Mean-Field Learning for Edge Computing in Mobile Blockchain Networks

Xiaojie Wang, Zhaolong Ning, Lei Guo, Song Guo, *Fellow, IEEE*, Xinbo Gao, and Guoyin Wang

Abstract—Blockchain has been leveraged to secure transactions for the m-commerce. However, the intensive computation in the mining process restricts the participation of mobile devices. Currently, some studies have deployed edge computing services to support the mining process, where edge servers managed by one Service Provider (SP) are considered. This paper investigates a more practical scenario with multiple SPs, where servers managed by different SPs have distinct capacities and prices, making miners' offloading decisions rather complicated. To tackle the above challenges, we consider task offloading, block propagation and miner mobility comprehensively to maximize utilities of miners. Specifically, we first formulate a Markov game, and then design a learning-based offloading algorithm for off-chain computation, where a novel learning model is constructed by integrating Deep Reinforcement Learning (DRL) and Mean Field Theory (MFT) to guarantee a Nash equilibrium. Different from existing studies, each miner merely needs to respond to the average effect from others in our system, insteading of knowing policies of others. Finally, both theoritical and performance results show that our designed algorithm has superiority on average miner utilities and algorithm convergence time compared with other representative algorithms.

Index Terms—Mobile blockchain, deep reinforcement learning, mean field theory, multiple service providers.

1 Introduction

Blockchain has been widely adopted in kinds of applications, including bitcoin [1], the Internet of things [2] and smart grids [3], by taking its advantage of security and distributed characteristics. Generally, blockchain is regarded as a distributed ledger, which can store transactions and allow them to be grouped into blocks, and even replicated as well as distributed across the entire network. To add transactions into a blockchain, a mining process needs to be conducted. Miners solve a computation-intensive problem, e.g., Proof-of-Work (PoW) puzzle, and broadcast their blocks to others for verification as soon as possible. If a consensus is reached by miners, a new block can be added to the blockchain. There are many consensus protocols [4], among which Nakamoto consensus [5] is widely utilized for blockchain applications. It rewards the miner that first solves the PoW puzzle. Typically, the blockchain can be divided into two categories, i.e., public and private ones. The former allows everyone to read the blockchain, carry out transactions and participate into the mining process. The latter merely allows a predefined number of participants to conduct consensus processes. Currently, most of existing studies focus on public blockchain.

Recently, blockchain is faced with a booming development. According to the report of Tractica (a market research firm), the annual revenue for blockchain-based applications in enterprises will be up to $19.9 billion by 2025 [6]. Meanwhile, with the development of networking technologies and ubiquitous services, the number of mobile applications has significantly increased. Followed that, more e-commerce services can be conducted on mobile devices, providing opportunities for mobile blockchain. It can provide the robustness against network failure and malicious attacks, as well as data trustworthiness for mobile applications, such as e-health and credit member systems. Although there are several consensus mechanism, such as Proof-of-State (PoS) and Delegated Proof-of-Stake (DPoS), PoW is still in a dominant place for the blockchain application due to its simplicity and efficiency for consensus reaching in a distributed manner. Thus, it is significant to support the PoW mining process in the mobile environment. However, the limited computing and energy resources of mobile devices restrict the mining process of PoW consensus mechanism. Fortunately, improvements of edge computing technologies and hashing algorithms make mobile blockchain become possible, where computation-intensive tasks can be offloaded to edge servers for off-chain processing [7].

Existing studies on task offloading for mobile blockchain are mainly based on one Service Provider (SP), i.e., one SP manages all edge servers, and miners merely need to decide whether to offload computation tasks or not. However, there are many SPs in reality, such as Cisco, IBM, and Cloudera [8]. Thus, miners not only need to decide whether to offload tasks, but also should choose suitable servers belonging to different SPs for task processing, which is more complicated and practical. Several studies focus on the multi-server multi-client edge computing environment. For example, authors in [9] propose a heterogeneous deep learning-based offloading algorithm to guarantee the qual-

- *X. Wang, Z. Ning and L. Guo are with the Institute of Intelligent Communication and Network Security, School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China. E-mail: xiaojie.kara.wang@ieee.org, z.ning@ieee.org, guolei@cqupt.edu.cn.*
- *S. Guo is with the Department of Computing, The Hong Kong Polytechnic University, Kowloon, Hong Kong, China. Email: song.guo@polyu.edu.hk.*
- *X. Gao is with the Chongqing Key Laboratory of Image Cognition, Chongqing University of Posts and Telecommunications, Chongqing 400065, China. Email: gaoxb@cqupt.edu.cn.*
- *G. Wang is with the Chongqing Key Laboratory of Computational Intelligence, Chongqing University of Posts and Telecommunications, Chongqing 400065, China. Email: wanggy@cqupt.edu.cn.*

ity of service in Mobile Edge Computing (MEC) networks. However, the competition among clients together with the coordination between offloading decisions and block size selection is ignored. Similarly, a data offloading algorithm in the multi-server multi-access edge computing environment is designed in [10], in which multi-servers are regarded as a resource pool, while user offloading decisions driven by different resource prices are not taken into consideration. When making decisions, users need to know policies of others beforehand. Similar with our work, an edge computing supported blockchain network is investigated in [11], where multiple service providers are considered. However, this study also assumes that miners know others' real-time service demands, which is impossible in real-world scenarios. In addition, the dynamic nature of miners' action is not considered, which is important in mobile networks since wireless devices can not access to all edge servers simultaneously.

Specifically, the challenges of task offloading for mobile blockchain with multiple SPs can be summarized as follows:

- On one hand, long-term expected utilities of miners need to be maximized, leading to the inefficiency of traditional optimization technologies. On the other hand, miners can move around, resulting in dynamic action sets. However, typical Deep Reinforcement Learning (DRL) algorithms require a fixed set of candidate actions as the input of policy networks, which is not suitable for the system with variable action sets. Consequently, novel learning algorithms are called for investigation.
- In the wireless network, miners can not observe the full system state, due to their limited communication coverage. Local observations cause miners be unaware of others' policies, and thereby they can not make suitable offloading decisions as well as block size selection. Thus, efficient scheduling policies for miners should be designed, not only to maximize their own utilities, but also to reach a Nash equilibrium among miners.
- With multiple SPs, servers are with different processing abilities and various resource prices, which complicate miners' decisions. They need to decide task processing modes, server selection as well as block sizes, and these factors have mutual impacts on each other. For example, whether to offload tasks affects server selection, and the block size has a direct relationship with the miner reward. Inversely, the obtained reward affects miners' utilities, which also have impacts on miners' budget in the next stage. Therefore, all factors should be comprehensively considered to maximize miners' utilities.

To solve the above challenges, we propose a learning-based computation offloading approach for mobile blockchain in multi-provider edge computing networks, named LOBE, to maximize the long-term expected utilities of miners. To the best of our knowledge, *this work is the first to support off-chain computing based on partial observations of miners in mobile blockchain networks with multiple SPs*. Our contributions can be summarized as follows:

- We formulate the computation offloading issue for miners as a Markov game, with the objective of maximizing their long-term expected utilities by comprehensively considering offloading decision, server selection and block sizes. Specifically, states, observations, actions, rewards and state transition are formally defined based on the formulated Partially Observable Markov Decision Process (POMDP).
- To resolve the formulated problem, we first form a short-term optimization problem, and prove that it is consistent with the long-term one. Based on it, actions of miners can be reformed to make the variable action set be compatible with the designed learning model, and modules can be built for each miner to learn the targeted action elements.
- Since the decision of one miner is dependent on others, a mean-field learning based algorithm is proposed, where interactions of miners are approximated by the average effect from neighbors, to overcome the shortsighted view caused by local observations of miners. Also, a novel neural network model is built based on the actor-critic algorithm and mean field theory for each miner, which can be trained and executed in a distributed manner.
- Both theoretical and experimental analyses are conducted to demonstrate the effectiveness of the designed algorithm. We prove that a Nash equilibrium always exists, and the real-world map of Manhattan is utilized for the evaluation. Performance results show that our algorithm has significant advantages in terms of average miner utilities and convergence speeds.

The rest of this paper is structured as follows: in Section 2, we review the related work; we present the system model in Section 3; in Section 4, we formulate the Markov game based on POMDP; we design the learning-based computation offloading algorithm with multiple SPs for mobile blockchain in Section 5; after that, performance evaluation is conducted in Section 6; finally, we conclude this work in Section 7.

2 Related Work

We review the related work of computation offloading and learning algorithms for mobile blockchain in this section.

2.1 Computation Offloading for Mobile Blockchain

Computation offloading algorithms for mobile blockchain can be generally grouped into the following three categories.

For the first category, researchers aim to maximize utilities of miners. For example, authors in [12] propose a joint task offloading and resource allocation algorithm for video transcoding, with the purpose of maximizing the average reward for transcoders. The blockchain-based framework and adaptive block sizes are considered for the offloading algorithm. The second category is to maximize server utilities. A learning-based edge computing resource management scheme for blockchain networks is designed in [7], with the purpose of maximizing the long-term profits of edge servers. Miners can offload their tasks to edge servers in

the mining process, and iterations between servers and miners are modeled by a stochastic Stackelberg game. In [13], an auction mechanism for edge resource allocation in blockchain networks is designed, which allows mining tasks to be offloaded to edge servers. Similar with [7], the purpose is also to maximize the profits of edge servers.

For the last category, its objective is to maximize both utilities of miners and servers. An edge resource auction mechanism is proposed in [14], which intends to maximize the social welfare by considering the total ex-post valuation and the cost of edge servers. In [15], the objective is not only to minimize the system cost, but also to maximize privacy levels of miners. To achieve that purpose, a joint optimization problem is formulated as an Markov Decision Process (MDP), where user privacy, mining profits and task offloading are jointly considered. Authors in [16] form a two-stage Stackelberg game to both maximize profits of servers and utilities of miners. A lightweight infrastructure of blockchain with PoW consensus processes is designed to allow miners for task offloading. A cyclic block coordinate descent based algorithm is proposed in [17] to maximize the system reward of all edge servers and miners, where the computation resource allocation of edge servers and revenue-sharing from miners to servers are jointly optimized. Although these studies investigate offloading algorithms for mobile blockchain, they merely consider one SP, which manages all edge servers in the system.

2.2 Learning Algorithms for Mobile Blockchain

Authors in [18] integrate blockchain and DRL for content caching in vehicular networks. Blockchain can ensure distributed and secure content caching, while the DRL-based approach can be explored to obtain an efficient content caching scheme. Similar with [18], a blockchain-based caching scheme is proposed in [19], where the expected reward of mobile edge caching nodes can be satisfied. The formulated MDP problem focuses on caching placement and node selection, which can be solved by DRL. A joint resource allocation and computation offloading framework is proposed in [20], where the blockchain technology is applied to ensure data irreversibility and reliability for edge services. The transaction throughput of blockchain and the computation rate of edge servers can be both maximized through an Asynchronous Advantage Actor-Critic (A3C) based algorithm.

A blockchain-enabled data sharing strategy for Unmanned Aerial Vehicles (UAV)-aided networks is designed in [21], where transaction recording and misbehavior tracing can be realized for vehicles and UAVs in the disaster environment. A two-tier Reinforcement Learning (RL)-based incentive algorithm is proposed to guarantee data sharing with high-quality services. In [22], the blockchain is utilized to support the trust mechanism in MEC networks, where fake service records and selfish edge attacks can be detected. An A3C based learning algorithm is proposed in [23] to obtain the resource price and allocate edge resources for miners, where a balance between rewards and costs can be derived for miners.

Although existing studies leverage DRL for resource management, they do not consider the scenario that action spaces of miners change with time, and the decision making of miners is based on their local observations. To our best knowledge, this work is the first to support off-chain computing with multiple SPs and based on partial observations of miners, where task offloading, server selection and miner mobilities are jointly considered to maximize utilities of miners.

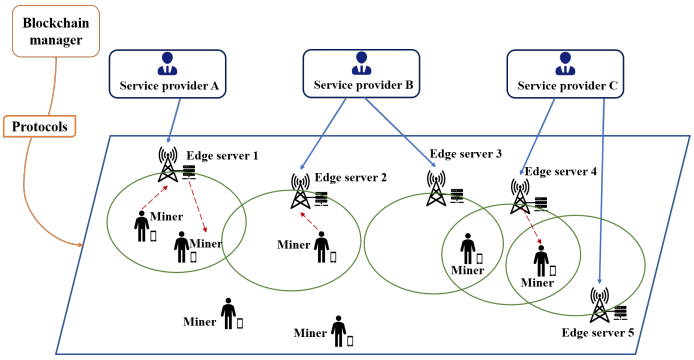


Fig. 1: Illustrative system model.

3 System Model

In this section, we first illustrate the system model. Similar with [14] and [7], we consider that a public blockchain application launched by a blockchain owner can be realized over an MEC network. Mobile users participant in the mining process, and can offload their computation tasks (solving the PoW puzzle) to edge servers managed by different SPs.

3.1 Network Model

As shown in Fig. 1, SPs, edge servers and mobile users coexist in the system. The set of SPs is denoted by $\mathbb{K} = \{1, ..., k, ..., K\}$, where K is the total number of SPs. Edge servers can be integrated with base stations to provide services for mobile users. The set of edge servers managed by SP k is denoted by $M_k = \{M_{k1}, ..., M_{ki}, ..., M_{k,m(k)}\}$, where $m(k)$ is the total number of edge servers managed by SP k. Thus, all edge servers can be grouped by $M = \{M_k\}_{k\in\mathbb{K}}$. The location of edge server M_{ki} managed by SP k can be represented by (x_{ki}, y_{ki}), where x_{ki} and y_{ki} are its horizontal and vertical coordinates, respectively. Generally, each edge server is connected to a based station via wired links. Then, the communication coverage radius of edge server $M_{ki} \in M$ is R, and its maximum computing resources in terms of Edge Computing Units (ECU) [7] can be denoted by f^c_{ki}. The total number of mobile users is N, and the set of mobile users participated in the blockchain application can be represented by $\mathbb{N} = \{1, \ldots, n, \ldots, N\}$, which can be called miners. Thus, miners and mobile users are interchangeable through the whole paper. Miner n always moves dynamically (they can take transportation tools) in the system with average speed ϑ_n, and its maximum ECU is f^u_n, where $f^u_n \ll f^c_{ki}$. The location of miner n can be represented by (x^τ_n, y^τ_n), where τ is the time instance in the system. Due to their limited computation capabilities, miners can offload their computation tasks related to the PoW puzzle to edge servers.

Miners compete against each other to solve the PoW puzzle through a try-and-guess strategy. The occurrence of

mining a block can be modeled by a Poisson process with mean rate of $1/T_b$, where T_b is the expected interval time between two blocks [24]. Then, the mining process can be represented by stages $\{0, 1, ..., t....\}$, and a new block is successfully committed into the global blockchain at the end of each stage. For stage 0, the start time instance is defined by 0, and that of stage t can be represented by τ^t. In stage t, the price of ECUs provided by SP k can be presented by p_k^t token/ECU·second, and the budget of miner n is B_n^t. In addition, the actual cost for mining a block in stage t is always no more than budget B_n^t. During one mining stage, all network parameters are constant, and all miners can participant in the mining process. When one miner has successfully mined a block, it immediately sends the block to others for verification, and the reward is sent to the first miner with a verified block, which is attached to the end of the main branch of the blockchain. Meanwhile, we assume that all miners and edge servers are faithful to follow blockchain protocols, and the security of the blockchain application can be guaranteed based on existing security technologies [25].

The communications between users and edge servers are based on the technology of Orthogonal Frequency Division Multiplexing (OFDM). The communication channel can be divided into several sub-channels, when multiple users access one edge server at the same time. Generally, the traditional wireless bandwidth based on 802.11 standard is 20 MHz, and the total number of sub-channels can reach 256, if Orthogonal Frequency-Division Multiple Access (OFDMA) technology is applied. In our system, uses can compute tasks either on one edge sever or locally, and can not always access one edge server simultaneously. Correspondingly, the number of sub-channel is enough since the maximum number of users and that of servers considered in our system are 110 and 15, respectively. Thus, the interference and competition for bandwidths among users can be neglected. In our blockchain-based systems, miners always offload their computation tasks related to the PoW puzzle to edge servers. Generally, the input of the PoW puzzle is the block head with 80 Bytes [2], and the communication cost for offloading the computational task can be also neglected, compared with the large cost for computing the PoW puzzle on edge servers.

3.2 Offloading Model

Miners can either offload their computation tasks related to the PoW puzzle to edge servers or process them locally. We assume that miners are aware of locations of edge servers, and can select one edge server along their routes for computation offloading. In practical scenarios, server locations can be detected by historical wireless beacon information and MAC address based on crowdsensing technology, and the received signal strength by the participant can be utilized to localize the server. Similar localization solutions can be found in [26] and [27], and we do not intend to specify this, since it is not our main focus. If miner n selects edge server M_{ki} at time instance τ, it should first move into the communication coverage of server M_{ki}, and then offload its computation task to M_{ki}. Different from existing edge computing schemes, we do not adopt any multi-hop data transmission strategy for tasks offloaded by miners to edge servers based on the consideration of security and data privacy. After completion, edge server M_{ki} returns back the processing result. Since the size of the result is small, the delay for returning back the result can be neglected. Thus, the delay for completing the computation task includes three parts, i.e., traveling, waiting and processing delays. The traveling delay refers to the delay for miner n to move into the communication coverage of server M_{ki}, i.e.,

$$d_{nki}^{r,t} = \begin{cases} \dfrac{r_{nki}^t - R}{\vartheta_n}, & \text{when} \quad r_{nki}^t > R; \\ 0, & \text{otherwise}, \end{cases} \tag{1}$$

where r_{nki}^t is the distance between miner n and server M_{ki} that is in the travel direction of miner n, when miner n makes offloading decisions in stage t. We assume that miners make offloading decisions at the beginning of each stage, and then $r_{nki}^t = \sqrt{(x_{ki} - x_n^{\tau^t})^2 + (y_{ki} - y_n^{\tau^t})^2}$ is founded. If miner n is in the communication coverage of server M_{ki} when making its offloading decisions, its traveling delay is 0.

Then, miner n can directly upload the computation task to server M_{ki} when arriving in its coverage. Generally, the input of the PoW puzzle is very small as analyzed in subsection 3.1, and thus the uploading delay from miner n to server M_{ki} can be neglected. After server M_{ki} receives computation tasks, it puts them in the waiting queue, which is processed in a First-In, First-Out (FIFO) order. Generally, the waiting time of the computation task can be obtained based on *M/G/1* queueing system [28]:

$$d_{nki}^{w,t} = \frac{\lambda_{ki}^t[(\bar{d}_{ki})^2 + \sigma^2]}{2(1 - \lambda_{ki}^t \bar{d}_{ki})}, \tag{2}$$

where λ_{ki}^t is the arrival task intensity of server M_{ki} in stage t, and $\bar{d}_{ki}$ is the average task processing delay of server M_{ki}. Symbol σ^2 is the variance of task processing delay.

The task processing delay caused by server M_{ki} for miner n can be computed by:

$$d_{nki}^{p,t} = \frac{F^t}{f_{ki}^c}, \tag{3}$$

where F^t is the required complexity in terms of ECUs to solve the PoW puzzle in stage t. If the computation task is processed locally by miner n, no traveling and waiting delays exist. Consequently, the local processing delay is computed by:

$$d_n^{l,t} = \frac{F^t}{f_n^u}. \tag{4}$$

We utilize α_n^t to represent whether the computation task of miner n is processed locally. If so, $\alpha_n^t = 1$ and verse visa. Similarly, β_{nki}^t is leveraged to denote whether the task is offloaded to server M_{ki}. Then, the total delay for completing the computation task of miner n in stage t can be expressed by:

$$d_n^{c,t} = \alpha_n^t d_n^{l,t} + (1 - \alpha_n^t) \sum_{k=1}^{K} \sum_{i=1}^{m(k)} \beta_{nki}^t \left[d_{nki}^{r,t} + d_{nki}^{w,t} + d_{nki}^{p,t} \right]. \tag{5}$$

3.3 Utility Model

In the public blockchain application, when miner n is the first to achieve a consensus on its generated block, it wins some gains from the blockchain manager. Otherwise, its gain is 0. According to [24], the gain of miner n in stage t can be computed by:

$$\mathfrak{R}_n^t = \begin{cases} \mathbb{R} + \varepsilon x_n^t, & \text{when miner } n \text{ wins;} \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

Symbol $\mathbb{R}$ is a fixed positive gain, and ε is a given variable gain factor. Variable x_n^t is the block size of miner n in stage t, referring to the number of transactions in the block. Generally, block size x_n^t does not affect the complexity of the mining process, but has impacts on the miner's reward and block propagation delay. When the computation task of miner n is finished (i.e., the PoW puzzle is solved), its generated block should be propagated to others for consensus achievement. On one hand, the larger size of the block is, the more reward can be obtained by the miner; on the other hand, a larger block size can cause a longer propagation delay, which increases the possibility of being orphaned (being discarded) [29].

The propagation delay can be formed by a linear relationship with block size x_n^t [14] [12], i.e.,

$$d_n^{o,t} = \kappa x_n^t, \tag{7}$$

where κ is a constant value reflecting the impact of block size x_n^t on propagation delay $d_n^{o,t}$. In addition, the cost of miner n can be denoted by:

$$\mathbb{C}_n^t = \begin{cases} \sum_{k=1}^{K} \sum_{i=1}^{m(k)} \beta_{nki}^t p_k^t d_{nki}^{p,t}, & \text{when} \quad \alpha_n^t = 0; \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

When $\alpha_n^t = 0$, miner n offloads its computation task to server M_{ki}; otherwise, it processes the computation task locally without any costs.

In traditional blockchain applications, the possibility for miner n to successfully solve a block is proportional to its hash rate for mining [30]. However, for the considered mobile blockchain applications, the corresponding possibility is not only related to hash rates (ECUs), but also depends on the traveling delay, since miners should move to the communication coverage of the server for task uploading. Thus, we define that the successful mining possibility is inversely proportional to the task completion time, due to the fact that the miner finishing solving the PoW puzzle with a shorter delay has a larger possibility to win the mining game. Similar with the definition in [7], we define the successful mining possibility as follows:

$$\mathbb{P}_n^{c,t} = \frac{\frac{\sum_{n' \in \mathbb{N}} d_{n'}^{c,t}}{d_n^{c,t}}}{\sum_{n' \in \mathbb{N}} \frac{\sum_{n' \in \mathbb{N}} d_{n'}^{c,t}}{d_{n'}^{c,t}}}, \tag{9}$$

where $\sum_{n' \in \mathbb{N}} d_{n'}^{c,t}$ is the sum of task completion delays for all miners, and $\sum_{n' \in \mathbb{N}} d_{n'}^{c,t} / d_n^{c,t}$ is the inverse ratio of miner n's task completion delay to the sum of task completion delays. The inverse proportional relationship between possibility $\mathbb{P}_n^{c,t}$ and delay $d_n^{c,t}$ is revealed in equation (9). As described before, the arrival of new blocks follows the Poisson process, and thus the orphaning probability of miner n in stage t can be expressed by:

$$\mathbb{P}_n^{h,t} = 1 - e^{-\frac{d_n^{o,t}}{T_b}}. \tag{10}$$

Consequently, the utility of miner n in stage t can be approximated as follows:

$$U_n^t = \begin{cases} \mathbb{R} + \varepsilon x_n^t - \mathbb{C}_n^t, & \text{with possibility} \quad \mathbb{P}_n^{c,t}(1 - \mathbb{P}_n^{h,t}); \\ -\mathbb{C}_n^t, & \text{with possibility} \quad 1 - \mathbb{P}_n^{c,t}(1 - \mathbb{P}_n^{h,t}). \end{cases} \tag{11}$$

4 Markov Game Formulation

In the mining process, miner n either offloads the computation task related to the PoW puzzle to one edge server, or processes it locally. We consider that the computation task is atomic, and can not be spitted into smaller ones as in [31]. Miners are totally competitive, since they intend to maximize their own utilities. Miners need to make proper offloading and block size decisions independently, although they are unaware of others' policies. We first model the mining process by a Markov game, and then formulate the optimization problem in this section. Main notations are listed in Table 1.

4.1 Game Definition

The task scheduling process of miners can be modeled by a POMDP, represented by tuple $\langle S, O, A, \mathcal{R}, P, N, \gamma \rangle$. The definitions of elements are illustrated as follows:

4.1.1 States and Observations

The state of the POMDP can be represented by $S \triangleq \{s^t = (S_1, S_2, S_3, S_4)\}$, $t \in \{0, 1, 2, ...\}$, where four components are included in each state:

1) S_1 represents the state of SPs, containing the total number, the resource price, the managed edge servers, represented by $S_1 = \{K, (p_k^t, M_k)_{k \in \mathbb{K}}\}$.

2) S_2 represents the state of edge servers, containing the location, the computation capability, the coverage radius, task arrival rates and the average task processing delay, expressed by $S_2 = \left\{x_{ki}, y_{ki}, f_{ki}^c, R, \lambda_{ki}^t, \bar{d}_{ki}\right\}_{k \in \mathbb{K}, 0 \le i \le m(k)}$.

3) S_3 denotes the state of miners, including the speed, the location, the budget, and the computation capability. Then, S_3 can be expressed by $S_3 = \left\{\vartheta_n, \left(x_n^{\tau^t}, y_n^{\tau^t}\right), B_n^t, f_n^u\right\}_{n \in \mathbb{N}}$.

4) S_4 is the state of blockchain, containing the complexity of the PoW puzzle, and the expected block generation interval, i.e., $S_4 = \{F^t, T_b\}$.

In stage t, miner n can not obtain the full observation of system state s^t, and can merely get its local observation o_n^t, which is a part of system state s^t. We consider that the following elements are observable to miner n, i.e., S_1, S_2 and S_4, but states of others are unknown to miner n.

4.1.2 Learning Agents, Actions and State Transition

Symbol N in the POMDP denotes that there are N learning agents in the system. That is, N miners act as learning agents, and they choose actions independently. Thus, miner n and learning agent n are interchangeable in this paper.

TABLE 1: Main notations

Notation	Description
$\mathbb{K}$, K	The set and the total number of SPs, respectively;
M_k	The set of edge servers managed by SP k;
$m(k)$	The total number of edge servers managed by SP k;
(x_{ki}, y_{ki})	The location of edge server M_{ki};
(x_n^τ, y_n^τ)	The location of miner n at time instance τ;
R	The communication coverage radius of edge server M_{ki};
ϑ_n	The speed of miner n;
f_n^u	The maximum ECUs of miner n;
τ	The time instance in the system;
f_{ki}^c	The maximum ECUs of server M_{ki};
T_b	The expected interval time between two blocks;
p_k^t	The price of computation resources provided by SP k in stage t;
B_n^t	The budget of miner n in stage t;
τ^t	The start time instance of stage t;
r_{nki}^t	The distance between miner n and server M_{ki} in stage t;
$d_{nki}^{r,t}$	The traveling delay for miner n to move into the communication coverage of server M_{ki} in stage t;
$d_{nki}^{w,t}$	The waiting time for the computation task of miner n in the processing queue of server M_{ki} in stage t;
$d_{nki}^{p,t}$	The processing delay for the task offloaded by miner n and processed by server M_{ki} in stage t;
λ_{ki}^t	The arrival task intensity of server M_{ki} in stage t;
$\bar{d}_{ki}$	The average task processing delay of server M_{ki};
σ^2	The variance of task processing delay;
F^t	The required complexity in terms of ECUs to solve the PoW puzzle in stage t;
$d_n^{l,t}$	The task processing delay caused by local processing of miner n in stage t;
$d_n^{c,t}$	The total delay for completing the computation task of miner n in stage t;
α_n^t	Binary value to denote whether the computation task of miner n is processed locally in stage t;
β_{nki}^t	Binary value to denote whether the task is offloaded to server M_{ki};
$\mathfrak{R}_n^t$, $\mathbb{C}_n^t$	The gain and the cost of miner n, respectively;
x_n^t	The block size of miner n in stage t;
$d_n^{o,t}$	The propagation delay of the block generated by miner n in stage t;
κ	A constant value reflecting the impact of block size x_n^t on propagation delay $d_n^{o,t}$;
$\mathbb{P}_n^{c,t}$	The possibility for successfully mining a block by miner n in stage t;
$\mathbb{P}_n^{h,t}$	The orphaning probability of miner n in stage t;
U_n^t	The utility of miner n in stage t;
a_n^t	The action of learning agent n in stage t;
$\mathcal{R}_n^t$	The reward of miner n in stage t;
$\hat{M}_n^t$	The available set of edge servers for task offloading by miner n in stage t;
o_n^t	The observation of miner n in stage t;
a_n^t	The action of learning agent n in stage t;
$v_n(s)$	The value function of miner n at state s;
$\mathbb{R}$	A fixed positive gain for $\mathfrak{R}_n^t$;
ε	A given variable gain factor for $\mathfrak{R}_n^t$;
$Q_{\pi_{-n}^*}(s^t, a_n^t)$	The marginal value for picking action a_n^t by miner n at state s^t;
$\pi(s^t, a^t)$	The possibility of picking action a^t by all miners at state s^t;
N_n^t	The total number of neighbors of miner n in stage t;
$\bar{a}_n^t$	The mean action of neighbors of miner n in stage t;
$L(\psi_n)$	The loss function for the value network of miner n;
L_p	The total loss for the policy network;
L_r	The loss function for offloading regulation and block size regulation modules;
μ_n, η_n	Offloading and block size modules of learning agent n, respectively.

The action space can be represented by $A \triangleq \{a^t = (a_n^t)\}_{n\in\mathbb{N}}$, where a^t is the joint action of all learning agents in stage t, and a_n^t is the action of learning agent n in stage t. Action a_n^t can be defined by $a_n^t = \{\alpha_n^t, \beta_{nki}^t, x_n^t\}$. From Section 3, we know that $\alpha_n^t \in \{0, 1\}$ and $\beta_{nki}^t \in \{0, 1\}$, when $k \in \mathbb{K}$ and $0 \le i \le m(k)$. However, for miner n in stage t, not all edge servers are available for its selection, due to its limited traveling speed and the distance from edge servers. That is to say, if miner n can not move into the communication coverage of edge server M_{ki} in stage t, it does not intend to select M_{ki} for task processing, since that leads to no successful possibility for mining. As described in subsection 3.1, the expected interval time between two blocks is T_b, which is generally set by 10 minutes in the literature. Thus, the delay for miner n to move into the communication coverage of server M_{ki} and waiting for processing can not exceed T_b. Based on the above analysis, we can draw the following theorem:

Theorem 1. *In stage t, the set of edge servers that is available for miner n to offload computation tasks can be represented by $\hat{M}_n^t = \{M_{ki}\} \in M$, which should satisfy the following two conditions:*

a) The distance between miner n and edge server M_{ki} should satisfy $r_{nki}^t \le T_b\vartheta_n - R^{'}$, where $R^{'}$ is the distance for miner n to move out of the communication coverage of edge server M_{ki};

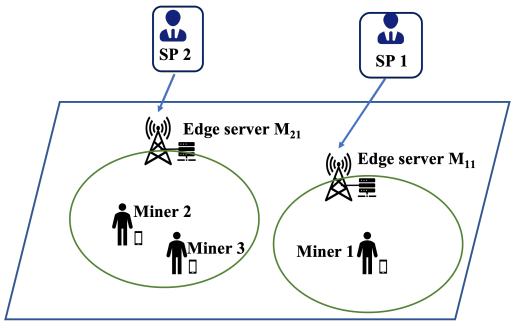
b) The task intensity of edge server M_{ki} should meet $\lambda_{ki}^t \le (2D_1 - \sigma^2)/((\bar{d}_{ki})^2 + 2D_1\bar{d}_{ki})$, where $D_1 = R^{'}/\vartheta_n - F^t/f_{ki}^c$.

The proof can be found in Appendix A of Supplemental File.

From Theorem 1, we can discover that the action space of miner n in each stage is not always the same. This is because miner n travels along its route, resulting in the change of its location. Then, the set of available edge servers that miner n can access also changes with time. Thus, we can obtain the following corollary:

Corollary 1. *The action space for miner n changes with mining stages.*

The state transition probability distribution is denoted by $P : S \times A \times S \to [0, 1]$, and $\rho_0 : S \to [0, 1]$ is the distribution of initial state s^0. State s^t can be transferred into s^{t+1} by taking action a^t based on probability $P(s^{t+1}|s^t, a^t)$. Let us take a simplified example by considering two servers and three miners in the system, where server M_{11} is managed by SP 1 and server M_{21} is manged by SP 2. Miner 1 is in the communication coverage of server M_{11}, and miners 2 and 3 are in the coverage of server M_{21}. In stage t, the service price of SP 1 is $p_1^t = 2$ \$/ECU and that of SP 2 is $p_2^t = 3$ \$/ECU, while the budgets of the three miners are $B_1^t = 4$ \$, $B_2^t = 3$ \$ and $B_3^t = 6$ \$, respectively. Miner 1 has two choices, i.e., offloading the computational task to server M_{11}, and processing it locally. We assume that offloading task to server M_{11} ($\alpha_1^t = 0$, $\beta_{111}^t = 1$, $\beta_{121}^t = 0$) costs 2 \$ with task completion delay of 5 minutes, while processing locally ($\alpha_1^t = 1$) costs 0 \$ with 11 minutes delay. Miner 2 can offload the computational task to server M_{21} ($\alpha_2^t = 0$, $\beta_{211}^t = 0$, $\beta_{221}^t = 1$) with 3 \$ cost and 6 minutes delay, while local computing ($\alpha_2^t = 1$) with 0 \$ cost and 10 minutes delay. For miner 3, it costs 4 \$ with 7 minutes delay for offloading ($\alpha_3^t = 0$, $\beta_{311}^t = 0$, $\beta_{321}^t = 1$) while costs 0 \$



System state (stage t)	Action			System state (stage t+1)
	Miner 1	Miner 2	Miner 3	
{2,3,4,3,6}	{1,{0,0}}	{1,{0,0}}	{1,{0,0}}	{1,2,4,3,11}
	{1,{0,0}}	{1,{0,0}}	{0,{0,1}}	{1,3,4,3,7}
	{1,{0,0}}	{0,{0,1}}	{0,{0,1}}	{1,4,4,5,2}
	{0,{1,0}}	{1,{0,0}}	{1,{0,0}}	{3,2,7,3,6}
	{0,{1,0}}	{1,{0,0}}	{0,{0,1}}	{3,4,7,3,2}
	{0,{1,0}}	{0,{0,1}}	{0,{0,1}}	{3,5,7,0,2}

Fig. 2: An example of state transition.

with 9 minutes delay for local computing ($\alpha_3^t = 1$). For simplicity, we consider block sizes of the three miners are the same. Thus, whether the mining game can be won is totally dependent on the task completion delay. We consider that the reward for winning the mining game in stage t is 5 \$. Then, the system state transition is shown in Fig. 2.

The system state is presented in form of $\{p_1^t, p_2^t, B_1^t, B_2^t, B_3^t\}$, and the action of miner n is presented by $\{\alpha_n^t, \{\beta_{n11}^t, \beta_{n21}^t\}\}$. For instance, when actions of three miners are all $\{1, \{0, 0\}\}$ as shown in the first row of the table in Fig. 2, i.e., the three miners all adopt local computing. Then, miner 3 with the minimum task completion delay wins the reward, and its budget becomes 11. Followed that, the system state transmits into $\{1, 2, 4, 3, 11\}$. For the service price of SPs, we assume that they have their own mechanism to change their own prices, and we randomly set their values since this is not our main focus.

4.1.3 Rewards and The Discounted Factor

Symbol $\gamma^t \in [0, 1]$ is a discounted factor that scales the impact of future rewards in the training process of mining stage t. After taking action a^t at state s^t, the immediate reward received by miner n can be represented by $\mathcal{R}_n^t : S \times A \rightarrow \mathcal{R}$. Since our objective is to maximize utilities of miners from a long-term perspective, we define $\mathcal{R}_n(s^t, a^t) = U_n^t$, simply represented by $\mathcal{R}_n^t$. When miner n wins in stage t and uses up all its budget B_n^t, its reward reaches the minimum, i.e.,

$$\mathcal{R}_n^t \geq \mathbb{R} + \varepsilon x_n^t - B_n^t. \tag{12}$$

To maximize their rewards, miners should always keep their minimum reward above zero when they win. Thus, the following theorem can be deduced:

Theorem 2. *The available range of block size x_n^t can be computed by $\max\left\{0, \lceil \frac{B_n^t - \mathbb{R}}{\varepsilon} \rceil\right\} \leq x_n^t \leq \lfloor \frac{T_b}{\kappa} \rfloor$ in stage t.*

The proof can be found in Appendix B of Supplemental File.

Based on Theorem 2, we can obtain the following theorem:

Theorem 3. *In stage t, budget B_n^t of miner n satisfies $B_n^t \leq \min\left\{\frac{\varepsilon T_b}{\kappa} + \mathbb{R}, \mathbb{U}_n^t\right\}$, where $\mathbb{U}_n^t$ is the current property of miner n, and can be computed by:*

$$\mathbb{U}_n^t = \begin{cases} B_n^0 + \sum_{\tau=0}^{t-1} U_n^\tau, & t > 0; \\ B_n^0, & t = 0. \end{cases} \tag{13}$$

The proof can be found in Appendix C of Supplemental File.

4.2 Problem Formulation

In the formulated Markov game, we need to find policy $\pi = \{\pi_1, ..., \pi_n, ..., \pi_N\}$ that can maximize the long-term expected cumulative discounted rewards for all miners. Then, our optimization problem can be formulated by:

$$\text{P1:} \quad \max_{\pi_n} \sum_{t=0}^{\infty} \gamma^t E\left[\mathcal{R}_n^t\right], \quad \forall n \in \mathbb{N}, \tag{14}$$

$$\text{s.t.} \quad \max\left\{0, \lceil \frac{B_n^t - \mathbb{R}}{\varepsilon} \rceil\right\} \leq x_n^t \leq \lfloor \frac{T_b}{\kappa} \rfloor, \tag{14.a}$$

$$\mathbb{C}_n^t \leq c, \tag{14.b}$$

$$\sum_{k=1}^{K} \sum_{i=1}^{m(k)} \beta_{nki}^t = 1, \tag{14.c}$$

$$\lambda_{ki}^t \leq (2D_1 - \sigma^2)/((\bar{d}_{ki})^2 + 2D_1\bar{d}_{ki}), \tag{14.d}$$

$$r_{nki}^t \leq T_b\vartheta_n - R^{'}. \tag{14.e}$$

As defined in Theorem 2, constraint (14.a) guarantees the value of block size x_n^t is in the available range. Constraint (14.b) makes cost C_n^t of miner n less than budget B_n^t, and constraint (14.c) guarantees the task of miner n can merely select one edge server for offloading or local computing in stage t. The definition of constraints (14.d) and (14.e) can be found in Theorem 1, to make the set of edge servers available for miner n to offload tasks.

However, the coupled parameters and their interdependence make the above formulated optimization problem difficult to resolve. The reasons are: a) The dependency of offloading decisions on others' actions, i.e., whether the miner can win the mining game, not only depends on its own decision, but also is affected by others; b) The trade-off between miners' rewards and block sizes, i.e., a large block size can both increase the reward and the orphaning probability; If the orphaning probability is increased, the miner may not win the mining game; c) To achieve the objective defined in Problem P1, miner n should maximize its own long-term expected rewards. However, optimizing the objective for each miner depends on the joint actions of all miners. Therefore, the concept of Nash equilibrium is important for this kind of problems. In a Nash equilibrium, each miner acts with the best response to others based on their best policies, which is equivalent to maximizing their own utilities provided that others follow their best policies. It has been revealed that at least one Nash equilibrium exists with

stationary policies [32]. In our system, the Nash equilibrium among miners is hard to reach based on local observations of miners. In the next section, we propose a mean-field learning-based computation offloading algorithm to solve the formulated optimization problem distributively.

5 A Mean-Field Learning Based Algorithm

We propose an online learning algorithm to solve Problem P1, and the designed algorithm can be trained and executed in a distributed manner. We first formulate the short-term optimization problem in subsection 5.1, based on which miner actions are reformed. Then, the whole algorithm is presented in subsection 5.2, where the learning model is specified on initialization, action execution, batch data collection and network training.

5.1 Short-Term Optimization

In this subsection, we first formulate the local optimization problem, and then reform miner actions. At last, interactions among miners are approximated to make the formulation problem solvable for miners.

5.1.1 Short-Term Optimization Problem Formulation

To solve Problem P1, we need to compute three values α_n^t, β_{nki}^t and x_n^t, which have also been included in action a_n^t of miner n. However, the three elements are interdependent on each other, and can not be directly obtained through a neural network, since the single neural network can not reflect their interrelationships. Thus, we first need to separate them from the whole objective, and form their local optimization problems to derive their optimal values.

In subsection 3.3, we have defined the utility function of miner n, and the relationship of three elements α_n^t, β_{nki}^t and x_n^t is revealed in equation (11). Based on that, we can compute the expected utility value of miner n by:

$$\begin{aligned} E\left[U_n^t\right] = &\left(\mathbb{R} + \varepsilon x_n^t - \mathbb{C}_n^t\right)\left[\mathbb{P}_n^{c,t}(1-\mathbb{P}_n^{h,t})\right] \\ &+ \left(-\mathbb{C}_n^t\right)\left[1-\mathbb{P}_n^{c,t}(1-\mathbb{P}_n^{h,t})\right]. \end{aligned} \tag{15}$$

To maximize $E\left[U_n^t\right]$ in mining stage t, we need to maximize possibility $\mathbb{P}_n^{c,t}$ for successfully mining a block as well as gain $\mathfrak{R}_n^t$, and minimize orphaning probability $\mathbb{P}_n^{h,t}$ as well as cost $\mathbb{C}_n^t$. Based on this condition, we can obtain the short-term optimization problem as follows:

$$\text{P2:} \quad \begin{cases} \max\limits_{\alpha_n^t, \beta_{nki}^t} \dfrac{\mathbb{P}_n^{c,t}}{\mathbb{C}_n^t + \varpi}; \\ \min\limits_{x_n^t} \mathfrak{R}_n^t(\mathbb{P}_n^{h,t} - 1). \end{cases} \tag{16}$$

$$\text{s.t.} \quad \text{inequations (14.a) - (14.e)},$$

where ϖ is a small fixed positive value to ensure the denominator is not zero. By solving those two functions in Problem P2, we can obtain the optimal value of α_n^t, β_{nki}^t and x_n^t for short-term optimization. In addition, we derive Theorem 4 for the relationship between the short-term optimization objective and the long-term one defined in subsection 4.2.

Theorem 4. *The short-term optimization objective defined in Problem P2 is consistent with the long-term one in Problem P1.*

The proof can be found in Appendix D of Supplemental File.

Due to the dependence on other miners' task completion delay $d_n^{c,t}$ and the variable range of β_{nki}^t caused by the movement of miners, actions of miners need to be further reformed.

5.1.2 Action Reformation

According to Problem P2, the three elements in action a_n^t can be divided into two groups: one is $a_{n,1}^t = \{\alpha_n^t, \beta_{nki}^t\}$, and the other is $a_{n,2}^t = \{x_n^t\}$. For the former one, the available value of β_{nki}^t for miner n changes from stage to stage as demonstrated in Corollary 1. This is because miners move along the road, and the set of available edge servers for computation offloading within their communication coverage is also changed. Generally, the policy network, built on neural networks of DRL, generates action chosen possibilities among a fixed set of candidate actions by an output layer. However, the variable action set in our system makes DRL algorithm unfitness. For the latter one, it can be discretized into L_n values to make it be consistent with other action elements.

To solve Problem P2, we first design two modules for the policy network of miner n, i.e., offloading module μ_n and block size module η_n. For the former one, we utilize observation and action embeddings as the input of offloading module μ_n similar with the policy in [33]. The size of the action space for offloading module μ_n is $1 + |\hat{M}_n^t|$, where $\hat{M}_n^t$ is the available set of edge servers for task offloading by miner n in stage t, and can be obtained based on Theorem 1. Symbol $|\hat{M}_n^t|$ is the size of set $\hat{M}_n^t$, and the output of offloading module μ_n is $a_{n,1}^t = \{\alpha_n^t, \beta_{nki}^t\}$. When miner n decides to process the computation task locally, i.e., $\alpha_n^t = 1$, the choice for β_{nki}^t is meaningless. Consequently, there are $1+|\hat{M}_n^t|$ available choices for the output action of offloading module μ_n. Then, we form $1 + |\hat{M}_n^t|$ observation-action pairs, each of which is represented by $\left(o_n^t, a_{n,1,g}^t\right)$, where $0 < g \leq 1 + |\hat{M}_n^t|$.

After that, we randomly choose Λ_n candidates from the observation-action pair set. For each candidate pair, it is input into offloading module μ_n, and the output ranking value is $\mu_n(o_n^t, a_{n,1,g}^t)$. Then, the Λ_n ranking values are fed into a selector realized by Boltzmann exploration strategy [34], and the action chosen probability of each candidate is computed by:

$$\mu_n(a_{n,1,g}^t | o_n^t) = \frac{e^{\chi \mu_n(o_n^t, a_{n,1,g}^t)}}{\sum_{g'=1}^{\Lambda_n} e^{\chi \mu_n(o_n^t, a_{n,1,g'}^t)}}, \tag{17}$$

where χ is a temperature parameter controlling the exploration rate. A higher value of χ can lead the selector to a fully greedy strategy, while a lower value makes it closer to a purely random one. Based on chosen probability $\mu_n(a_{n,1,g}^t | o_n^t)$, we can choose the best value for action elements α_n^t and β_{nki}^t.

5.1.3 Interaction Approximation

Both of Problems P1 and P2 for miner n depend on actions of other miners, and thus we approximate interactions

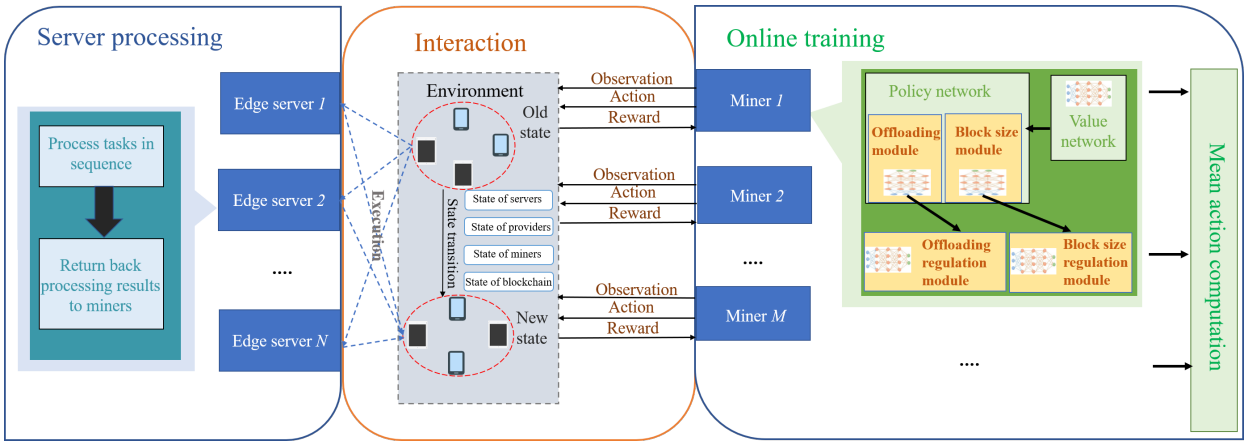


Fig. 3: Structure of the designed algorithm.

among miners to model others' policies. Generally, the value function of miner n is defined by its expected reward, i.e.,

$$\begin{aligned} v_n(s) &= E_\pi \left[\sum_t \gamma^t E(\mathcal{R}_n^t) \right] \\ &= E_\pi \left[\sum_t \gamma^t \sum_{a^t \in A} \mathcal{R}_n(s^t, a^t) \pi(s^t, a^t) \right], \end{aligned} \tag{18}$$

where $\pi(s^t, a^t)$ is the possibility of picking action a^t by all miners at state s^t. In our system, miners pick up actions independently without the awareness of others' policies. Thus, possibility $\pi(s^t, a^t)$ can be computed by $\pi(s^t, a^t) = \prod_{n=1}^{N} \pi_n(s^t, a_n^t)$, where $\pi_n(s^t, a_n^t)$ is the possibility of picking action a_n^t by miner n at state s^t. Then, $\pi_{-n}(s^t, a_{-n}^t)$ can be regarded as the possibility of picking actions a_{-n}^t by others expect miner n at state s^t, i.e., $\pi_{-n}(s^t, a_{-n}^t) = \prod_{n'=1, n' \neq n}^{N} \pi_{n'}(s^t, a_{n'}^t)$. Based on equation (18), Problem P1 can be written into:

$$\begin{aligned} \text{P3:} \quad & v_n^*(s^t) = \max_{\pi_n} v_n(s^t) \\ & = \max_{\pi_n} \left\{ E_{\pi_n} Q_{\pi_{-n}^*}(s^t, a^t) \right\}, \\ \text{s.t.} \quad & \text{inequations (14.a) - (14.e),} \end{aligned} \tag{19}$$

where

$$\begin{aligned} Q_{\pi_{-n}^*}(s^t, a^t) = E_{\pi_{-n}^*} \big[& \mathcal{R}_n(s^t, a^t) \\ & + \gamma^t \sum_{s^{t+1}} P(s^{t+1} | s^t, a^t) v_n(s^{t+1}) \big]. \end{aligned} \tag{20}$$

Variable $Q_{\pi_{-n}^*}(s^t, a^t)$ represents the marginal value for picking action a_n^t by miner n at state s^t. In our system, due to the local observation of miners, they can not obtain state s^t. Thus, we utilize o_n^t to replace s^t, and formulate the following problem to approach Problem P3:

$$\begin{aligned} \text{P4:} \quad & v_n^*(s^t) = \max_{\pi_n} v_n(o_n^t) \\ & = \max_{\pi_n} \left\{ E_{\pi_n} Q_{\pi_{-n}^*}(o_n^t, a^t) \right\}, \\ \text{s.t.} \quad & \text{inequations (14.a) - (14.e).} \end{aligned} \tag{21}$$

Since Q-function $Q_{\pi_{-n}}(o_n^t, a^t)$ depends on a_{-n}^t, which can not be obtained by miner n, we can approximate $Q_{\pi_{-n}}(o_n^t, a^t)$ based on mean field theory [35]. That is, one single miner playing with the average effect from its neighbors can be utilized to approximate interactions among all miners. Herein, we consider that possible actions of miners are expressed in one-hot encodings [36], and define $\bar{a}_n^t$ as the mean action of neighbors of miner n, i.e.,

$$\bar{a}_n^t = \frac{\sum_{n'} a_{n'}^t}{N_n^t}, \tag{22}$$

where N_n^t is the total number of neighbors of miner n in stage t. Then, action $a_{n'}^t$ of neighbor n' can be expressed by $a_{n'}^t = \bar{a}_n^t + \xi a_{n,n'}$, where the product of ξ and $a_{n,n'}$ represents a small fluctuation from the mean action. Then, Theorem 5 is founded for Q-function:

Theorem 5. *Q-function $Q_{\pi_{-n}^*}(o_n^t, a^t)$ of miner n in stage t can be approximated by mean action $\bar{a}_n^t$ and action a_n^t, i.e.,*

$$Q_{\pi_{-n}^*}(o_n^t, a^t) \triangleq Q_{\pi_{-n}^*}(o_n^t, a_n^t, \bar{a}_n^t). \tag{23}$$

The proof can be found in Appendix F of Supplemental File.

Thus, Q-function $Q_{\pi_{-n}^*}(o_n^t, a^t)$ can be approximated by $Q_{\pi_{-n}^*}(o_n^t, a_n^t, \bar{a}_n^t)$. Miner n does not need to know the policies of others, and merely needs to approximate $Q_{\pi_{-n}^*}(o_n^t, a^t)$ based on neighbors' actions. Then, miner n should solve the following problem:

$$\begin{aligned} \text{P5:} \quad & v_n^*(s^t, \bar{a}_n^t) = \max_{\pi_n} v_n(o_n^t, \bar{a}_n^t), \\ \text{s.t.} \quad & \text{inequations (14.a) - (14.e),} \end{aligned} \tag{24}$$

where

$$v_n(o_n^t, \bar{a}_n^t) = E_{a_n^t \sim \pi_n} \left[Q_{\bar{a}_n^t \sim \pi_{-n}^*}(o_n^t, a_n^t, \bar{a}_n^t) \right]. \tag{25}$$

5.2 The Whole Algorithm

In our considered system, miners decide whether to offload their computation tasks related to the PoW puzzle in each stage based on their own budgets and utilities. If miner n decides to offload its task, when and where to offload should be determined. The above process can be formed by a Markov game, and expressed by a long-term expected reward maximization problem as shown in Problem P1. To reform miner actions, we form short-term optimization problem P2 and prove that it is consistent with long-term optimization problem P1. However, Problems P1 and P2

are both dependent on policies of other miners. Thus, we approximate the interaction among miners based on mean field theory, where the average effect from neighbors can be utilized to train the learning model. Then, Q-function $Q_{\pi^*_{-n}}(o^t_n, a^t)$ can be approximated by $Q_{\pi^*_{-n}}(o^t_n, a^t_n, \bar{a}^t_n)$. Each miner merely needs to solve Problem P5, and maintains as well as trains its constructed neural networks locally. The training process of miner n can be specified as follows:

Algorithm 1 Pseudo-code of training processes for miners

Input: Batch size B, initial policies with policy parameters $\{\omega_n\}^N_{n=1}$, offloading regulation parameters $\{\iota_n\}^N_{n=1}$, block size regulation parameters $\{\nu_n\}^N_{n=1}$ and value parameters $\{\psi_n\}^N_{n=1}$
Output: Learned policy $\{\pi_{\omega_n}\}^N_{n=1}$.
for round $k = 1, 2, ...$ **do**
  Sample a mini-batch of state transition $\left(\{o^t_n\}^N_{n=1}, \{a^t_n\}^N_{n=1}, \{\mathcal{R}^t_n\}^N_{n=1}, \{o^{t+1}_n\}^N_{n=1}\right)$ and $\left(\{\varphi_n(a^t_{n,1}, a^t_{-n,1})\}^N_{n=1}, \{\rho_n(a^t_{n,2}, a^t_{-n,2})\}^N_{n=1}\right)$ with size B, where actions $\{a^t_n\}^N_{n=1}$ are sampled by Boltzmann softmax selector based on equation (17).
  Compute mean action $\{\bar{a}^t_n\}^N_{n=1}$ by equation (22).
  Store $\left(O, A, \bar{A}, \mathcal{R}, o^{'}, \varphi, \rho\right)$ in buffer $\mathscr{B}$.
  for miner $n = 1, 2, ..., N$ **do**
    Sample a mini-batch of miner n from buffer $\mathscr{B}$.
    Compute predicted value Y^t_n according to equation (30).
    Update the value network by minimizing the loss function in equation (29).
    Update the policy network by minimizing the loss function in equation (32).
    Update offloading and block size regulation modules by minimizing the loss function in equation (33).
  end for
end for

5.2.1 Neural Network Initialization

Miner n needs to construct its own learning model. Similar with [37] and [38], we consider that the actor-critic algorithm is adopted by each miner to improve the learning efficiency. Then, the policy network acts as the actor, and the value network is the critic. The actor needs to determine $\{\alpha^t_n, \beta^t_{nki}, x^t_n\}$, and the critic computes the value function and scores actions chosen by the actor. Recall that, we reform action a^t_n that includes two elements as illustrated in subsection 5.1.2, i.e., $a^t_n = \{a^t_{n,1}, a^t_{n,2}\}$, where $a^t_{n,1} = \{\alpha^t_n, \beta^t_{nki}\}$ and $a^t_{n,2} = \{x^t_n\}$. For action $a^t_{n,1}$, offloading module μ_n formed by neural networks with optimization parameter θ_n is constructed. For action $a^t_{n,2}$, block size module η_n with parameter ϕ_n is formed. Thus, policy π_n can be approximated by tuple (μ_n, η_n). For the value network, i.e., the critic, its optimization parameter is ψ_n.

In subsection 5.1.1, we form the short-term optimization problem in Problem P2, and prove that they are consistent

Algorithm 2 Pseudo-code of the LOBE algorithm

Input: State S_1 of SP, state S_2 of edge servers and state S_4 of blockchain.
Output: Reward $E(\mathcal{R}^t_n)$ of miners.
for stage $t = 0, 1, 2, ...$ **do**
  for miner $n = 1, 2, ..., N$ **do**
    Get local observation o^t_n.
    Get action a^t_n by the learning model trained based on Algorithm 1.
    if $\alpha^t_n == 1$ **then**
      Process the computation task locally.
    else
      Select the edge server and the block size according to action a^t_n.
      Offload the computation task to the selected edge server.
    end if
  end for
  for SP $k = 1, 2, ...$ **do**
    for edge server M_{ki}, where $i = 1, 2, ..., m(k)$ **do**
      Process tasks in the waiting queue.
      Return back the processing result.
    end for
  end for
  for miner $n = 1, 2, ..., N$ **do**
    Broadcast its block in the system.
    Compute its cost.
  end for
  The blockchain manager decides the winner and rewards it.
end for

with the long-term one defined in Problem P1. Then, we can define the following function based on mean field theory:

$$\begin{aligned}\varphi(\alpha^t_n, \beta^t_{nki}, \alpha^t_{-n}, \beta^t_{-nki}) &= \varphi(a^t_{n,1}, a^t_{-n,1}) \\ &= \varphi(a^t_1) = \frac{1}{N^t_n}\sum_{n'}\varphi(a^t_{n,1}, a^t_{n',1}) \\ &\approx \varphi(a^t_{n,1}, \bar{a}^t_{n,1}) = -\frac{\mathbb{P}^{c,t}_n}{\mathbb{C}^t_n + \varpi}.\end{aligned} \tag{26}$$

Similar with the proof of Theorem 5, function $\varphi(a^t_{n,1}, a^t_{n',1})$ can be expressed in the form of Taylor's expansion, i.e.,

$$\begin{aligned}\varphi(a^t_1) &= \frac{1}{N^t_n}\sum_{n'}\varphi(a^t_{n,1}, a^t_{n',1}) \\ &= \frac{1}{N^t_n}\sum_{n'}\varphi(a^t_{n,1}, \bar{a}^t_{n,1}) \\ &+ \bigtriangledown_{\bar{a}^t_{n,1}}\varphi(a^t_{n,1}, \bar{a}^t_{n,1})\left[\frac{1}{N^t_n}\sum_{n'}\xi a_{n,n',1}\right] \\ &+ \frac{1}{2N^t_n}\sum_{n'}\left[\bigtriangledown^2_{\tilde{a}^t_{n,1}}\varphi(a^t_{n,1}, \tilde{a}^t_{n,1})\xi^2 a^2_{n,n',1}\right].\end{aligned} \tag{27}$$

Since the last two parts in equation (27) approach to 0, then we can obtain equation (26). Similar with equation (27), we can also obtain the following expression:

$$\begin{aligned}\rho(x^t_n, x^t_{-n}) &= \rho(a^t_{n,2}, a^t_{-n,2}) \\ &\approx \rho(a^t_{n,2}, \bar{a}^t_n) = \mathfrak{R}^t_n(\mathbb{P}^{h,t}_n - 1).\end{aligned} \tag{28}$$

To make the output actions of the learning model accurate, and regularize miner behaviors in each stage, we construct other two modules, i.e., offloading regulation module $\hat{\varphi}_n$ and block size regulation module $\hat{\rho}_n$ to approximate functions defined in equations (26) and (28). The former one is with optimization parameter ι_n, and its input is the output of offloading module θ_n. The latter one is with parameter ν_n, and its input is the output of block size module η_n.

5.2.2 Action Execution

In stage t, miner n chooses action a_n^t based on its policy π_n. First, miner n inputs its observation o_n^t into offloading module μ_n and block size module η_n, and its output is $\pi_n(o_n^t, a_n^t, \bar{a}_n^t) = (\mu_n(o_n^t, a_{n,1}^t, \bar{a}_n^t), \eta_n(o_n^t, a_{n,2}^t, \bar{a}_n^t))$. Based on the output possibilities, miner n decides whether to offload computation tasks, which edge server to select, and how to set the block size.

5.2.3 Batch Data Collection

Aiming to make the learning model obtain good scheduling results, behavior trajectories of miners are collected in mini-batches. Their local observations, actions, predicated state values, as well as outputs of offloading, block size, offloading regulation, and block size regulation modules are recorded together in batches for each miner. Then, miners train their local learning models to minimize losses based on the collected batch data.

5.2.4 Model Training

For miner n, it needs to train its own learning model independently. Since we adopt the actor-critic algorithm, the loss function for the value network is:

$$L(\psi_n) = E[\| Y_n^t - Q_{\psi_n}(o_n^t, a_n^t, \bar{a}_n^t) \|^2], \tag{29}$$

where

$$Y_n^t = \mathcal{R}_n^t + \gamma^t V_{\psi_n}(o_n^{t+1}, \bar{a}_n^{t+1}). \tag{30}$$

To train the value network, we merely need to minimize the loss function defined in equation (29). For the policy network, modules of offloading, block size, offloading regulation, and block size regulation need to be all trained. First, to realize the long-term optimization objective, the following loss gradient for the policy network is defined:

$$\bigtriangledown L_\pi(\omega_n) = E[\bigtriangledown_{\omega_n} \log \pi_{\omega_n}(o_n^t) Q_{\psi_n}(o_n^t, a_n^t, \bar{a}_n^t)], \tag{31}$$

where parameter $\omega_n \approx (\theta_n, \phi_n)$. To realize the short-term optimization objective, we formulate losses $L_{\hat{\varphi}_n}(\iota_n) = E[\hat{\varphi}_n(a_{n,1}^t, \bar{a}_n^t)]$ and $L_{\hat{\rho}_n}(\nu_n) = E[\hat{\rho}_n(a_{n,2}^t, \bar{a}_n^t)]$. Then, the total loss for the policy network is:

$$L_p = L_\pi(\omega_n) + g_{\hat{\varphi}_n} L_{\hat{\varphi}_n}(\iota_n) + g_{\hat{\rho}_n} L_{\hat{\rho}_n}(\nu_n), \tag{32}$$

where $g_{\hat{\varphi}_n}$ and $g_{\hat{\rho}_n}$ are weighted factors for losses $L_{\hat{\varphi}_n}(\iota_n)$ and $L_{\hat{\rho}_n}(\nu_n)$, respectively. In addition, to make offloading regulation and block size regulation modules have good approximations, we train them by minimizing the mean square error between actual values and predicted ones, which is,

$$\begin{aligned} L_r = {} & E[\| \varphi_n(a_{n,1}^t, a_{-n,1}^t) - \hat{\varphi}_n(a_{n,1}^t, \bar{a}_n^t) \|^2] \\ & + E[\| \rho_n(a_{n,2}^t, a_{-n,2}^t) - \hat{\rho}_n(a_{n,2}^t, \bar{a}_n^t) \|^2]. \end{aligned} \tag{33}$$

Thus, miner n trains its learning model periodically and interdependently by minimizing the loss function defined in equations (32) and (33).

5.3 Algorithm Analysis

In this subsection, we provide theoretical analyses for the designed LOBE algorithm. Since miners react to the system based on the mean action of others, there should be a Nash equilibrium among policies of miners. In traditional learning-based systems, a Nash equilibrium can be achieved when every player has no individual benefits even changing its policy [31], i.e., $v_{\pi_i^*, \pi_{-i}^*}(s^t) \geq v_{\pi_i, \pi_{-i}^*}(s^t)$. In our work, we utilize mean action $\bar{a}_n^t$ to approximate the aggregation among actions of others, and the learning model is trained with action $\bar{a}_n^t$ based on mean field theory, due to the unawareness of others' policies and the changeable action set of miners. In this case, when a Nash equilibrium is reached among miners, we call it Mean Field (MF) Nash equilibrium [39], which is defined as follows:

Definition 1. *(MF Nash equilibrium) In our considered system, given Markov strategy $\pi^* = \langle \pi_1^*, \pi_2^*, ..., \pi_N^* \rangle$, if variable $\epsilon > 0$ exists to guarantee $v_{\pi_n^*, \pi_{-n}^*}(o_n^t, \bar{a}_n^t) \geq v_{\pi_n, \pi_{-n}^*}(o_n^t, \bar{a}_n^t) - \epsilon$, where $n \in \mathbb{N}$ and $o_n^t \in O$, the formulated stochastic game is an MF ϵ-Nash equilibrium. When $\epsilon = 0$, an MF Nash equilibrium holds.*

Based on Definition 1, we can prove that there exists an MF Nash equilibrium among miner policies in our system, which is presented in Theorem 6.

Theorem 6 (Existence of the MF Nash equilibrium)**.** *There exists an MF Nash equilibrium with the formulated optimization problem defined in Problem P5 for miners, and the Nash point is the fixed point of mapping*

$$\begin{bmatrix} \arg\max v(o_1^t, (\mathbf{\Lambda}_1^\top \otimes I_N)(\cdot)) \\ \vdots \\ \arg\max v(o_N^t, (\mathbf{\Lambda}_N^\top \otimes I_N)(\cdot)) \end{bmatrix}.$$

The proof can be found in Appendix G of Supplemental File.

In addition, we analyze the overall complexity of the designed LOBE algorithm for each agent in Theorem 7:

Theorem 7. *In the execution process of the designed LOBE algorithm, the overall computational complexity for each miner is $\mathcal{O}\left(\left(\sum_{z^\mu=1}^{Z^\mu} n_{z^\mu} \cdot n_{z^\mu-1} + \sum_{z^\eta=1}^{Z^\eta} n_{z^\eta} \cdot n_{z^\eta-1}\right) T\right)$, where Z^μ is the total number of layers for the offloading module and n_{z^μ} is the number of neurons in the z^μ-th layer of the offloading module. Similarly, Z^η and $n_{z^\eta-1}$ are the corresponding parameters for the block size module, respectively.*

The proof can be found in Appendix H of Supplemental File.

6 Performance Evaluation

6.1 Simulation Setup

We conduct the experiment based on the platform of Tensorflow 2.2 and Python 3.7, with the purpose of demonstrating the feasibility of LOBE. For performance validation, the city map of Manhattan as illustrated in Fig. 4 is utilized. An area with 3 km $\times$ 3 km squared by red lines is selected, where fixed edge servers with different SPs are uniformly distributed in the area. Generally, mobile clients (miners) represent walkers or riders holding mobile terminals [31].

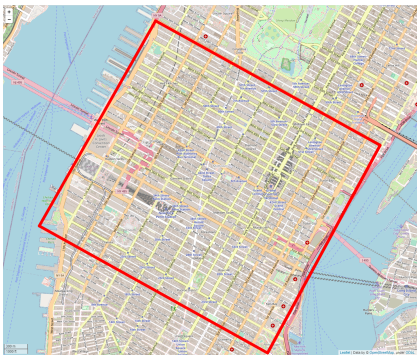


Fig. 4: The city map of Manhattan.

The movements of miners in the considered area follow Manhattan mobility model [40]. The simulated time horizon follows the mining stage because of the high repeatability of miners' activities. In our experiment, the miner with the minimum sum of completion and propagation delays wins the game and obtains the reward. Then, the mining stage moves on to the next one. Simulated parameters are presented in Table 2, and they are the settings in [41] and [42].

In addition, multi-layer perceptions are leveraged for the learning model, and four fully connected layers are constructed for the policy network, offloading and block size regulation modules, as well as the value network, respectively. The designed LOBE algorithm is compared with the following four schemes:

1) Approx RL [7]: Stackelberg game is applied to allocate edge resources for miners, by considering both the budget of miners and the profit of the unique SP. We apply it in our system, and consider that when the miner has more than one choice for offloading, it randomly selects one. For the block size, miners select one that can maximize their rewards from the candidate group of block sizes based on their local observations.

2) An auction mechanism for Multi-Demand miners in Blockchain networks (MDB auction) [43]: It is a resource allocation scheme for the blockchain network, where miners bid for edge computing resources managed by one SP. It is applied in our system similar with Approx RL algorithm.

3) Random Computation Offloading (RCO): Miners randomly select available edge servers for task offloading. If they are not in the wireless communication coverage of any edge servers, miners choose local computing.

4) Advanced Alternating Direction Method of Multipliers (ADMM)-based algorithm [11]: A multi-leader multi-follower game-theoretic approach to support PoW mining by miners, where different SPs and prices are considered. Miner utilities are optimized in a distributed manner, and the full network state is observable to all miners, which is different from ours.

TABLE 2: Simulation Parameters

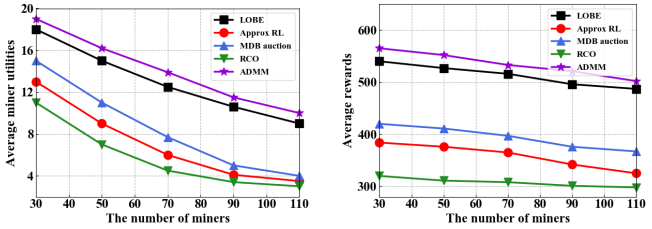
Parameter description	Value
The number of SPs	$[2, 5]$
The number of edge servers	$[4, 15]$
The number of mobile clients	$[30, 110]$
Computing units of each edge server	$[10, 90]$ ECUs
Prices for computing units of each server	$[0.05, 0.3]$ token/ECU·s
Required computing units of the PoW puzzle	$[200, 360]$ ECUs
Computing units of each miner	$[5, 10]$ ECUs
Fixed positive gain	300 tokens
Transmission power of edge servers	3 W
Transmission power of miners	0.1 W
The learning rate for training models	0.01
The batch size	50

6.2 Simulation Results

6.2.1 *Performance based on Different Numbers of Miners*

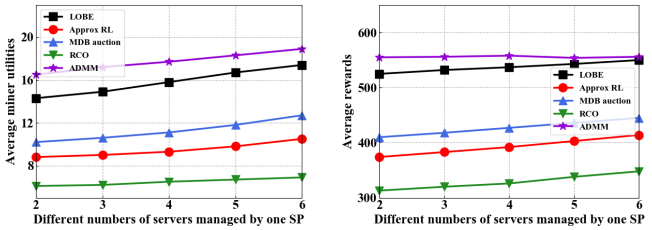
The performance of average miner utilities for Approx RL, MDB auction, RCO, ADMM-based algorithm and the designed algorithm, LOBE, with different numbers of miners is illustrated in Fig. 5(a). It is shown that average miner utility of LOBE is higher than that of Approx RL, MDB auction, and RCO algorithms, while slightly lower than that of ADMM-based algorithm. This is because our algorithm aims at maximizing the long-term expected user utility by designing a DRL-based algorithm, where a Nash equilibrium can be achieved. Although interactions between servers and miners are modeled by a Stackelberg game, Approx RL algorithm tries to maximize long-term profits of servers. It also does not consider different offloading choices for miners, and is not suitable for the mobile environment with multiple SPs. As a result, its corresponding performance is worse than that of LOBE. MDB auction attempts to maximize the social welfare by considering server costs and miner rewards simultaneously. However, it can not make miners always select the best offloading choice similar with Approx RL algorithm. RCO algorithm randomly decides whether to offload tasks, and where to offload. Thus, its performance can not be guaranteed. ADMM-based algorithm aims to maximize miner utilities based on the full knowledge of network states. However, miners make decisions merely based on their local observations by reacting to the average effect from neighbors in the designed algorithm, LOBE. Thus, the performance of ADMM-based algorithm is better than ours.

Fig. 5(b) illustrates the performance of average rewards based on different numbers of miners. The performance on average rewards of LOBE is higher than that of Approx RL, MDB auction, and RCO algorithms, while lower than that of ADMM-based algorithm. This is because LOBE algorithm makes suitable offloading decisions by selecting block sizes, intending to maximize miner utilities. The miner utility is the difference between its reward and cost. Whereas, Approx RL and MDB auction select the block size based on miners' local observations without the awareness of others' actions, resulting in low rewards. RCO algorithm randomly selects the block size and leads to poor performance. ADMM-based algorithm makes offloading decisions based on the full network knowledge, and always computes the optimal computing service demand by iterations of bargain. Our designed algorithm can largely approach to the optimal resource allocation based on partial network observations. When the number of miners becomes big, the trend of average rewards drops. This is because when there are more miners to compete for computing resources in the system for Approx RL, MDB auction, RCO and LOBE algorithms, larger computing and traveling delays can be



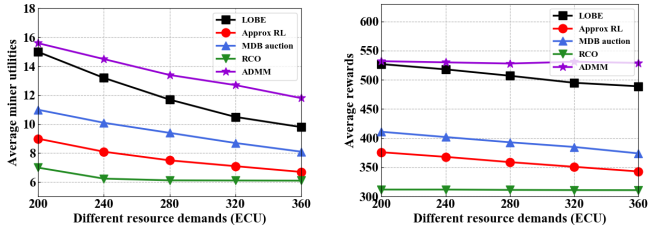
(a) Average miner utilities (b) Average rewards

Fig. 5: Performance with different numbers of miners.



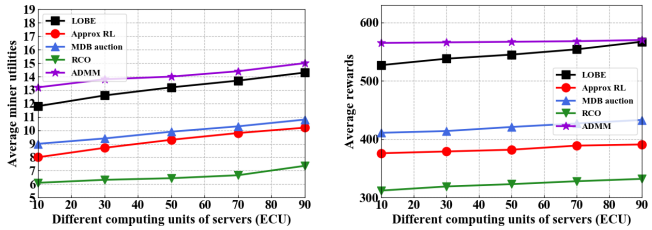
(a) Average miner utilities (b) Average rewards

Fig. 6: Performance with different numbers of edge servers managed by one SP.



(a) Average miner utilities (b) Average rewards

Fig. 7: Performance with different resource demands (ECU).



(a) Average miner utilities (b) Average rewards

Fig. 8: Performance with different computing units of servers.

caused. Then, miners can adjust their block sizes to shorten their propagation delays with the purpose of winning the game. Thus, shorter block sizes result in lower rewards. Although the block size of ADMM algorithm follows a normal distribution and the reward is proportional to the block size, the average reward becomes small when the number of miners becomes large.

6.2.2 Performance based on Different Numbers of Servers

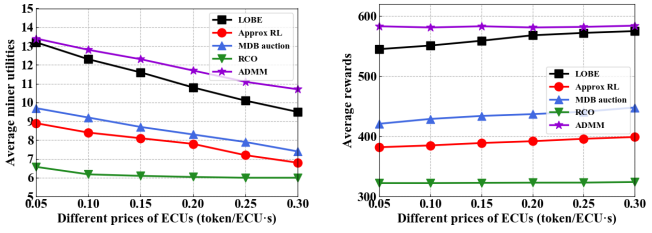
Fig. 6(a) shows average miner utilities based on different numbers of edge servers managed by each SP. It is obvious that the performance of the five algorithms becomes better when the number of edge servers managed by each SP becomes bigger. This is because more edge servers lead to more computation resources, which can be utilized for task processing. Then, processing and traveling delays can be correspondingly reduced, which lowers miner costs. In addition, the performance of LOBE algorithm is better than that of Approx RL, MDB auction, and RCO algorithms, and worse than that of ADMM-based algorithm. The reason is that LOBE algorithm can always make suitable offloading decisions for miners based on the mean action of others. However, the purpose of Approx RL is to maximize the profits of edge servers, and that of MDB auction is to maximize the social welfare of the system. Thus, they perform worse than ours. In addition, miners in ADMM-based algorithm select edge servers for task offloading based on the full knowledge of others' service demands, and can make more accurate decisions compared with the designed algorithm.

Fig. 6(b) illustrates the performance of average rewards based on different numbers of edge servers. Average rewards of LOBE, Approx RL, MDB auction, and RCO algorithms increase, when the number of edge servers becomes big. This is because a bigger number of edge servers leads to more computing resources. Then, transmission and processing delays can be reduced, and miners can slightly increase the block size to win more rewards. The average reward of ADMM-based algorithm keeps stable when the number of edge servers increases. This is because the block size in ADMM-based algorithm follows the normal distribution and its average value is fixed, leading to the fixed value of average rewards when the number of miners is not changed.

6.2.3 The Impact of Resource Demands

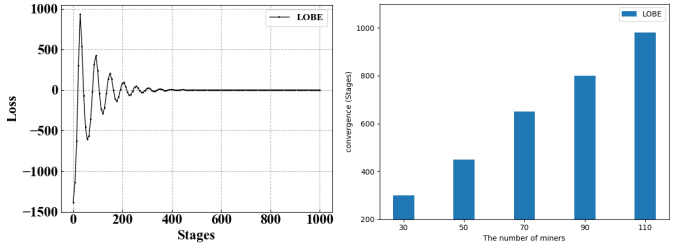
The impact of resource demands on the system performance is illustrated in Fig. 7. Resource demands refer to the required ECUs for solving the PoW puzzle. The trend of average miner utilities is illustrated in Fig. 7(a), and it decreases when the number of resource demands becomes big. For example, when the resource demand is 240 ECUs, average miner utilities of LOBE, Approx RL, MDB auction, RCO and ADMM-based algorithms are 13.2, 10.1, 6.24, 7.1, 6.7 and 14.5, respectively. When the resource demand is 320 ECUs, average miner utilities of the five algorithms are 10.5, 7.1, 8.7, 6.11, and 12.7, respectively. The reason is that, more computing resources are required when the number of resource demands becomes big, and costs of miners increase correspondingly. Then, the trend of average miner utilities drops.

Fig. 7(b) is the trend of average rewards with different resource demands. When the number of resource demands becomes big, the trend of average rewards drops for LOBE, Approx RL, MDB auction, and RCO algorithms. For example, when the resource demand is 240, their average rewards are 518, 368, 402, and 312, respectively. When the resource demand is 320, average rewards of the four algorithms are 495, 351, 385 and 311.1, respectively. This is because when the value of the resource demand becomes large, more computing and traveling delays can be caused. If a miner intends to win, it may reduce its propagation delay



(a) Average miner utilities (b) Average rewards

Fig. 9: Performance with different prices of ECUs (token/ECU·s).



(a) Loss (b) Convergence time

Fig. 10: Performance of convergence.

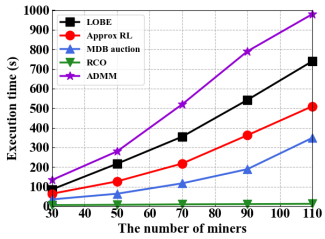


Fig. 11: Execution time.

to increase its success probability, which inversely degrades its reward. In addition, the average reward of ADMM-based algorithm almost has no change with different resource demands, since its reward is merely affected by the block size. If the average block size has no change, the average reward keeps stable.

6.2.4 The Impact of Computing Units of Servers

System performance with different computing units of servers is illustrated in Fig. 8. In this case, we consider that all servers even managed by different SPs have the same computing unit in each mining stage. We can observe that, the designed algorithm, LOBE, performs better compared with Approx RL, MDB auction, and RCO algorithms, while worse than ADMM-based algorithm. This is because LOBE intends to maximize the long-term expected miner utility by designing a reasonable computation offloading strategy based on local observations of miners. ADMM-based algorithm intends to maximize miner utilities based on the full knowledge of network states. However, Approx RL algorithm aims to maximize profits of servers while satisfying miners' budgets, and miner utilities can not be maximized. MDB auction algorithm aims to maximize the social welfare based on an approximation algorithm, and rewards of miners as well as the cost of edge servers are considered.

The performance trend of average rewards for the five algorithms is illustrated in Fig. 8(b). When the number of computing units for each edge server increases, average rewards of LOBE, Approx RL, MDB auction, and RCO algorithms become large. For instance, when the computing units of each server are 30, their average rewards are 12.6, 8.7, 9.4, and 6.32, respectively. When the computing units of each server increase to 70, average rewards of the four algorithms are 13.7, 9.8, 10.3 and 6.66, respectively. The reason is that more computing resources in the system can lead to higher task processing rates, which makes miners increase their block sizes and tolerate long propagation delays. However, the block size of all miners in each mining stage is the same in ADMM-based algorithm, and follows a normal distribution with a fixed average value. Thus, the average reward of ADMM-based algorithm has no change.

6.2.5 The Impact of ECU Prices

Fig. 9 shows the impact of different ECU prices on the system performance. The ECU price refers to the price for the usage of unit ECU in each second, and we consider that ECU prices for all SPs are the same to evaluate the impacts of prices. From Fig. 9(a), when the price of ECUs becomes high, the values of average miner utilities for LOBE, Approx RL, MDB auction, RCO and ADMM-based algorithms become small. For example, when the price of ECUs is 0.1 token/ECU·s, average miner utilities are 12.3, 8.4, 9.2, 12.8 and 6.18, respectively. When the price is 0.25 token/ECU·s, those are 10.1, 7.2, 7.9, 11.1 and 6.04, respectively. This is because when the price becomes high, the cost for employing ECUs to process tasks increases, resulting in the decreasing trend of average user utilities.

Fig. 9(b) is the performance of average rewards for the five algorithms. Different from Fig. 9(a), the trend of average rewards rises when the price of ECUs becomes high for LOBE, Approx RL, MDB auction, and RCO algorithms. This is because miners need to increase rewards by enlarging the block size against the increasing cost, and keep their utilities be positive in LOBE algorithm. Otherwise, they may select local computing. Approx RL algorithm intends to maximize profits of servers, and miner rewards can not be maximized. MDB auction establishes a positive relationship between miner rewards and the social welfare function, and maximizing the social welfare can increase the reward to some extent. Thus, the performance of MDB auction algorithm is better than that of Approx RL algorithm. The block size of ADMM algorithm follows a normal distribution and the reward is proportional to the block size. Thus, the average reward keeps fixed when the price of ECUs increases.

6.2.6 The Performance of Convergence

We randomly select one miner, and draw the trend of losses for the policy network with training stages in Fig. 10(a). It is observed that at the beginning of training stages, the loss of the policy network has a big vibration. When the number of stages increases, the amplitudes of the loss trend gradually become small. After 400 stages, the loss fluctuates around 0, i.e., the learning algorithm converges. This is because the designed algorithm, LOBE, can learn good offloading and

block size decisions based on the mean actions of neighbors, and gradually converge to the Nash equilibrium point. Fig. 10(b) is the convergence time of the designed algorithm with different numbers of miners. Since merely our algorithm depends on the neural network training, we evaluate its convergence time, which is acceptable in reality. It is obvious that the convergence time becomes long when the number of miners increases. The reason is that more miners with more offloading tasks are involved in the game, and more iterations are needed to make them converge to the Nash equilibrium point.

6.2.7 The Performance of Execution Time

The algorithm execution time of the five algorithms is shown in Fig. 11 with 5000 simulated mining stages. We can observe that the execution time of LOBE is longer than that of Approx RL, MDB auction and RCO algorithms, while shorter than that of ADMM-based algorithm. This is because LOBE adopts neural networks with two modules (offloading and block size modules) to generate actions, which is more complicated and consumes more time than that of Approx RL and MDB auction algorithms. RCO algorithm randomly selects an available choice for offloading and block size decisions, the execution time of which is much lower due to its simple operations. For ADMM-based algorithm, edge servers and miners need to bargain with multiple rounds to reach a consensus among them in each mining stage, which consumes more time than the designed algorithm, LOBE. When the number of miners increases, the execution time of the five algorithms becomes long. This is because more miners are involved into the execution process for resource scheduling and decision making, increasing the complexity of the whole algorithm.

7 Conclusion

In this paper, we proposed a mean field learning-based computation offloading algorithm for mobile blockchain with multiple SPs. Our objective is to maximize long-term expected utilities for users. First, we formulated the Markov game by considering block sizes and task completion delays of miners. To make it solvable, we formed the short-term optimization problem, which is demonstrated to be consistent with the long-term one. Then, offloading decisions and block sizes for each miner can be learned independently by the designed neural network model, where opponent actions can be approximated by mean actions of neighbors based on mean field theory. Finally, both theoretical and performance results showed that the designed algorithm performs better on average miner utilities and average rewards compared with other representative algorithms.

Acknowledgments

This work was supported by the Natural Science Foundation of China under Grants 61971084, 62025105, and 62001073, by the National Natural Science Foundation of Chongqing under Grants cstc2019jcyj-cxttX0002, cstc2021ycjh-bgzxm0013, cstc2021ycjh-bgzxm0039 and cstc2021jcyj-msxmX0031, by the Chongqing Municipal Education Commission (key cooperation project HZ2021008 and CXQT21019), by the Support Program for Overseas Students to Return to China for Entrepreneurship and Innovation under Grants cx2021003 and cx2021053, by the Young Talent fund of China Institute of Communications under Grants YESS20200214.

References

[1] Z. Ning, S. Sun, X. Wang, L. Guo, S. Guo, X. Hu, B. Hu, and R. Kwok, "Blockchain-enabled intelligent transportation systems: A distributed crowdsensing framework," *IEEE Transactions on Mobile Computing, DOI:10.1109/TMC.2021.3079984*, 2021.

[2] M. S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli, and M. H. Rehmani, "Applications of blockchains in the Internet of things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1676–1717, 2018.

[3] Z. Ning, H. Chen, X. Wang, S. Wang, and L. Guo, "Blockchain-enabled electrical fault inspection and secure transmission in 5G smart grids," *IEEE Journal of Selected Topics in Signal Processing, DOI:10.1109/JSTSP.2021.3120872*, 2021.

[4] W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, and I. K. Dong, "A survey on consensus mechanisms and mining strategy management in blockchain networks," *IEEE Access*, vol. 7, pp. 122328–22370, 2019.

[5] Z. Yu, X. Liu, and G. Wang, "A survey of consensus and incentive mechanism in blockchain derived from P2P," in *Proc. IEEE ICPADS*, pp. 1010–1015, 2018.

[6] Z. Ning, S. Sun, X. Wang, L. Guo, G. Wang, X. Gao, and R. Y. Kwok, "Intelligent resource allocation in mobile blockchain for privacy and security transactions: A deep reinforcement learning based approach," *Science China Information Sciences*, vol. 64, no. 6, pp. 1–16, 2021.

[7] 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, 2019.

[8] R. Nambiar, "A standard for benchmarking big data systems," in *IEEE International Conference on Big Data*, pp. 18–20, 2014.

[9] L. Huang, X. Feng, L. Zhang, L. Qian, and Y. Wu, "Multi-server multi-user multi-task computation offloading for mobile edge computing networks," *Sensors*, vol. 19, no. 6, p. 1446, 2019.

[10] P. A. Apostolopoulos, E. E. Tsiropoulou, and S. Papavassiliou, "Risk-aware data offloading in multi-server multi-access edge computing environment," *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1405–1418, 2020.

[11] Z. Xiong, J. Kang, D. Niyato, P. Wang, and H. V. Poor, "Cloud/edge computing service management in blockchain networks: Multi-leader multi-follower game-based ADMM for pricing," *IEEE Transactions on Services Computing*, vol. 13, no. 2, pp. 356–367, 2020.

[12] M. Liu, F. R. Yu, Y. Teng, V. C. Leung, and M. Song, "Distributed resource allocation in blockchain-based video streaming systems with mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 18, no. 1, pp. 695–708, 2018.

[13] N. C. Luong, Z. Xiong, P. Wang, and D. Niyato, "Optimal auction for edge computing resource management in mobile blockchain networks: A deep learning approach," in *Proc. IEEE ICC*, pp. 1–6, 2018.

[14] Y. Jiao, P. Wang, D. Niyato, and Z. Xiong, "Social welfare maximization auction in edge computing resource allocation for mobile blockchain," in *Proc. IEEE ICC*, pp. 1–6, 2018.

[15] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "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, 2020.
[16] Z. Xiong, S. Feng, W. Wang, D. Niyato, P. Wang, and Z. Han, "Cloud/fog computing resource management and pricing for blockchain networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4585–4600, 2018.
[17] Y. Wu, X. Xu, L. Qian, B. Ji, Z. Shi, and W. Jia, "Revenue-sharing based computation-resource allocation for mobile blockchain," in *Proc. INFOCOM WKSHPS*, pp. 56–61, 2020.
[18] Y. Dai, D. Xu, K. Zhang, S. Maharjan, and Y. Zhang, "Deep reinforcement learning and permissioned blockchain for content caching in vehicular edge computing and networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4312–4324, 2020.
[19] R. Zhang, F. R. Yu, J. Liu, R. Xie, and T. Huang, "Blockchain-incentivized D2D and mobile edge caching: A deep reinforcement learning approach," *IEEE Network*, vol. 34, no. 4, pp. 150–157, 2020.
[20] J. Feng, F. R. Yu, Q. Pei, X. Chu, J. Du, and L. Zhu, "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, 2019.
[21] Z. Su, Y. Wang, Q. Xu, and N. Zhang, "LVBS: Lightweight vehicular blockchain for secure data sharing in disaster rescue," *IEEE Transactions on Dependable and Secure Computing, DoI:10.1109/TDSC.2020.2980255*, vol. 19, no. 1, pp. 19–32, 2022.
[22] L. Xiao, Y. Ding, D. Jiang, J. Huang, D. Wang, J. Li, and H. V. Poor, "A reinforcement learning and blockchain-based trust mechanism for edge networks," *IEEE Transactions on Communications*, vol. 68, no. 9, pp. 5460–5470, 2020.
[23] J. Du, W. Cheng, G. Lu, H. Cao, X. Chu, Z. Zhang, and J. Wang, "Resource pricing and allocation in MEC enabled blockchain systems: An A3C deep reinforcement learning approach," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 1, pp. 33–44, 2022.
[24] N. Houy, "The bitcoin mining game," *Ledger*, vol. 1, pp. 53–68, 2016.
[25] R. Zhang, R. Xue, and L. Liu, "Security and privacy on blockchain," *ACM Computing Surveys*, vol. 52, no. 3, pp. 1–34, 2019.
[26] G. Yang, X. Shi, L. Feng, S. He, Z. Shi, and J. Chen, "Cedar: A cost-effective crowdsensing system for detecting and localizing drones," *IEEE Transactions on Mobile Computing*, vol. 19, no. 9, pp. 2028–2043, 2019.
[27] D. Liu, P. Ning, and W. Du, "Detecting malicious beacon nodes for secure location discovery in wireless sensor networks," in *Proc. IEEE ICDCS*, pp. 609–619, 2005.
[28] M. U. Thomas, "Queueing systems. volume 1: Theory (leonard kleinrock)," *SIAM Review*, vol. 18, no. 3, pp. 512–514, 1976.
[29] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *IEEE P2P Proceedings*, pp. 1–10, 2013.
[30] P. R. Rizun, "A transaction fee market exists without a block size limit," *Block Size Limit Debate Working Paper*, pp. 2327–4697, 2015.
[31] X. Wang, Z. Ning, and S. Guo, "Multi-agent imitation learning for pervasive edge computing: A decentralized computation offloading algorithm," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 2, pp. 411–425, 2020.
[32] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," in *Proc. ICML*, pp. 5571–5580, 2018.
[33] M. Li, Z. Qin, Y. Jiao, Y. Yang, J. Wang, C. Wang, G. Wu, and J. Ye, "Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning," in *The World Wide Web Conference*, pp. 983–994, 2019.
[34] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[35] C. Domb, *Phase transitions and critical phenomena*. Elsevier, 2000.
[36] J. Buckman, A. Roy, C. Raffel, and I. Goodfellow, "Thermometer encoding: One hot way to resist adversarial examples," in *International Conference on Learning Representations*, pp. 1–22, 2018.
[37] R. Lowe, Y. WU, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in Neural Information Processing Systems*, vol. 30, pp. 6379–6390, 2017.
[38] X. Wang and D. Klabjan, "Competitive multi-agent inverse reinforcement learning with sub-optimal demonstrations," in *International Conference on Machine Learning*, pp. 5143–5151, 2018.
[39] S. Grammatico, F. Parise, M. Colombino, and J. Lygeros, "Decentralized convergence to Nash equilibria in constrained deterministic mean field control," *IEEE Transactions on Automatic Control*, vol. 61, no. 11, pp. 3315–3329, 2015.
[40] G. Walunjkar and A. K. Rao, "Simulation and evaluation of different mobility models in disaster scenarios," in *Proc. IEEE RTEICT*, pp. 464–469, 2019.
[41] T. Zhu, J. Li, Z. Cai, Y. Li, and H. Gao, "Computation scheduling for wireless powered mobile edge computing networks," in *Proc. IEEE INFOCOM*, pp. 596–605, 2020.
[42] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 1784–1797, 2017.
[43] Y. Jiao, P. Wang, D. Niyato, and K. Suankaewmanee, "Auction mechanisms in cloud/fog computing resource allocation for public blockchain networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 9, pp. 1975–1989, 2019.

Xiaojie Wang received the PhD degree from Dalian University of Technology, Dalian, China, in 2019. After that, she was a postdoctor in the Hong Kong Polytechnic University. Currently, she is a full professor with the College of Communication and Information Engineering, the Chongqing University of Posts and Telecommunications, Chongqing, China. Her research interests are wireless networks, mobile edge computing and machine learning. She has published over 50 scientific papers in international journals and conferences, such as IEEE TMC, IEEE JSAC, IEEE TPDS and IEEE COMST.

Zhaolong Ning (SM'18) received the Ph. D. degree from Northeastern University, China in 2014. He was a Research Fellow at Kyushu University from 2013 to 2014, Japan. Currently, he is a full professor with the College of Communication and Information Engineering, the Chongqing University of Posts and Telecommunications, Chongqing, China. His research interests include Internet of things, mobile edge computing, deep learning, and resource management. He has published over 120 scientific papers in international journals and conferences. Dr. Ning serves as an associate editor or guest editor of several journals, such as IEEE Transactions on Industrial Informatics, IEEE Transactions on Social Computational Systems, The Computer Journal and so on.

Lei Guo received the Ph.D. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2006. He is currently a full professor with Chongqing University of Posts and Telecommunications, Chongqing, China. He has authored or coauthored more than 200 technical papers in international journals and conferences. He is an editor for several international journals. His current research interests include communication networks, optical communications, and wireless communications.

Song Guo (M'02-SM'11-F'20) is a Full Professor in the Department of Computing at The Hong Kong Polytechnic University. He also holds a Changjiang Chair Professorship awarded by the Ministry of Education of China. His research interests are mainly in the areas of big data, edge AI, mobile computing, and distributed systems. He co-authored 4 books, co-edited 7 books, and published over 500 papers in major journals and conferences. He is the recipient of over 12 Best Paper Awards from IEEE/ACM conferences, journals and technical committees. His work was also recognized by the 2016 Annual Best of Computing: Notable Books and Articles in Computing in ACM Computing Reviews. Prof. Guo's research has been sponsored by RGC, NSFC, MOST, industry, etc. He is the Editor-in-Chief of IEEE Open Journal of the Computer Society and the Chair of IEEE Communications Society (ComSoc) Space and Satellite Communications Technical Committee. He was an IEEE ComSoc Distinguished Lecturer and a member of IEEE ComSoc Board of Governors. He has also served for IEEE Computer Society on Fellow Evaluation Committee, Transactions Operations Committee, Editor-in-Chief Search Committee, etc. Prof. Guo has been named on editorial board of a number of prestigious international journals like IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Cloud Computing, IEEE Internet of Things Journal, etc. He has also served as chairs of organizing and technical committees of many international conferences. Prof. Guo is an IEEE Fellow, a Highly Cited Researcher (Web of Science), and an ACM Distinguished Member.

Xinbo Gao received the B.Eng., M.Sc., and Ph.D. degrees in electronic engineering, signal and information processing from Xidian University, Xi'an, China, in 1994, 1997, and 1999, respectively. From 1997 to 1998, he was a Research Fellow with the Department of Computer Science, Shizuoka University, Shizuoka, Japan. From 2000 to 2001, he was a Postdoctoral Research Fellow with the Department of Information Engineering, Chinese University of Hong Kong, Hong Kong. Since 2001, he has been with the School of Electronic Engineering, Xidian University, where he is currently a Cheung Kong Professor of Ministry of Education, and a Professor of Pattern Recognition and Intelligent System, and a Professor of Computer Science and Technology with the Chongqing University of Posts and Telecommunications, Chongqing, China. He has published six books and around 300 technical articles in refereed journals and proceedings. His current research interests include Image processing, computer vision, multimedia analysis, machine learning, and pattern recognition.

Prof. Gao is on the editorial boards of several journals, including *Signal Processing* (Elsevier) and *Neurocomputing* (Elsevier). He served as the General Chair/Co-Chair, the Program Committee Chair/Co-Chair, or a PC Member for around 30 major international conferences. He is a Fellow of the Institute of Engineering and Technology and the Chinese Institute of Electronics.

Guoyin Wang (SM'03) received the B.S., M.S., and Ph.D. degrees in computer science and technology from Xi'an Jiaotong University, Xi'an, China, in 1992, 1994, and 1996, respectively. During 1998-1999, he was a Visiting Scholar with the University of North Texas, Denton, TX, USA, and the University of Regina, Regina, SK, Canada. Since 1996, he has been with the Chongqing University of Posts and Telecommunications, Chongqing, China, where he is currently a Professor, a Vice-President of the university, and the Director with the Chongqing Key Laboratory of Computational Intelligence. He was appointed as the Director of the Institute of Electronic Information Technology, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, in 2011. He is the author of 13 books and has more than 300 reviewed research publications. His current research interests include rough set, granular computing, knowledge technology, data mining, neural network, and cognitive computing. Dr. Wang is the Editor of dozens of proceedings of international and national conferences.

Dr. Wang was the President of International Rough Set Society (IRSS) from 2014 to 2017. He is currently a Vice-President of the Chinese Association for Artificial Intelligence (CAAI) and a Council Member of the China Computer Federation (CCF).