_k / r^u_{n,m}(t)$ denotes the corresponding energy consumption. Therefore, the overall cost for workloads offloading from the MDs associated with MEC server m in time slot t is

$$C^o_m(t) = \sum_{k \in \mathcal{K}} \left\{ B^k_m(t) \left[\varphi T^k_{n,m}(t) + (1-\varphi) E^k_{n,m}(t) \right] \right\}, \tag{5}$$

where φ is the relative weight between delay and energy consumption. The $B^k_m(t) = \sum_{n \in \mathcal{N}_{mm}} \beta^k_{n,m}(t) \pi_{n,m}(t) y^k_{n,m}(t) + \sum_{l \in \mathcal{M}, l \neq m} \beta^k_{n,m}(t) \pi_{n,m}(t) y^k_{n,l}(t)$ is the overall workloads of type k generated by the MDs in $\mathcal{N}_m$ that need to be offloaded at time slot t. In the first term ($\sum_{n \in \mathcal{N}_{mm}} \beta^k_{n,m}(t) \pi_{n,m}(t) y^k_{n,m}(t)$) of the expression $B^k_m(t)$, which means that the overall workloads of type k are offloaded by MDs (in $\mathcal{N}_{mm}$) to MEC server m at time slot t; The second term ($\sum_{l \in \mathcal{M}, l \neq m} \beta^k_{n,m}(t) \pi_{n,m}(t) y^k_{n,l}(t)$) of the expression $B^k_m(t)$, which is the overall workloads of type k are offloaded by MDs (in $\{\mathcal{N}_{ml}\}_{l \in \mathcal{M}, l \neq m}$) associated with MEC server m to the other MEC servers directly at time slot t.

2.4.2 Load Balancing Among MECS

Recall that the data transmission among MECS is through a wireline LAN with limited capacity, which incurs congestion delay. According to [30], the service capacity of the LAN is denoted as $1/\eta$, which follows the negative exponentially distribution. The data transmission among MECS for load balancing is modeled as an M/M/1 queuing system [31], which can be described as:

$$T_{m,g}(t) = B_m(t) \cdot \frac{1}{1/\eta - B(t)}, \quad B(t) < \frac{1}{\eta}, \tag{6}$$

where $B_m(t) = \sum_{l \in \mathcal{M}, l \neq m} B_{m,l}(t)$ is the total amount of workloads of MEC server m for load balancing in time slot t, and $B_{m,l}(t) = \sum_{k \in \mathcal{K}} B^k_{m,l}(t)$ is the total amount of workloads of all types transmitted from MEC server m to MEC server l. $B^k_{m,l}(t) = \sum_{n \in \mathcal{N}_{mm}} z^k_{m,l}(t) \left[\beta^k_{n,m}(t) \pi_{n,m}(t) y^k_{n,m}(t)\right]$ is the amount of type k workloads that will be transmitted from MEC server m to MEC server l. And $B(t) = \sum_{m \in \mathcal{M}} B_m(t)$ is the total data traffic rate in the LAN for load balancing among the MECS at time slot t. The energy consumption for load balancing among MECS in time slot t is given by

$$E_{m,g}(t) = P_{m,g}(t) T_{m,g}(t), \tag{7}$$

where $P_{m,g}(t)$ is the energy consumption per unit time for data transmission. Therefore, we obtain the overall system cost of MEC server m for load balancing among MECS at time slot t, which can be written as:

$$C^l_m(t) = \varphi T_{m,g}(t) + (1-\varphi) E_{m,g}(t). \tag{8}$$

2.4.3 Workload Execution at MECS

The total amount of workloads of the type k computed by MEC server m at time slot t is denoted as $\lambda^k_m(t)$, and it follows the Poisson process with rate $\lambda^k_m(t)$, which can be described as:

$$\begin{aligned} \lambda^k_m(t) = & \sum_{n \in \mathcal{N}_{mm}} \beta^k_{n,m}(t) \pi_{n,m}(t) y^k_{n,m}(t) \\ & + \sum_{l \in \mathcal{M}, l \neq m} \sum_{n \in N_{lm}} \beta^k_{n,m}(t) \pi_{n,m}(t) y^k_{n,m}(t) \\ & + \sum_{l \in \mathcal{M}, l \neq m} B^k_{l,m}(t) - \sum_{l \in \mathcal{M}, l \neq m} B^k_{m,l}(t), \end{aligned} \tag{9}$$

where the first term is for the workloads offloaded from MDs in $\mathcal{N}_{mm}$ to MEC server m at time slot t. The second term indicates the workloads transmitted by MDs in the overlapping areas between MEC server m and the other MECS, in which $\{N_{lm}\}_{l \in \mathcal{M}, l \neq m}$ is the set of MDs associated with MEC server l in the overlapping areas of MEC server l and MEC server m,

and $y_{n,m}^k(t)$ means that the workloads are transmitted to MEC server m from MD n associated with MEC server l directly. The third and fourth items are the workloads transmitted by other MECS to MEC server m and the workloads transmitted by MEC server m to the other MECS, respectively.

According to the M/M/1 queuing model [31] and Little's law [32], we obtain the average execution delay for type k workloads at MEC server m as follows:

$$T_{m,p}^k(t) = \frac{1}{f_{m,p}^k(t)/h_k - \lambda_m^k(t)}, \tag{10}$$

where $f_{m,p}^k(t)$ is the allocated computation capability of MEC server m for the k-type workloads at time slot t. $f_{m,p}^k(t)/h_k$ is the service capacity of MEC server m for workloads execution related to type k workloads, which follows a negative exponential distribution [30]. We obtain the average energy consumption computed by MEC server m for type k workloads in time slot t as:

$$E_{m,p}^k(t) = P_{m,p}^k(t) T_{m,p}^k(t), \tag{11}$$

where $P_{m,p}^k(t) = \kappa_m [f_{m,p}^k(t)]^3$ is the power consumption for processing the workloads at MEC server m in time slot t, and κ_m is the constant related to the structure of the CPU [33]. Thus, the overall system cost for workload execution at MEC server m in time slot t is given by

$$C_m^p(t) = \sum_{k \in \mathcal{K}} \left\{ \lambda_m^k(t) \left[\varphi T_{m,p}^k(t) + (1-\varphi) E_{m,p}^k(t) \right] \right\}. \tag{12}$$

Therefore, the total cost of MEC server m in the cooperative MEC system at time slot t can be written as

$$C_m(t) = C_m^o(t) + C_m^l(t) + C_m^p(t), \tag{13}$$

the total system cost is closely related to the MEC's service caching decisions, workload offloading decisions, and load balancing decisions.

2.5 System Reward

In the MEC blockchain network, the MECS can be rewarded in the following two ways: (i) providing workload processing services for MDs; (ii) being the first miner to solve the consensus problem. We next present the models for the workload execution payoffs and mining payoffs in detail.

2.5.1 Payoffs for Workload Execution

In order to incentivize load balancing among MECS, we introduce the payoffs for workloads execution. The payoff is related to not only the data size of the workloads, but also the reputation of each MEC server. Let $e_{n,m}^k(t)$ be the service evaluation results given by MD n for processing type k workloads at MEC server m in time slot t. Then $\boldsymbol{e}_m(t) = \{e_{n,m}^k(t) | n \in \mathcal{N}, k \in \mathcal{K}\}$ is the set of service evaluation results of MDs for MEC server m for processing all types of workloads.

Denote $c_{n,m}^k(t) \in \{0, 1\}$ as the credibility of MEC server m at time slot t, i.e., if $e_{n,m}^k(t)$ is sufficiently high relative to $\boldsymbol{e}_m(t)$, we have $c_{n,m}^k(t) = 1$; otherwise we have $c_{n,m}^k(t) = 0$ [34]. We obtain the credibility evaluation results of MEC server m by MD n at time slot t as $\hat{e}_{n,m}^k(t) = c_{n,m}^k(t) e_{n,m}^k(t)$, the reputation of MEC server m is given by Bayesian inference [35]:

$$Y_m^k(t) = \xi/n_1 \sum_{n \in N_{mm}} \hat{e}_{n,m}^k(t) + (1-\xi)/n_2 \sum_{l \in \mathcal{M}, l \neq m} \sum_{n \in N_{lm}} \hat{e}_{n,m}^k(t), \tag{14}$$

where the first term is the credibility evaluation results of MEC server m by MDs in $\mathcal{N}_{mm}$, the second term means that the credibility evaluation results of MEC server m by MDs in the overlapping areas between MEC server m and the other MECS, ξ is the weight coefficient, and n_1 and n_2 are the corresponding number of MDs.

According to the data size computed by MEC server m and the reputation of MEC server m at time slot t, we obtain the payoff of MEC server m for processing type k workloads at time slot t as

$$R_{m,p}^k(t) = \upsilon Y_m^k(t) \lambda_m^k(t), \tag{15}$$

where υ is the unit system payoff of MEC server m for executing type k workloads.

In summary, the MEC sever with higher reputations and processed more workloads will obtain more payoffs. Thus, the network is more inclined to load balancing among multiple MECS for maximizing the utilization of computation resources to process more

workloads, and each MEC server will also pay more attention to its own reputation (which is related to quality of service). Therefore, the MEC system empowered by blockchain help us establish a more decentralized and secure cooperative MECS network.

2.5.2 Mining Payoffs

In the proposed system, MEC server m also acts as a miner to process the mining service to obtain the mining payoffs in each time slot t. Let $f_{m,b}(t)$ be the allocated computation capability by MEC server m for mining service at time slot t. The ratio of $f_{m,b}(t)$ over the sum of the computing capability allocated to mining services by other MECS can be expressed as

$$\mu_m(t) = \frac{f_{m,b}(t)}{\sum_{m\in\mathcal{M}} f_{m,b}(t)}, \tag{16}$$

which is directly proportional to the success of mining competition and satisfies $\sum_{m\in\mathcal{M}} \mu_m(t) = 1$ [3]. In addition, in the propagation stage for mined block of MEC server m in the blockchain system, a slow propagation speed will lead to loss of the mined block and no reward (which is called orphaning [36]). The probability of orphaning is calculated as

$$\varrho(t) = 1 - e^{-\delta\zeta(s_m)}, \tag{17}$$

where δ is a constant, and $\zeta(s_m)$ indicates the propagation time for block size s_m of MEC server m [8]. Thus we obtain the probability of MEC server m successfully mining a block as

$$P_m(t) = \mu_m(t)(1 - \varrho(t)) = \mu_m(t)e^{-\delta\zeta(s_m)}. \tag{18}$$

Denote r_b as the mining reward for the winning MEC server. The expected mining payoffs of MEC server m can be expressed as

$$\begin{aligned} R_{m,b}(t) &= r_b P_m(t) \\ &= r_b \cdot \frac{f_{m,b}(t)}{\sum_{m\in\mathcal{M}} f_{m,b}(t)} \cdot e^{-\delta\zeta(s_m)}. \end{aligned} \tag{19}$$

III. PROBLEM FORMULATION

In order to achieve a higher network revenue, achieve load balancing, and encourage MECS to participate in cooperative workloads execution, the MEC blockchain network operators need to make optimal decisions for workload offloading, service caching, load balancing among MECS, and computation capability allocation in each time slot t. Let

$$\boldsymbol{\psi}(t) \triangleq \{\boldsymbol{x}_m(t), \boldsymbol{y}_m(t), \boldsymbol{z}_m(t), \boldsymbol{f}_m(t)\}$$

be the decisions set, where $\boldsymbol{f}_m(t) = \{f^k_{m,p}(t)\}_{k\in\mathcal{K}} \bigcup \{f_{m,b}(t)\}$. We aim to maximize the network revenue, i.e., to balance the cost and reward of the MEC system and the blockchain system. We denote the utility function of MEC server m at time slot t as

$$U_m(t) = (1-\rho)\Big\{[\sum_{k\in\mathcal{K}} R^k_{m,p}(t)] - C_m(t)\Big\} + \rho R_{m,b}(t), \tag{20}$$

where ρ (greater than zero) is the weight parameter for the utility between the MEC system and the blockchain system. We formulate a problem that maximizes the weighted and time-averaged sum of network revenue in the long-term time horizon as

$$\begin{aligned} \mathcal{P}_1: \quad & \max_{\boldsymbol{\psi}(t)} \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{m=1}^{M} U_m(t) \\ & s.t. \quad (1) - (3), (6) \\ & \sum_{l\in\mathcal{M}} y^k_{nl}(t) = 1, \sum_{l\in\mathcal{M}} z^k_{ml}(t) = 1 \qquad (21) \\ & \sum_{k\in\mathcal{K}} \beta^k_n(t) = 1, \sum_{m\in\mathcal{M}} \mu_m(t) = 1 \qquad (22) \\ & \sum_{k\in\mathcal{K}} f^k_{m,p}(t) + f_{m,b}(t) \le F_m. \qquad (23) \end{aligned}$$

In Problem $\mathcal{P}_1$, Constraint (1) represents the storage capacity constraint of MEC server m. Constraint (2) describes that the relationship between service caching, workload offloading, and load balancing decisions. Constraint (3) shows the limit of overall bandwidth resources of MEC server m. Constraint (6) enforces the limit of service capacity of the wireline LAN. Constraint (21) guarantees that the sum of the workload offloading decisions and the load balancing decisions of the MEC server m at time slot t are both equal to 1. The first term in Constraints (22) guarantees the percentage of each type of workloads generated by MD n at time slot t, and the second term of (22) indicates the limit of the proportional to the

success of mining competition. Constraint (23) ensures that the sum of allocated computation capabilities for workload computing and mining service cannot exceed the computation capability of MEC server m at time slot t.

The formulated problem of long-term network revenue maximization is a mixed integer nonlinear programming (MINLP) problem. As the number of MDs in the MEC blockchain networks is increased, the complexity of the problem will also increase greatly, which is difficult to solve by traditional methods. Therefore, we propose a highly competitive solution based on DRL to drive the strategy $\boldsymbol{\psi}(t)$.

IV. LEARNING-BASED JOINT SERVICE CACHING AND LOAD BALANCING POLICY

In this section, we consider Problem $\mathcal{P}_1$ as an MDP problem. We aim to design a learning-based joint service caching and load balancing policy to find a highly competitive solution to the original problem $\mathcal{P}_1$.

4.1 The DRL Framework

We first reformulate the problem as an MDP, and define the state, action, and reward function as follows.

4.1.1 State

The state of MEC server m at time slot t consists of the workload arrival rate $\boldsymbol{\pi}_m(t) = \{\pi_{1,m}(t), \pi_{2,m}(t), ..., \pi_{N,m}(t)\}$, the proportion of different types of workloads $\boldsymbol{\beta}_m(t) = \{\boldsymbol{\beta}_{1,m}(t), \boldsymbol{\beta}_{2,m}(t), ..., \boldsymbol{\beta}_{N,m}(t)\}$, the reputation $\boldsymbol{Y}_m(t) = \{Y_m^1(t), Y_m^2(t), ..., Y_m^K(t)\}$, the channel conditions $\boldsymbol{H}_m(t) = \{H_{1,m}(t), H_{2,m}(t), \ldots, H_{N,m}(t)\}$, the storage capacity R_m and the computation capability F_m, which can be written by a tuple as

$$s(t) \triangleq \left\{\boldsymbol{\pi}_m(t), \boldsymbol{\beta}_m(t), \boldsymbol{H}_m(t), \boldsymbol{Y}_m(t), R_m, F_m\right\}.$$

4.1.2 Action

In the MEC blockchain network, we consider four types of actions, including service caching $\boldsymbol{x}_m(t)$, workload offloading $\boldsymbol{y}_m(t)$, load balancing $\boldsymbol{z}_m(t)$, and computation capability allocation $\boldsymbol{f}_m(t)$. We denote

$$a(t) \triangleq \left\{\boldsymbol{x}_m(t), \boldsymbol{y}_m(t), \boldsymbol{z}_m(t), \boldsymbol{f}_m(t)\right\}$$

as the action space of MEC server m at time slot t. To simplify the problem, we divide the number of workloads and service resources of the MEC server m into countable parts to discretize the action space.

4.1.3 Reward Function

In this paper, we aim to maximize the network revenue by jointly optimizing the decisions for workload offloading, service caching, load balancing among MECS, and computation capability allocation. Therefore, the reward function needs to take these objectives into consideration, which is defined as

$$r(t) = \sum_{m=1}^{M} U_m(t). \tag{24}$$

4.2 Learning-Based Algorithm

Reinforcement learning (RL) is used to describe and solve the problem of reward maximization or achieving specific goals through learning strategies in the process of interacting with the environment, usually described as an MDP. It is an autonomous learning process, where the agent makes decisions periodically and gradually, relying on the feedback from the environment to improve the strategy, until the best strategy

$$\pi^* = \arg\max_{\pi} Q^*(s(t), a(t))$$

is learned. The agent aims to achieve the expected long-term reward, which can be expressed as:

$$r(t) = r(t) + \gamma r(t+1) + \gamma^2 r(t+2) + ... + \gamma^{T-t} r(T), \tag{25}$$

where $\gamma \in [0, 1]$ is the reward discount coefficient, indicating the influence of future rewards on the response of the current action.

DRL is an effective method to combine deep learning and RL to address problems with large action space and sample space, where a neural network called DQN is incorporated to approximate the Q value. In the DQN architecture, for given system state and

action inputs, the output Q value, $Q(s(t), a(t)) \approx Q'(s(t), a(t); \theta)$, can be obtained directly, where θ denotes the parameter of the neural network. The neural network is trained by iteratively updating the parameter θ to minimize the loss function:

$$L(\theta(t)) = \mathbb{E}\Big\{\big[r(t) + \gamma \max Q(s(t+1), a(t+1); \theta(t+1)) - Q(s(t), a(t); \theta(t))\big]^2\Big\}, \quad (26)$$

where $r(t) + \gamma \max Q(s(t+1), a(t+1); \theta(t+1))$ is the target Q value and will be updated every once in a while.

To overcome the Q value overestimation problem encountered in the DQN algorithm, we propose a double-dueling DQN based joint service caching and load balance algorithm. The key idea is to use different objective functions to select and evaluate actions, and then the target Q value in the double-dueling DQN can be expressed as:

$$r(t) + \gamma \max Q(s(t+1), a_{\max}(s(t)|\theta(t)); \theta(t+1)), \quad (27)$$

the $a_{\max}(s(t)|\theta(t)) = \arg\max_{a(t)} Q(s(t), a(t); \theta(t))$ is the best action, which is obtained through the current Q network.

The Q value is divided into two parts in the double-dueling DQN model. The first part is just based on the state and does not take into account the specific action to be performed, which is called value function and expressed as $V(s)$. The second part is called advantage function and denoted as $A(s, a)$, which is based on the current state and action. Thus, we obtain the Q value in the double-dueling DQN architecture as

$$Q(s, a) = A(s, a) + V(s). \quad (28)$$

In the implementation of the proposed double-dueling DQN based joint service caching and load balance algorithm, we set a fully connected feed-forward 5-layer neural network, and each hidden layer has 20 neurons [37]. In each training step, the state information in current system will be fed into the Q network. Then the Q network returns the optimal action, which is selected in accordance with the ϵ-greedy policy. Based on the optimal action (service caching, workload offloading, load balancing, and computation capability allocation decisions), we can obtain the network utility by solving (27). And then we obtain the value of reward function by solving (20) and obtain the next state $s(t+1)$. All the experience $(s(t), a(t), r(t), s(t+1))$ in the training process will be accumulated in the experience replay pool $\mathcal{D}$. A small group of samples will be selected from the pool to train the current network parameters, and the target network will be directly copied from the current network, with the same structure and parameters. The detailed algorithm is presented in Algorithm 1.

Algorithm 1. *The double-dueling DQN based joint service caching and load balancing algorithm.*

Input: Learning rate α; exploration rate ϵ; discount factor γ; experience replay memory $\mathcal{D}$; update step length C; action space $a(t)$.
Output: Optimal strategy $\psi(t)$.
Initialize the current network parameter θ;
Initialize the target network parameter $\theta' = \theta$;
Initialize the experience replay buffer;
for $t = 1, 2, ...$ **do**
  Observe the initial state $s(t)$;
  Select probability p randomly;
  if $p \geq \epsilon$ **then**
    Choose action $a_{\max}(s(t)|\theta(t)) = \arg\max_{a(t)} Q(s(t), a(t); \theta(t))$;
  else
    Choose an action randomly;
  end if
  Decrease the exploration probability ϵ;
  Execute action $a(t)$;
  Based on the service caching, workload offloading, load balancing, and computation capability allocation decisions to obtain the network utility by solving (20);
  Compute the reward function $r(t)$ by solving (24) and obtain the next state $s(t+1)$;
  Store the experience $(s(t), a(t), r(t), s(t+1))$ in the memory $\mathcal{D}$;
  Sample random mini-batch from $\mathcal{D}$;
  Calculate the target Q value by solving (27);
  Calculate the loss function and calculate network parameter θ;
  Every C steps reset $\theta' = \theta$;
end for

Table 1. *List of simulation parameters.*

Parameter	*Numerical value*	*Unit*
W_m	30	MHz
$\pi_{n,m}(t)$	[10,40]	workload/sec
κ	10^{-27}	-
σ^2	-174	dBm/Hz
f_{TX}	900	MHz
N_L	20	-
α	0.01	-
γ	0.9	-
s_m	[2,6]	MB
r_b	20	tokens
ϵ	0.1	-

V. PERFORMANCE EVALUATION

5.1 Simulation Configuration

In this section, we validate the performance of our proposed algorithm by simulations using the Pytorch with Python 3.7 (tensorflow) on a desktop with Windows 64 bits, 3.59 GHz AMD Ryzen 5 3600 6-Core Processor, and 16 GB RAM, and comparison with several baseline schemes. We consider an MEC blockchain networks including 30 MDs and 4 MECS. There are overlapping coverage areas between the MECS. Each MEC server serves a dedicated set of MDs that are associated with it. MDs can generate a total of four types of workloads. Assume that MECS have strong service capabilities to serve all types of workloads, and each MEC server can cache the corresponding application data in advance based on the caching policies. For each type of workload in the MEC system, the data size of each workload of type k is $d_k = [0.5, 1]$ Mb/-workload, the required CPU cycles for processing one type k workload is $h_k = [20, 40]$ CPU cycles/Mb, and the required storage capacity is $a_k = [20, 80]$ GB. The storage capacity and computation capacity of MEC servers are set to $[100, 200]$ GB and $[5, 10]$ GHz, respectively. The channel gain for wireless data transmission is modeled by the indoor loss model [14]:

$$L[dB] = 20\log(f_{TX})[\text{MHz}] + N_L\log(d[\text{m}]) - 28.$$

The noise power σ^2 is -174 dBm/Hz [38]. The weight factor φ between delay and energy consumption are set to 0.6 and 0.4, respectively. For the blockchain network, $\eta = 1/600$ sec [14] and the mining reward is set to $r_b = 20$ tokens. Other simulation parameters are listed in Table 1.

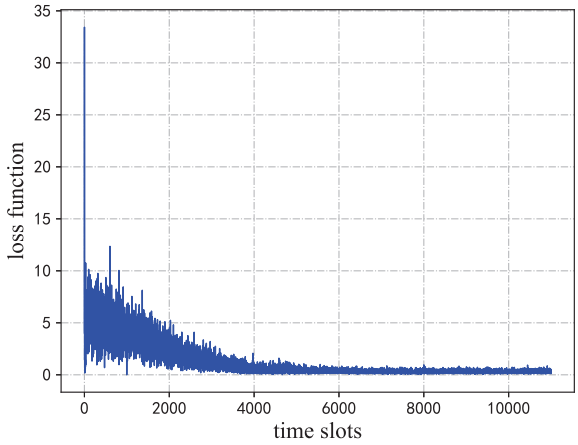

Figure 2. *Convergence performance of the proposed algorithm as indicated by the evolution of the loss function.*

We evaluate the performance of our proposed algorithm and compare it with the following baseline schemes under various system configurations:

1. No direct communications among MDs and their un-associated MECS (termed NDC): unlike our proposed scheme, in this scheme, the MDs in the overlapping coverage area of multiple MECS can only allow to offload workloads to its associated MEC server, and cannot directly offload workloads to other MECS that covering them.
2. Greedy offloading scheme (termed GO): in this scenario, each MEC server hopes to serve as many MDs as possible. As long as the MEC server caches the corresponding applications data to serve such type of workloads, the MEC server will reserve as many workloads as possible and ignore its computing capability and the current system state. For unserviceable workloads, the MEC server only considers the computing capability and ignores the reputation value when balancing the workloads to other MEC servers.
3. Random offloading scheme (termed RO): both MDs and MECS randomly select a feasible MEC server for workloads offloading with equal probability.

5.2 Results and Analysis

We first show the convergence of our proposed algorithm with respect to the loss function and learning rate in Figure 2 and Figure 3, respectively. In Figure 2, we present the convergence performance of the

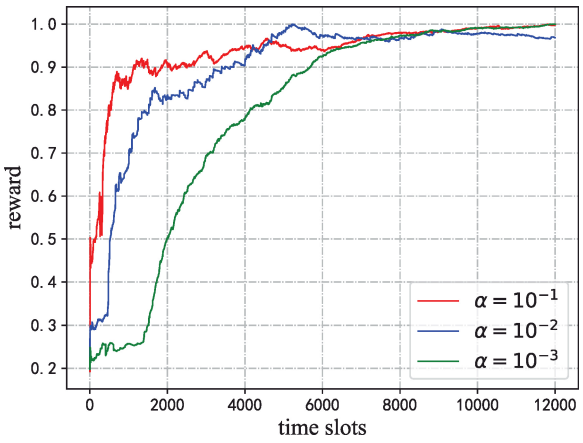

Figure 3. *Reward function value vs. different learning rates.*

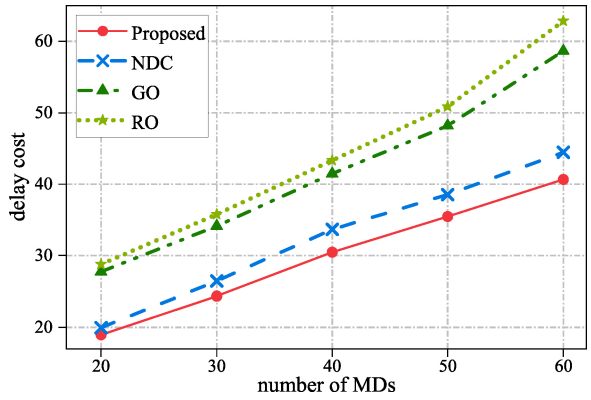

Figure 4. *The network delay cost under different numbers of mobile devices.*

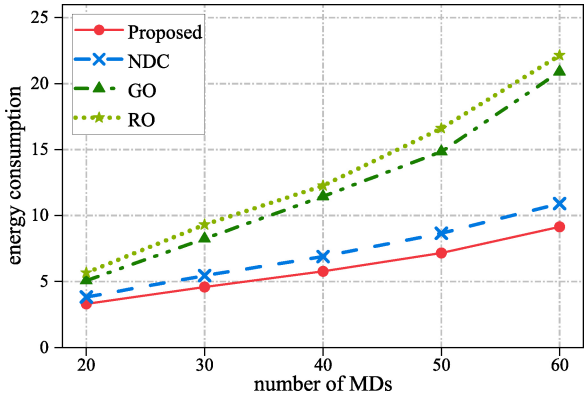

Figure 5. *The Energy consumption under different numbers of mobile devices.*

proposed algorithm as shown by the evolution of the loss function. At the beginning of the training process, since the double-dueling DQN agent does not have enough information to make reasonable decisions, the loss function assumes large values. As the training process goes on, the value of loss function decreases gradually and eventually approaches a relatively stable value after about 4,000 time slots. We then examine the influence of different learning rates on the convergence of the proposed algorithm in Figure. 3. We simulate the change of reward function values at $\alpha = 10^{-1}, 10^{-2}, 10^{-3}$ over 12,000 time slots. The vertical axis is the long-term averaged reward value, which is normalized by introducing $\hat{r} = \frac{r}{r_{\max}}$ for ease of viewing. It can be observed that the averaged reward value of the network gradually increases and approaches 1 as the learning process progresses. Furthermore, since the state of the network in each time slot may change dynamically, e.g., due to the dynamic workload arrival process, the curve will still fluctuate slightly even after convergence. Moreover, as the learning rate is increased from 10^{-3} to 10^{-1}, the convergence rate of the proposed algorithm also increases gradually.

Next, we examine the network delay cost under different numbers of MDs. The results are presented in Figure 4. As the increase of the number of MDs, the amount of workloads will also increase. Due to the limited computation capabilities of MECS, all the four curves show high network delay costs. Compared with the three baseline schemes, our proposed algorithm achieves the smallest network delay costs. In the NDC scheme, each MD can only offload workloads to its associated MEC server, and then the MEC server may transfer the workloads to other MEC servers that can execute the workloads. Such an approach increases the data transmission delay between MECS. Thus, the network delay cost is slightly higher than our proposed algorithm. In addition, the GO scheme offloads workloads to the MEC server with the largest computational capability other than itself. Thus its network delay cost is lower than that of the RO scheme.

We also demonstrate the energy consumption of the four schemes under different numbers of MDs in Figure 5. It can be seen that the energy consumption trends of the four algorithms are similar to the network delay cost trends shown in Figure 4.

Figure 6 verifies the effect of different numbers of MDs on the payment rewards received by MEC servers. According to (15), the payment rewards of each MEC server are related to the credibility evalua-

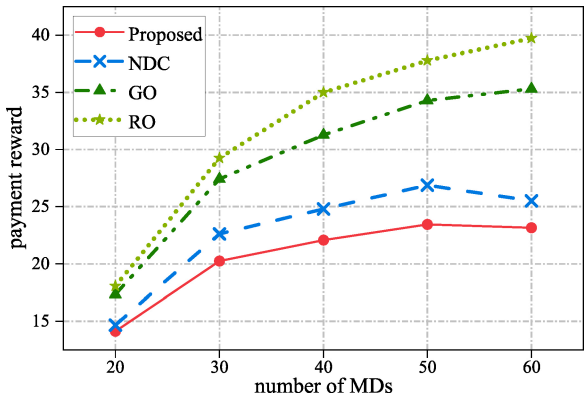

Figure 6. *The Payment reward under different numbers of mobile devices.*

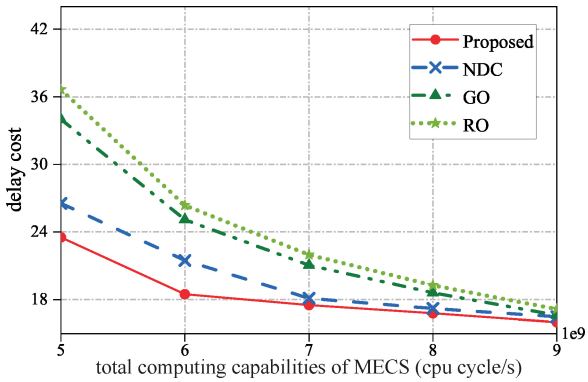

Figure 7. *Delay cost versus total computational capability.*

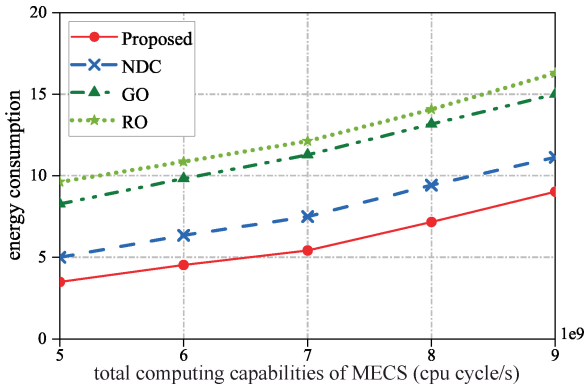

Figure 8. *Energy consumption vs. computing capability.*

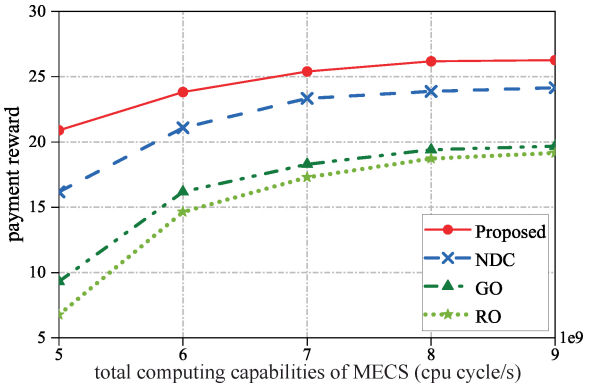

Figure 9. *Payment reward vs. computational capability.*

tion results of each MEC server provided by MDs and the amount of workload processing. As the number of MDs is increased, the amount of workloads increases, and the payment rewards of the four algorithms all become larger. However, due to the constant increase in the number of MDs on the premise of maintaining the same computing capabilities of MECS, the network delay cost gradually increases (as shown in Figure 4), which leads to poorer credibility evaluation results of MECS. Thus, the trends of payment rewards for the four algorithms will slow down or even decrease when the number of MDs becomes large.

Next, we examine the effect of different total computation capabilities of MECS on the network delay cost and the energy consumption of MECS in Figure 7 and Figure 8, respectively. Figure 7 shows the decreasing network delay cost as the total computation capabilities of MECS are increased. It can be seen that when the computing capabilities of MECS are sufficient to process the current workloads, the decrease of the network delay cost gradually slows down in all the curves of the four algorithms. Figure 8 shows that the energy consumption increases with the computing capabilities of MECS.

In Figure 9, we present the relationship between the payment reward obtained by the MECS and the total computing capabilities. According to (15), when the MEC server processes the same amount of workloads, a better value of credibility evaluation will provide the MEC server a higher payment reward. As the computing capabilities of MECS is increased, the processing delay gradually decreases, and the credibility evaluation results of MECS gradually become better. Thus, as the MECS computing capabilities are increased, the payment rewards obtained by the MECS get better and better. When the computing capabilities of MECS are sufficient to meet the current workloads, the payment rewards obtained by the MECS will be gradually stabilized.

Figure 10 shows the influence of the current block size on the value of the objective function. According to (20), the value of the objective function is re-

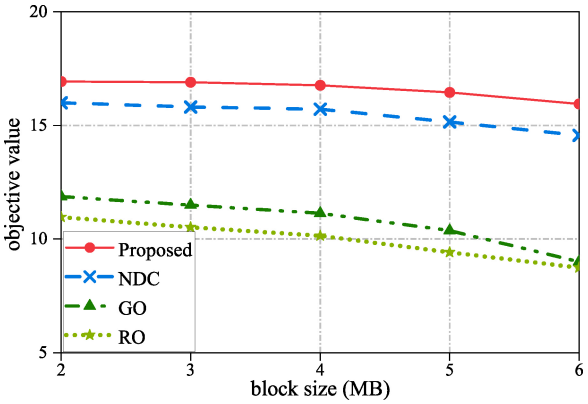

Figure 10. *Objective function value vs. block sizes.*

lated to the mining payoffs of MECS. A larger block size value leads to a longer propagation time for the corresponding block, which is more likely to be lost during the propagation process. Therefore, the probability of success of block mining will becomes smaller. And thus the objective function values of the four algorithms decrease gradually with the increase of block size.

VI. CONCLUSION

In this paper, we considered the problem of joint service caching and load balancing for blockchain-authorized MEC networks with multiple cooperative MECS and multiple types of workloads. We established a secure load balancing mechanism among cooperative MECS based on the blockchain technology to maximize resources utilization. We formulated a long-term network revenue maximization MDP problem and developed a double-dueling DQN algorithm for network revenue maximization while satisfying the requirements of MDs. We analyzed the convergence and feasibility of the proposed algorithm by extensive simulations. Compared with three baseline schemes, our proposed algorithm achieved a superior performance in terms of the energy consumption, the network delay cost, and the payment reward in MEC blcokchain networks.

ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China 62072096, the Fundamental Research Funds for the Central Universities under Grant 2232020A-12, the International S&T Cooperation Program of Shanghai Science and Technology Commission under Grant 20220713000, the Young Top-notch Talent Program in Shanghai, the "Shuguang Program" of Shanghai Education Development Foundation and Shanghai Municipal Education Commission, the Fundamental Research Funds for the Central Universities and Graduate Student Innovation Fund of Donghua University CUSF-DH-D-2019093. S. Mao's research is supported in part by the NSF under grants CNS-2107190 and ECCS-1923717.

References

[1] Y. Xu, T. Zhang, *et al.*, "UAV-assisted MEC networks with aerial and ground cooperation," *IEEE Transactions on Wireless Communications*, vol. 20, no. 12, pp. 7712–7727, Dec. 2021.

[2] X. Qiu, L. Liu, *et al.*, "Online deep reinforcement learning for computation offloading in blockchain-empowered mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 8050–8062, Aug. 2019.

[3] Y. Jiao, P. Wang, *et al.*, "Social welfare maximization auction in edge computing resource allocation for mobile blockchain," in *Proc. IEEE ICC 2018*, pp. 1–6, Kansas City, MO, May 2018.

[4] H. Li, P. Gao, *et al.*, "Blockchain technology empowers telecom network operation," *China Communications*, vol. 19, no. 1, pp. 274–283, 2022.

[5] J. Kang, R. Yu, *et al.*, "Blockchain for secure and efficient data sharing in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4660–4670, June 2019.

[6] D. C. Nguyen, P. N. Pathirana, *et al.*, "Privacy-preserved task offloading in mobile blockchain with deep reinforcement learning," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2536–2549, Dec. 2020.

[7] Z. Hong, H. Huang, *et al.*, "QoS-aware cooperative computation offloading for robot swarms in cloud robotics," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 4027–4041, Apr. 2019.

[8] D. Nguyen, M. Ding, *et al.*, "Cooperative task offloading and block mining in blockchain-based edge computing with multi-agent deep reinforcement learning," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2021.

[9] Y. Li, Z. Chen, *et al.*, "Coded caching with device computing in mobile edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 20, no. 12, pp. 7932–7946, Dec. 2021.

[10] Z. Chen, Z. Chen, *et al.*, "Joint optimization of task caching, computation offloading and resource allocation for mobile edge computing," *China Communications*, 2022.

[11] H. Wu, L. Chen, *et al.*, "Online geographical load balancing for energy-harvesting mobile edge computing," in *Proc. IEEE ICC 2018*, pp. 1–6, Kansas City, MO, May 2018.

[12] Y. Chen, M. Wen, *et al.*, "Exploiting reconfigurable intelligent surfaces in edge caching: Joint hybrid beamforming and content placement optimization," *IEEE Transactions on*

Wireless Communications, vol. 20, no. 12, pp. 7799–7812, Dec. 2021.
[13] K. Zhang, S. Leng, *et al.*, "Cooperative content caching in 5g networks with mobile edge computing," *IEEE Wireless Communications*, vol. 25, no. 3, pp. 80–87, June 2018.
[14] L. Chen, S. Zhou, *et al.*, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1619–1632, Aug. 2018.
[15] A. Vera-Rivera, A. Refaey, *et al.*, "Blockchain-based collaborative task offloading in MEC: A hyperledger fabric framework," in *Proc. IEEE ICC 2021 Workshops*, pp. 1–6, Montreal, Canada, June 2021.
[16] H. Wu, K. Wolter, *et al.*, "EEDTO: An energy-efficient dynamic task offloading algorithm for blockchain-enabled iot-edge-cloud orchestrated computing," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2163–2176, Feb. 2021.
[17] X. Ye, M. Li, *et al.*, "Blockchain and MEC-assisted reliable billing data transmission over electric vehicular network: An actor-critic RL approach," *China Communications*, vol. 18, no. 8, pp. 279–296, Aug. 2021.
[18] H. Yang, Y. Liang, *et al.*, "Distributed blockchain-based trusted multidomain collaboration for mobile edge computing in 5G and beyond," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 11, pp. 7094–7104, Nov. 2020.
[19] Y. Xu, H. Zhang, *et al.*, "Transaction throughput optimization for integrated blockchain and mec system in IoT," *IEEE Transactions on Wireless Communications*, pp. 1–1, 2021.
[20] J. Feng, F. Richard Yu, *et al.*, "Cooperative computation offloading and resource allocation for blockchain-enabled mobile-edge computing: A deep reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6214–6228, July 2020.
[21] A. Asheralieva and D. Niyato, "Learning-based mobile edge computing resource management to support public blockchain networks," *IEEE Transactions on Mobile Computing*, vol. 20, no. 3, pp. 1092–1109, Mar. 2021.
[22] Y. Wang, K. Wang, *et al.*, "Traffic and computation co-offloading with reinforcement learning in fog computing for industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 976–986, Feb. 2019.
[23] J. Kang, Z. Xiong, *et al.*, "Toward secure blockchain-enabled internet of vehicles: Optimizing consensus management using reputation and contract theory," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2906–2920, Mar. 2019.
[24] H. Guo, J. Liu, *et al.*, "Mobile-edge computation offloading for ultradense IoT networks," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4977–4988, Dec. 2018.
[25] V. Mnih, K. Kavukcuoglu, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.
[26] J. Jiao, X. Sun, *et al.*, "An overview of wireless communication technology using deep learning," *China Communications*, vol. 18, no. 12, pp. 1–36, 2021.
[27] L. Yang, M. Li, *et al.*, "Energy-efficient resource allocation for blockchain-enabled industrial internet of things with deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2318–2329, Feb. 2021.
[28] Y. Gao, W. Wu, *et al.*, "Deep reinforcement learning based task scheduling in mobile blockchain for IoT applications," in *Proc. IEEE ICC 2020*, pp. 1–7, Dublin, Ireland, June 2020.
[29] L. Tan, Z. Kuang, *et al.*, "Energy-efficient joint task offloading and resource allocation in ofdma-based collaborative edge computing," *IEEE Transactions on Wireless Communications*, vol. 21, no. 3, pp. 1960–1972, 2022.
[30] L. Chen, J. Xu, *et al.*, "Computation peer offloading in mobile edge computing with energy budgets," in *Proc. IEEE GLOBECOM 2017*, pp. 1–6, Singapore, Dec. 2017.
[31] R. Cooper, *Introduction to Queueing Theory*. Amsterdam, The Netherlands: Academic Press, 1981.
[32] S. Ross, "Introduction to probability models," *Amsterdam, The Netherlands: Academic Press*, 2014.
[33] Y. Mao, J. Zhang, *et al.*, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 16, no. 9, pp. 5994–6009, Sept. 2017.
[34] Z. Yang, K. Yang, *et al.*, "Blockchain-based decentralized trust management in vehicular networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1495–1505, Apr. 2019.
[35] M. Raya, P. Papadimitratos, *et al.*, "On datacentric trust establishment in ephemeral ad hoc networks," in *Proc. IEEE INFOCOM 2008*, pp. 1238–1246, Phoenix, AZ, Apr. 2008.
[36] D. C. Nguyen, P. N. Pathirana, *et al.*, "Blockchain for 5G and beyond networks: A state of the art survey," in *Journal of Network and Computer Applications*, vol. 166, p. 102693, Sept. 2020.
[37] D. Shi, J. Ding, *et al.*, "Deep Q-network-based route scheduling for tnc vehicles with passengers' location differential privacy," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7681–7692, 2019.
[38] J. Du, L. Zhao, *et al.*, "Enabling low-latency applications in lte-a based mixed fog/cloud computing systems," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1757–1771, Feb. 2018.

Biographies

Wenqian Zhang received the Ph.D. degree in information and communication intelligence systems from Donghua University, Shanghai, China in 2022. From 2019 to 2020, she was a visiting student with the Department of Electrical and Computer Engineering, Auburn University, Auburn, AL. She is currently an assistant professor with the Department of Communication Engineering, Shanghai Maritime University, Shanghai. Her research interests include edge intelligence, mobile edge computing, and blockchain.

Wenya Fan received the B.S. degree in Communication engineering from Zhejiang SciTech University in 2019. She is currently a Master student with Department of Communication Engineering, College of Information Science and Technology, Donghua University. Her research interests include task offloading, service caching and computation resources allocation in mobile edge computing, etc.

Guanglin Zhang received the Ph.D. degree in information and communication engineering from Shanghai Jiao Tong University in 2012. From 2013 to 2014, he was a Postdoctoral Research Associate with the Institute of Network Coding, Chinese University of Hong Kong, China. He is currently a professor and the department chair with the Department of Communication Engineering, and he is the vice dean with the College of Information Science and Technology, Donghua University. His research interests include capacity scaling of wireless networks, vehicular networks, smart micro-grid, and mobile edge computing. He serves as a Technical Program Committee Member for IEEE Globecom 2016-2017, IEEE ICC 2014, 2015, 2017, IEEE VTC2017-Fall, IEEE/CIC ICCC 2014, and WCSP 2014, APCC 2013, and WASA 2012. He serves as the Local Arrangement Chair of ACM TURC 2017, and Vice TPC Co-Chair of ACM TURC 2018. He serves as Editors on the Editorial Board of China Communications and Journal of Communications and Information Networks.

Shiwen Mao [S'99-M'04-SM'09-F'19] received his Ph.D. in electrical engineering from Polytechnic University, Brooklyn, NY in 2004. He joined Auburn University, Auburn, AL in 2006, and held the McWane Professorship from 2012 to 2015 and the Samuel Ginn Professorship from 2015 to 2020. Currently, he is a professor and Earle C. Williams Eminent Scholar Chair, and Director of the Wireless Engineering Research and Education Center at Auburn University. His research interest includes wireless networks, multimedia communications, and smart grid. He is an Associate Editor-in-Chief of IEEE/CIC China Communications, an Area Editor of IEEE Transactions on Wireless Communications, IEEE Internet of Things Journal, IEEE Open Journal of the Communications Society, and ACM GetMobile, and an Associate Editor of IEEE Transactions on Cognitive Communications and Networking, IEEE Transactions on Network Science and Engineering, IEEE Transactions on Mobile Computing, IEEE Multimedia, and IEEE Networking Letters, among others. He is a Distinguished Lecturer of IEEE Communications Society and the IEEE Council of RFID. He received the IEEE ComSoc TC-CSR Distinguished Technical Achievement Award in 2019 and NSF CAREER Award in 2010. He is a co-recipient of the 2021 Best Paper Award of Elsevier/KeAi Digital Communications and Networks Journal, the 2021 IEEE Internet of Things Journal Best Paper Award, the 2021 IEEE Communications Society Outstanding Paper Award, the IEEE Vehicular Technology Society 2020 Jack Neubauer Memorial Award, the 2018 Best Journal Paper Award and the 2017 Best Conference Paper Award from IEEE ComSoc MMTC, and the 2004 IEEE Communications Society Leonard G. Abraham Prize in the Field of Communications Systems. He is a co-recipient of the Best Paper Awards from IEEE ICC 2022, IEEE GLOBECOM 2019, 2016, and 2015, IEEE WCNC 2015, and IEEE ICC 2013, and the Best Demo Awards from IEEE INFOCOM 2022 and IEEE SECON 2017.