Learning-Based Mobile Edge Computing Resource Management to Support Public Blockchain Networks

Alia Asheralieva and Dusit Niyato, *Fellow, IEEE*

Abstract—We consider a public blockchain realized in the mobile edge computing (MEC) network, where the blockchain miners compete against each other to solve the proof-of-work puzzle and win a mining reward. Due to limited computing capabilities of their mobile terminals, miners offload computations to the MEC servers. The MEC servers are maintained by the service provider (SP) that sells its computing resources to the miners. The SP aims at maximizing its long-term profit subject to miners' budget constraints. The miners decide on their hash rates, i.e., computing powers, simultaneously and independently, to maximize their payoffs without revealing their decisions to other miners. As such, the interactions between the SP and miners are modeled as a stochastic Stackelberg game under private information, where the SP assigns the price per unit hash rate, and miners select their actions, i.e., hash rate decisions, without observing actions of other miners. We develop a hierarchical learning framework for this game based on fully- and partially-observable Markov decision models of the decision processes of the SP and miners. We show that the proposed learning algorithms converge to stable states in which miners' actions are the best responses to the optimal price assigned by the SP.

Index Terms—Blockchain, deep learning, game theory, incomplete information, Markov decision process, mining, mobile edge computing, partially-observable Markov decision process, reinforcement learning, resource management

1 Introduction

Developed originally as an accounting method for the cryptocurrency Bitcoin [1], blockchains have been adopted in numerous applications, e.g., FinTech, and as a backbone of the distributed open-access virtual machines (VMs) for decentralized token-driven resource management in autonomous communication networks and cyber-physical systems [2], [3], [4]. Blockchain is a distributed ledger technology (DLT) that enables to store data with in-build robustness, since it cannot be controlled by a single entity and able to avoid a single point of failure. Public blockchains are also characterized by their transparency, disintermediation and open access. In a blockchain, data is collected in the form of blocks, e.g., records of transactions which form a linked list data structure to preserve logical relations in the appended information. The blocks are copied and distributed across the entire blockchain network comprising all blockchain users. This provides improved data integrity and security when compared to the centralized ledger approaches [2], [3]. The process by which the transactions are confirmed and added to the blockchain is referred to as a mining [1]. During mining, a blockchain user, i.e., a miner, must solve a computationally intensive proof-of-work (PoW) puzzle and broadcast its solution to other users. The transaction is verified, and a new block is added to the current blockchain only if the respective solution reaches a consensus. Most consensus protocols are based on Nakamoto consensus [4], [5] that provides a certain reward to the first miner successfully solving the PoW puzzle. The probability of winning a reward is proportional to the miner's relative hash rate, i.e., computing power [1], [2], [3], [4], [5]. This leads to the competition among miners and incentivizes the mining process.

Unfortunately, although blockchains are already utilized in several distributed system scenarios, e.g., content delivery networks [6] and smart-grids [7], they are not yet widely adopted in the Internet of Things (IoT) and other mobile systems. The main reason is that the IoT devices and users' mobile terminals have limited computing capabilities. Hence, they cannot run blockchain applications directly and independently. To address this issue, edge computing has been proposed as a promising solution to facilitate future IoT and mobile blockchain applications [8], [9]. In MEC networks, virtual cloud computing capabilities are extended at the network "edges" through MEC servers attached to base stations (BSs) in proximity to IoT and mobile terminals [10], [11]. As such, the devices running blockchain applications can offload their PoW puzzle computations to nearest BSs equipped with MEC servers. Compared to traditional cloud computing where computations are offloaded to centralized clouds located in the network core (far from end-users) resulting in the high propagation delays and heavy backhaul load, MEC allows to reduce the network congestion and backhaul traffic,

- *A. Asheralieva is with the Department of Computer Science and Engineering, Southern University of Science and Technology, 1088 Xueyuan Avenue, Nanshan District, Shenzhen, Guangdong 518055, China. E-mail: aasheralieva@gmail.com.*
- *D. Niyato is with the School of Computer Science and Engineering, Nanyang Technological University, 50 Nanyang Ave, 639798, Singapore. E-mail: dniyato@ntu.edu.sg.*

Manuscript received 3 Oct. 2018; revised 6 Oct. 2019; accepted 10 Dec. 2019. Date of publication 16 Dec. 2019; date of current version 3 Feb. 2021. (Corresponding author: Alia Asheralieva.)
Digital Object Identifier no. 10.1109/TMC.2019.2959772

and also decreases latency and power consumption of IoT and mobile terminals. All advantages of integrating blockchain with MEC systems are detailed in [12].

In this paper, we consider a public blockchain application implemented in the MEC network, where the set of blockchain users or miners offload their PoW puzzle com-putations to the associated BSs operated by the MEC SP. In such a scenario, the cost of computing power in the MEC network becomes a critical issue. In particular, the SP assigns the price per unit of hash rate to maximize its long-term payoff from the MEC services while taking into account the constraints on miners' budgets, i.e., the highest price that miners can afford to pay during the entire mining process. In response to the assigned price, every miner decides on its hash rate to maximize its long-term payoff defined as a difference between the miner's long-term mining reward and the total cost of computing power payable to the SP.

Two critical issues must be addressed when modeling the interactions between the SP and the miners:

- First, in the blockchain network, miners make their hash rate decisions simultaneously and independently [1], [13]. Besides, as miners compete against each other to earn the reward, they will not reveal their decisions to each other. As such, a hash rate decision of any miner is the private information which is unknown and unobservable by other miners.
- Second, the stochasticity of the mobile environment (e.g., time-varying wireless channel quality and miners' locations) must be taken into account in the model. That is, the SP and miners should be able to dynamically adjust their respective pricing and hash rate decisions in response to random environmental changes.

The main contributions of this paper are as follows:

1) Given unobservable information about miners' actions (i.e., hash rate decisions), we demonstrate that the interactions between the SP and the miners can be described by a stochastic Stackelberg game under private information about the miner's actions. The game is stochastic because the long-term payoffs of the SP (or a leader) and miners (i.e., followers) are affected by the random mobile environment with unknown state transitions.
2) We prove the existence and uniqueness of Nash equilibrium (NE) for the game played by the miners. Since miners cannot observe the actions of other miners, we model their decision making by a partially-observable Markov decision process (POMDP) and show that its solution determines the miners' best response strategies which maximize their expected long-term payoffs.
3) We formulate a stochastic optimization problem of the SP where both the objective (expected long-term payoff of the SP) and the constraints (on miners' budgets) are the functions of a stochastic state determined by the miners' actions and rewards. To solve this problem, we represent the SP's decision making as a Markov decision process (MDP) and show that its solution is equivalent to the solution of the SP's optimization problem.
4) Based on the presented MDP and POMDP models, we propose a hierarchical reinforcement learning (RL) fra-mework for the SPs and miners. The framework allows the SP to maximize its long-term payoff by dynamically adjusting the price per unit hash rate. It also enables the miners to select the best response strategies and update beliefs about the unobservable actions of other miners through repeated interactions with each other and the mobile environment.

The rest of the paper is organized as follows. In Section 2, we review the related work on blockchain applications in wireless networks. The system model of blockchain applications in the MEC system is developed in Section 3. The game modeling the interactions of blockchain miners in the blockchain network, the POMDP model of the miners' decision process, and RL algorithms based on this model are proposed in Section 4. The stochastic optimization problem for the SP, the MDP model of the SP's decision process, RL algorithms based on this model, and a hierarchical learning procedure for the SP and the miners are formulated in Section 5. The numerical results are provided in Section 6. The conclusions are given in Section 7.

2 Related Work

Existing research in the area of blockchain applications in wireless networks can be arbitrarily divided into two categories. The first category ([14], [15], [16], [17], [18], [19], [20], [21]) analyzes the design and implementation of the various blockchain-based protocols for secure and decentralized data communication in wireless networks. For example, in [14], the authors present a distributed mechanism which exploits a secure public key encryption scheme, namely, a hierarchical identity-based encryption (HIBE), for content distribution in information centric networks. The mechanism deploys a Bitcoin-like blockchain network, Namecoin, for data registration and transfer in the system, where the content name is utilized as HIBE public keys. In [15], the authors propose to directly map the account addresses in the blockchain networks to the addresses in the named data networking (NDN) and show that such blockchain system over NDN can overcome some problems, such as eavesdropping and traffic analysis, of Internet protocol (IP) networks. The studies in [16] and [17] explore feasibility of distributed consensus powered smart contracts in the content delivery networks. The proposed content delivery model utilizes a blockchain-based brokering mechanism with a series of smart contracts. It adopts the blockchain network as a backbone for executing and auditing transactions among different network entities to enable self-organization and decentralization. Blockchain-based spectrum sharing protocols for cognitive radio networks are analyzed in [18]. Here, the blockchain network serves as a medium access protocol, by which the competing cognitive radios can lease and access available spectrum resources in a secure and decentralized manner. The key management scheme for secure group communication in vehicular communication systems is presented in [19]. The proposed framework uses a blockchain-based network topology to decentralize and simplify the key transfer transactions which are recorded, shared and copied in blocks to preserve the data integrity. In [20], the authors

develop a security enhancement solution for a blockchain-enabled Internet of Vehicles (BIoV) system that comprises two stages: i) miner selection and ii) block verification. The first stage is a voting scheme where the candidate vehicle is evaluated based on its reputation which depends on the historical data and opinions of other vehicles (the candidate with the high reputation is selected to be an active miner). In the second stage, a newly generated block is verified and audited by standby (i.e., inactive) miners to prevent internal collusions among active miners. The participation of standby miners in the block verification process is incentivized by adopting the contract theory. A comprehensive survey on the state-of-the-art decentralized consensus mechanisms in the blockchain network is provided in [21]. The survey focuses on the perspectives of the design of distributed consensus mechanisms, its impact on the blockchain network architecture, self-organization by individual blockchain nodes, and emerging blockchain applications.

The second category ([8], [9], [22], [23]) investigates the pricing and resource management for supporting blockchain applications in wireless networks. The focus here is on the mining under Nakamoto consensus [4], [5] which results in the competition among miners to receive a mining reward. Due to limited computing resources of their mobile terminals, miners offload their PoW computations to local MEC servers. For example, in [8] and [9], the interplay between the MEC SP and miners is modeled as a two-stage Stackelberg game. At stage one, the SP (aka leader) assigns the price per unit hash rate with the aim to maximize its one-stage profit. At stage two, given the unit price, the miners (followers) select hash rates to maximize their stage payoffs. The auction mechanisms for resource management in the MEC system with blockchain applications are analyzed in [22] and [23]. The objective of the auction in [22] is to maximize the social welfare while ensuring its incentive compatibility (i.e., trustfulness for the miners). In [23], the performance of auction to maximize the revenues of the MEC SPs is improved by applying deep learning. To summarize, the above studies provide the possibility of enhancing and extending the computing abilities of mobile and IoT terminals through MEC services offered by some external SP. Nevertheless, they still have several limitations which are listed below:

- The main assumption of the works in [8], [9], [22], [23] is that the hash rate decision of each miner is known to other miners. This is unrealistic in the blockchain networks, since miners choose their actions simultaneously and independently, without announcing their decisions to other miners [1], [13].
- It is considered that the miners' budgets are unconstrained, i.e., the miners can pay any price for MEC services, but in real-world scenarios, the miners' budgets are finite.
- The dynamics and randomness of the wireless environment (e.g., channel quality and miners' locations) are ignored in the models in [8], [9], [22], [23]. That is, the games and auctions are played in a static and deterministic environment, and their solutions are for a short term (one-stage).

Similar to above works, this paper studies pricing and computational offloading for public blockchain mining under the Nakamoto consensus. However, unlike these works, it is based on more realistic assumptions that are applicable to the practical MEC networks. In particularly, we assume that:

1) The information about the miners' hash rate decisions is private, i.e., unknown by other miners. Hence, the miners must form and update their beliefs about the actions of other miners through RL.
2) The miners' budgets are limited. That is, when deciding on the price, the SP must also consider possible expenditures of miners, which makes the problem much more complicated.
3) The proposed Stackelberg model is stochastic, i.e., it is played in a dynamic and random environment, and re-turns the long-term (multi-stage) solutions.

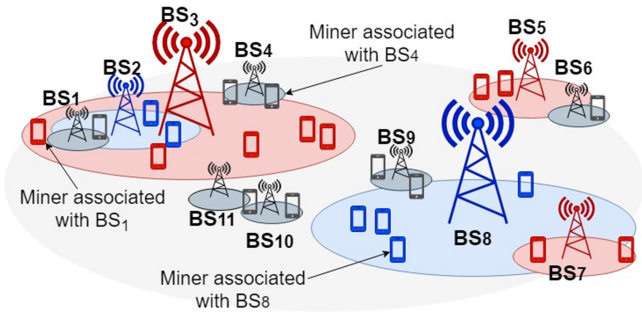

Fig. 1. Blockchain application in the MEC network with $M = 11$ BSs.

3 Blockchain Over the MEC Network

3.1 System Model

Consider a public blockchain application realized over a wireless MEC network, e.g., as in [8], [9]. As shown in Fig. 1, the MEC network is formed by the set $\mathbf{M} = \{1, \ldots, M\}$ of macro- and small-cell BSs, labeled as $\mathrm{BS}_1, \ldots, \mathrm{BS}_M$, each of which is equipped with the MEC server. In the network, the set $\mathbf{N} = \{1, \ldots, N\}$ of users or miners, labelled as $\mathrm{U}_1, \ldots, \mathrm{U}_N$, offload the PoW puzzle computations to MEC servers of their associated BSs. We denote by $\mathbf{N}_m \subseteq \mathbf{N}$, $m \in \mathbf{M}$, the subset of miners associated with BS_m, such that $\bigcup_{m\in\mathbf{M}} \mathbf{N}_m = \mathbf{N}$ and $\bigcap_{m\in\mathbf{M}} \mathbf{N}_m = \emptyset$. Note that since the locations of miners in the MEC network may change over time, the corresponding miner-BS associations and, hence, the subsets $\mathbf{N}_m$, $m \in \mathbf{M}$, may also change. The network operates on the orthogonal frequency-division multiplexing (OFDM) spectrum. All BSs are maintained by the SP that sells its computing resources counted in terms of edge computing units (ECUs) to miners. The computing resources of each BS are limited by the maximal number of ECUs supported by its MEC server. We denote by $\mathcal{A}_m \in \mathbb{N}_+$, $m \in \mathbf{M}$, where $\mathbb{N}_+ = \mathbb{N}\backslash\{0\}$ is the set of positive natural numbers, the maximal computing power of BS_m (in ECUs).

The miners compete against each other to solve a chain of blocks representing the PoW puzzle through a try-and-guess strategy. The occurrence of mining a block is modeled as a random variable following a Poisson distribution with the mean of $1/T_b$, where T_b is the expected block interval time, typically equal to 600 seconds [9], [13], [24]. The process of mining a blockchain consists of the number of stages, denoted as $t = 0, \ldots, T$, during which every subsequent block of PoW puzzle is solved. The SP and all miners remain in the system

for an indefinitely-long time. Thus, even if the duration of their stay in the system may be finite, it is unknown. Hence, the total number of mining stages is $T \to \infty$. It is assumed that during one mining stage, all the network parameters (e.g., locations of miners, miner-BS associations, wireless channel quality and channel gains) remain constant.

At the beginning of stage t, the SP assigns the current price $p_t \in \mathbf{P}$ per ECU and announces it to all miners. To be compatible with the analytical framework of this paper, we assume that all payments in the blockchain can take any value from the finite discrete set $\mathbf{P} = \{1, \ldots, P\}$, where P is a maximal possible payment. Similarly, the budgets of miners can take any value from from the finite discrete set $\mathbf{B} = \{1, \ldots, B\}$, where B is a maximal miner's budget. Given the ECU's price p_t, every miner U_n, $n \in \mathbf{N}$, decides on its hash rate, i.e., computing power $a_t^n \in \mathbf{A}^n$ (in ECUs), where a finite discrete set $\mathbf{A}^n = \{1, \ldots, a_n\}$ of possible miner's hash rate decisions is limited by the maximal hash rate a_n that can be selected by miner U_n. As such, a_n is contrained by the maximal computing power of the BS associated with miner U_n. That is, for any BS_m, we must have $\sum_{n \in \mathbf{N}_m} a_n \leq \mathcal{A}_m$. For example, the SP can restrict the maximal hash rate of each miner U_n associated with some BS_m to stay within $a_n \leq \lfloor \mathcal{A}_m / N_m \rfloor$, where $\lfloor x \rfloor$ is the floor of x; $N_m = |\mathbf{N}_m|$ is the number of miners associated with BS_m. After deciding on their hash rates, miners start (simultaneously) solving a new block of puzzle. Once the solution is found, it is propagated across the entire blockchain network to reach a consensus. Any miner which is the first to find a solution that reaches consensus receives a mining reward. At this moment, the current stage t of the mining process concludes and the next stage $t+1$ begins.

3.2 Mining Rewards and Payoffs of the Miners

Let $r_t^n \in \mathbf{R}^n$ be the reward of miner U_n resulting from its hash rate decision a_t^n at stage t, where $\mathbf{R}^n = \{0,\ R + rx^n\}$ is a finite discrete set of possible miner's rewards. Specifically, $r_t^n = R + rx^n > 0$, if the miner's solution is the first to reach consensus, and $r_t^n = 0$ otherwise. A positive reward includes a fixed part $R \in \mathbf{P}$ and a variable part rx^n. A variable part is equal to the product of a given reward factor $r \in \mathbf{P}$ and a block size defined as the number of transactions $x^n \in \mathbb{N}_+$ chosen by miner U_n. The block size does not affect the complexity or cost of its mining. However, the chances of winning a mining reward are decreasing when the block propagates more slowly. That is, if the size x^n is too large, the block is likely to be discarded due to long latency which is called orphaning [9], [13], [24].

Let $(a_t^n, \mathbf{a}_t^{-n}) \in \mathbf{A} = \times_{i \in \mathbf{N}} \mathbf{A}^i$ be the miners' hash rate profile, where $\mathbf{a}_t^{-n} = \{a_t^i\}_{i \in \mathbf{N} \backslash \{n\}} \in \mathbf{A}^{-n} = \times_{i \in \mathbf{N} \backslash \{n\}} \mathbf{A}^i$ denotes the hash rates of all miners except U_n. Then, for any miner U_n, the probability of reward $r^n \in \mathbf{R}^n$ given the hash rate profile $(a_t^n, \mathbf{a}_t^{-n})$ is defined by

$$\Pr\{r_t^n = r^n | a_t^n, \mathbf{a}_t^{-n}\} = \begin{cases} P_s^n(a_t^n, \mathbf{a}_t^{-n})\big(1 - P_o^n\big), & r^n = R + rx^n, \\ 1 - P_s^n(a_t^n, \mathbf{a}_t^{-n})\big(1 - P_o^n\big), & r^n = 0 \end{cases}. \tag{1}$$

In (1), $P_s^n(a_t^n, \mathbf{a}_t^{-n})$ is the probability that miner U_n solves the block that is proportional to a relative hash rate of the miner, i.e.,

$$P_s^n\big(a_t^n, \mathbf{a}_t^{-n}\big) = \frac{a_t^n}{\sum_{i \in \mathbf{N}} a_t^i}; \tag{2}$$

P_o^n is the probability of orphaning which depends on the transmission delay in the wireless channel between miner U_n and its associated BS and can approximated as [9], [24]

$$P_o^n = 1 - e^{-z^n x^n / T_b}, \tag{3}$$

where $z^n > 0$ is the delay factor, i.e., transmission time per transaction of miner U_n. Note that $P_s^n(a_t^n, \mathbf{a}_t^{-n}) \in (0, 1]$ and $P_o^n \in (0,\ 1)$, since $a_t^n > 0$ and $z^n x^n > 0$, for all $n \in \mathbf{N}$.

Apparently, z^n can be computed from the current data rate of a wireless channel between the miner and its associated BS, as in

$$z^n \propto \frac{1}{b^n \log_2\left(1 + \sum_{m \in \mathbf{M}} \frac{\mathbf{1}_{n \in \mathbf{N}_m} p^n G^{n,m}}{\sum_{i \in \mathbf{N} \backslash \{n\}} \beta^{i,n} p^i G^{i,m} + \sigma^2}\right)}, \tag{4}$$

where b^n is the bandwidth of wireless channel of miner U_n; $\beta^{i,n} \in \{0,\ 1\}$ is the spectrum overlap indicator, such that $\beta^{i,n} = 1$, if the bandwidth b^n of miner U_n overlaps with the bandwidth b^i of another miner U_i; p^i and $G^{i,m}$ are the transmit power of miner U_i and the channel gain of wireless link between miner U_i and BS_m, respectively; σ^2 is the variance of a zero-mean additive white Gaussian noise (AWGN) power; $\mathbf{1}_{n \in \mathbf{N}_m} = 1$, if $n \in \mathbf{N}_m$ and $\mathbf{1}_{n \in \mathbf{N}_m} = 0$, otherwise. Note that the expression in (4) presumes the possibility of interference between transmissions of miners, although the interference-free scenario is also possible. For example, the bandwidth can be allocated so that $\sum_{n \in \mathbf{N}} \sum_{i \in \mathbf{N}} \beta^{i,n} = 0$ and $\sum_{n \in \mathbf{N}} b^n \leq \mathcal{B}$, where $\mathcal{B}$ is the operating bandwidth of the network. Note that at any mining stage, miner U_n knows the current exact values of b^n, p^n, $G^{n,m}$, and σ^2, for its wireless channels (between the miner and its associated BS_m). All other parameters, i.e., the values of p^i and $G^{i,m}$ for any miner U_i transmitting over the overlapping band (i.e., for all $i \in \{j \in \mathbf{N} \backslash \{n\} \mid \beta^{j,n} = 1\}$, can be reported to U_n by its associated BS at the end or beginning of each mining stage. Hence, at any mining stage, miner U_n can accurately estimate its current delay factor z^n that reflects the time- and location-dependent characteristics of the wireless channel between miner U_n and its associated BS.

Given the ECU's price p_t assigned by the SP, the instantaneous payoff that miner U_n receives at the end of stage t from its hash rate decision $a_t^n \in \mathbf{A}^n$ resulting in the reward $r_t^n \in \mathbf{R}^n$ is expressed by

$$u^n\big(a_t^n, r_t^n | p_t\big) = r_t^n - p_t a_t^n. \tag{5}$$

The payoff of miner U_n in (5) is a function of:

- ECU's price p_t observable by miner U_n;
- hash rate a_t^n decided by miner U_n;
- a random mining reward r_t^n which (as follows from the probability expression in (1)) depends on the hash rates $(a_t^n, \mathbf{a}_t^{-n})$ selected by all miners, where $\mathbf{a}_t^{-n}$ is unobservable by miner U_n.

As no miner U_n is certain about the hash rate profile $\mathbf{a}_t^{-n}$ of other miners, it cannot know its payoff in (5) while making its hash rate decision. Nevertheless, any miner U_n can construct

its own estimate of the profile $\mathbf{a}_t^{-n}$. This estimate, also called a "belief", represents a joint probability distribution $B_t^n = \{B_t^n(\mathbf{a}^{-n})\}_{\mathbf{a}^{-n}\in\mathbf{A}^{-n}}$, where each $B_t^n(\mathbf{a}^{-n}) = \prod_{i\in\mathbf{N}\setminus\{n\}} B_t^n(a^i)$ is the probability $\Pr\{\mathbf{a}_t^{-n} = \mathbf{a}^{-n}\}$ that miner U_n assigns to other miners having the hash rate profile $\mathbf{a}^{-n}$ at stage t and $B_t^n(a^i)$ is the marginal probability. The initial miner's belief $B_0^n(\mathbf{a}^{-n})$ is estimated (as in [25], [26], [27]) based on, e.g., historical data on $\mathbf{a}^{-n}$. If no data are available, the initial distribution B_0^n is uniform, i.e., all hash rates are assumed equiprobable. After this, the beliefs $B_t^n, t = 1, \ldots, T$, are updated using Bayes' theorem [25], [26], [27], as described in Section 4.

3.3 Stochastic Optimization Problem for the SP

The SP aims at maximizing the long-term payoff from its services. Since the SP remains in the system for an indefinitely-long time, its long-term payoff is defined by the infinite discounted sum of its stage payoffs [26], [27], [28], i.e.,

$$V = \sum_{t=0}^{\infty} \gamma^t u(p_t|\mathbf{a}_t), \tag{6}$$

where $\gamma \in (0, 1]$ is a discount factor; $u(p_t|\mathbf{a}_t)$ is the payoff from the ECU's price p_t at stage t given the miners' hash rates $\mathbf{a}_t = \{a_t^n\}_{n\in\mathbf{N}} \in \mathbf{A} = \times_{n\in\mathbf{N}}\mathbf{A}^n$, expressed by

$$u(p_t|\mathbf{a}_t) = (p_t - \xi T_b) \sum_{n\in\mathbf{N}} a_t^n. \tag{7}$$

In (7), ξT_b is the expected service cost per ECU equal to the product of the cost of power ξ and expected block interval time T_b.

Theoretically, the SP can set any price per ECU, including the maximal possible price (i.e., $p_t = P$). In reality, however, the miners have limited budget, i.e., the amount of currency that they can afford to spend during an entire mining process. Let $\sigma_t^n \in \mathbf{\Sigma}^n \subseteq B$ be the budget of miner U_n at the beginning of stage t, where the set of miner's budgets $\mathbf{\Sigma}^n$ is finite and discrete, as it represents the subset of the finite and discrete set $\mathbf{B}$. Before the start of a mining process (i.e., at stage $t = 0$), the information about the initial budget σ_0^n of every miner U_n is passed to the SP, which must ensure that no miner leaves the blockchain network because its budged is depleted. This requirement is common to many pricing models (e.g., [29], [30]). Formally, it can be expressed by the following budget constraint:

$$\sigma_{t+1}^n = \sigma_t^n - p_t a_t^n + r_t^n \quad \text{and} \quad \lim_{t\to\infty} \sigma_t^n \geq 0, \forall n \in \mathbf{N}. \tag{8}$$

Then, the optimization problem for the SP is defined by

$$\underset{\{p_t\in\mathbf{P}\}_{t\in\mathbb{N}}}{\text{maximize}} \sum_{t=0}^{\infty} \gamma^t u(p_t|\mathbf{a}_t), \tag{9}$$

subject to (8).

Note that the SP assigns the ECU's price p_t at the beginning of stage t, i.e., before knowing the miners' hash rate decisions $\mathbf{a}_t$. That is, $\mathbf{a}_t$ represents a random variable vector, although the SP can observe the value of $\mathbf{a}_t$ at the end of each stage t, i.e., after assigning the price p_t. Moreover, the reward r_t^n of each miner U_n is also a random variable that depends on the probabilities $P_s^n(a_t^n, \mathbf{a}_t^{-n})$ and P_o^n in (2) and (3), respectively. Therefore, (9) is a stochastic optimization problem.

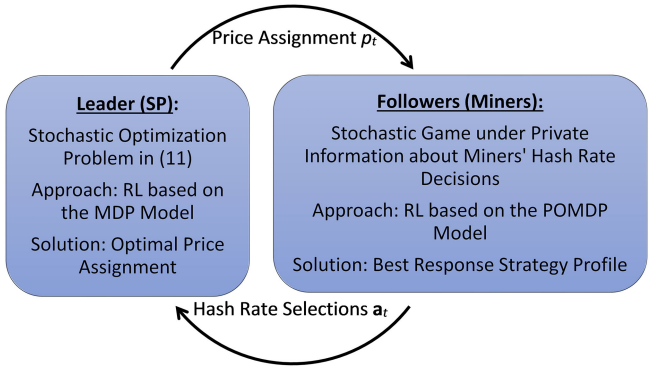

Fig. 2. Data flow between the SP and the miners in the game model.

3.4 Stackelberg Model of the Blockchain Over MEC

Based on the above description of the mining process, the interactions between the miners and the SP can be modeled as a stochastic Stackelberg game with private or incomplete information. In this game, at the beginning of mining stage t, the leader, i.e., the SP, assigns the ECU's price p_t by solving the stochastic optimization problem in (9). Every follower, i.e., miner U_n, responds to the current price assignment by selecting the hash rate a_t^n that maximizes its long-term payoff defined by the discounted sum of its stage payoffs over an infinite period in the future, i.e., $\sum_{t=0}^{\infty} \gamma^t u^n(a_t^n, r_t^n|p_t)$. At any stage t, the payoff $u^n(a_t^n, r_t^n|p_t)$ is affected not only by the miner's hash rate a_t^n and ECU's price p_t, but also by the reward r_t^n resulting from its hash rate decision. The reward r_t^n is a random variable which depends on the hash rates $\mathbf{a}_t = (a_t^n, \mathbf{a}_t^{-n})$ selected by all miners. Since no miner U_n is able to observe and know the hash rate profile $\mathbf{a}_t^{-n}$ of other miners, it must form its belief $B_t^n(\mathbf{a}^{-n})$ about $\mathbf{a}_t^{-n}$ based on the available observations r_t^n. The data flow between the SP and the miners at every stage of a stochastic Stackelberg game under private information is shown in Fig. 2.

Note that the above game model is stochastic because at every stage of the mining process, the payoffs of the leader and its followers are determined by the set of stochastic parameters with unknown state transitions. Some of these parameters, in particular, the hash rate decisions a_t^n of each miner U_n, are also private, i.e., known only by miner U_n and not by other miners. Therefore, the followers operate under incomplete information. Furthermore, recall that in the conventional Stackelberg game, the leader assigns the price by estimating the best responses of its followers by using backward induction [31]. In our game, however, the best responses of the followers depend on their beliefs about the unobservable hash rate profiles. These beliefs are unknown to a leader and, thus, it cannot estimate the best response strategies directly. Instead, it can learn these strategies via direct observations performed during repeated interactions with the followers. In the following, we formulate the game describing the interactions among the miners in the blockchain network and develop the POMDP model of miners' decision process to find the game solution. Based on the POMDP model, we propose the RL framework which will be used by the miners to update their beliefs about the hash rates of other miners and select their best response strategies in response to the price set by the SP. Next, we model the interactions between the miners and the SP as the MDP.

Based on the MDP model, we develop learning algorithms that will allow the SP to dynamically adjust the ECU's price in order to maximize its long-term expected payoff from the MEC services subject to miners' budget constraints.

4 Interactions Among Miners in Blockchain Network

4.1 Mining as a Stochastic Non-Cooperative Game With Unobservable Actions

The interactions among miners can be described by the stochastic non-cooperative game with unobservable actions played repeatedly at every stage of the mining process. In the game, in addition to external environmental states which have fully-observable but stochastic transitions, no player can observe the past or current actions of other players, i.e., the information about players' actions is private. That is, the game generalizes both the repeated games and POMDPs with partially-observable random state and unknown state transitions [32], [33]. In particular, in our game, denoted as $\mathbf{\Gamma}$, at any stage t, the private information of each player, i.e., miner U_n, represents its hash rate decision or action $a_t^n \in \mathbf{A}^n$. A fully-observable environmental state $s_t^n \in \mathbf{S}^n$ of miner U_n is represented by the reward-cost pair $(r_t^n, p_t) \in \mathbf{R}^n \times \mathbf{P}$ and has the state transitions which depend on the partially-observable action profile $(a_t^n, \mathbf{a}_t^{-n}) \in \mathbf{A}$.

The game is defined by the following elements:

- the set of players or miners $\mathbf{N}$ and, for each player U_n, $n \in \mathbf{N}$,
- the set of actions or hash rate decisions $\mathbf{A}^n$;
- a partially-observable state space $\mathbf{\Omega}^n = \mathbf{S}^n \times \mathbf{A}^{-n}$ which includes an observable state space $\mathbf{S}^n = \mathbf{R}^n \times \mathbf{P}$ equivalent to a set of all possible reward-cost pairs of player U_n and the unobservable state space $\mathbf{A}^{-n}$ equivalent to a set of possible action profiles of other players;
- the transition dynamics $\Pr\{\acute{s}^n, \acute{\mathbf{a}}^{-n} | a^n, s^n, \mathbf{a}^{-n}\}$, i.e., the probability of transitioning to state $(\acute{s}^n, \acute{\mathbf{a}}^{-n}) \in \mathbf{\Omega}^n$ given the action $a^n \in \mathbf{A}^n$ executed in state $(s^n, \mathbf{a}^{-n}) \in \mathbf{\Omega}^n$, which can be factored into two conditional distributions, $\Pr\{\acute{s}^n | a^n, s^n, \mathbf{a}^{-n}\}$ and $\Pr\{\acute{\mathbf{a}}^{-n} | \mathbf{a}^{-n}\}$;
- the payoff $u^n(a^n, s^n) = u^n(a^n, r^n | p)$ received from the action $a^n \in \mathbf{A}^n$ in state $s^n = (r^n, p) \in \mathbf{S}^n$.

Note that in conventional stochastic games (with observable actions), a strategy $\pi^n : \mathbf{\Omega}^n \rightarrow \mathbf{A}^n$ of each player U_n is a mapping from the state space $\mathbf{\Omega}^n$ to the action space $\mathbf{A}^n$ [28]. In our case, however, the state space $\mathbf{\Omega}^n$ of each player U_n comprises two subsets, $\mathbf{S}^n$ and $\mathbf{A}^{-n}$. Since the latter subset is not directly observable, the player forms its belief $B^n(\mathbf{a}^{-n})$ about unobservable actions $\mathbf{a}^{-n} \in \mathbf{A}^{-n}$. As a result, the player's strategy $\pi^n(B^n, s^n)$ in game $\mathbf{\Gamma}$ is a mapping from its belief B^n and observable state $s^n \in \mathbf{S}^n$ to the action $a^n \in \mathbf{A}^n$.

The game proceeds as follows. At the beginning of stage t, the game is in some state $s_t^n \in \mathbf{S}^n$. After observing the state s_t^n, the player U_n selects a strategy $\pi^n(B_t^n, s_t^n) = a_t^n$ that maps from its current belief B_t^n and state $s_t^n \in \mathbf{S}^n$ to the action $a_t^n \in \mathbf{A}^n$. As a result of the action $a_t^n \in \mathbf{A}^n$ executed in state $s_t^n \in \mathbf{S}^n$, the game moves to a new random state $s_{t+1}^n \in \mathbf{S}^n$ and the player U_n receives a payoff $u^n(a_t^n, s_{t+1}^n)$. The distribution of state s_{t+1}^n depends on the past state s_t^n and action a_t^n. The procedure is repeated in new state and the game continues for an infinite number of mining stages $T \rightarrow \infty$. Thus, the value, i.e., the long-term payoff of player U_n is defined by the infinite discounted sum of its stage payoffs [26], [27], [28], i.e.,

$$V^n = \sum_{t=0}^{\infty} \gamma^t u^n\big(\pi^n\big(B_t^n, s_t^n\big), s_t^n\big). \tag{10}$$

Consequently, at any stage t, given belief B_t^n and state s_t^n, player U_n will select a strategy $\pi^{n*}(B_t^n, s_t^n) = \pi_t^{n*}$ that maximizes its current expected long-term payoff, i.e.,

$$\pi_t^{n*} = \arg\max_{a^n \in \mathbf{A}^n} V_{t+1}^n\big(B_t^n, s_t^n | a^n\big). \tag{11}$$

In (11), $V^n(B_t^n, s_t^n | a^n)$ is the current expected long-term payoff of player U_n given belief B_t^n and state s_t^n, defined as

$$\begin{aligned} V^n\big(B_t^n, s_t^n | a^n\big) &= \mathrm{E}\left\{ \sum_{\tau=t}^{\infty} \gamma^{\tau - t} u^n\big(\pi^n\big(B_t^n, s_t^n\big), s_t^n\big) |\ B_t^n, s_t^n, a^n \right\} \\ &= \sum_{\mathbf{a}^{-n} \in \mathbf{A}^{-n}} B_t^n(\mathbf{a}^{-n}) V^n\big(s_t^n | a^n, \mathbf{a}^{-n}\big), \end{aligned} \tag{12}$$

where

$$\begin{aligned} V^n\big(s_t^n | a^n, \mathbf{a}^{-n}\big) = \sum_{s^n \in \mathbf{S}^n} \Pr\big\{s^n | a^n, s_t^n, \mathbf{a}^{-n}\big\} \big(u^n(a^n, s^n) \\ + \gamma V^n(s^n | a^n, \mathbf{a}^{-n})\big), \end{aligned} \tag{13}$$

is the expected long-term payoff of player U_n given state s_t^n and action profile of other players $\mathbf{a}^{-n}$.

A commonly-used solution notion for non-cooperative games is the NE, i.e., an action profile $\mathbf{a}^* = (a^{n*}, \mathbf{a}^{-n*})$ that satisfies

$$V^n(s^n | a^{n*}, \mathbf{a}^{-n*}) \geq V^n(s^n | a^n, \mathbf{a}^{-n*}), \tag{14}$$

for all $a^n \in \mathbf{A}^n$, $s^n \in \mathbf{S}^n$, $n \in \mathbf{N}$. The following lemmas establish the existence of NE in game $\mathbf{\Gamma}$ and the condition for the uniqueness of this equilibrium.

Lemma 1. *Game $\mathbf{\Gamma}$ admits at least one Nash equilibrium.*

Lemma 2. *The uniqueness of the Nash equilibrium in game $\mathbf{\Gamma}$ is guaranteed, provided that the following condition*

$$\big(1 - P_o^n\big)(R + rx^n) \geq 4p \sum_{i \in \mathbf{N} \setminus \{n\}} a_i, \quad \forall n \in \mathbf{N}, \tag{15}$$

is satisfied.

The proofs of Lemma 1 and 2 are provided in Appendices A and B, respectively. Based on these lemmas, game $\mathbf{\Gamma}$ admits the NE which is unique given the condition in (15). Note, however, that in our model, no player U_n can directly compute its best-response action a^{n*} satisfying (14), as it cannot observe the actions of other players. Instead, any player U_n can estimate a strategy $\pi^{n*}(B^n, s^n)$, which is optimal with respect to its current belief B^n in state s^n, i.e.,

$$V^n(B^n, s^n | \pi^{n*}(B^n, s^n)) \geq V^n(B^n, s^n | \pi^n(B^n, s^n)). \tag{16}$$

In the repeated game settings, player U_n can update its belief B^n about unobservable actions of other players by adopting stochastic dynamic programming and RL. Thus, in the following, we develop the RL framework that allows the

players to select the optimal strategies and learn the distributions of unobservable actions and observable states through repeated interactions with each other and the stochastic environment.

4.2 POMDP Model of the Miners' Decision Making

We start with the brief description of POMDPs which will be further used to develop our RL model. Formally, the POMDP is defined by the tuple $(\mathbf{\Omega}, \mathbf{S}, \mathbf{A}, \Pr, u,\ \gamma)$ comprising the following elements [26], [27], [28]:

- partially-observable state space $\mathbf{\Omega} = \mathbf{S} \times \mathbf{\Theta}$ that consists of the observation space, i.e., the set of fully-observable states $\mathbf{S} \subseteq \mathbf{\Omega}$, and the unknown state space $\mathbf{\Theta}$;
- set of actions $\mathbf{A}$;
- transition dynamics $\Pr\{\acute{s}, \acute{\theta}|a, s, \theta\} = \Pr\{\acute{s}|a, s, \theta\}\ \Pr\{\acute{\theta}|\theta\}$, i.e., the probability of transiting to state $(\acute{s}, \acute{\theta}) \in \mathbf{\Omega}$ given the action $a \in \mathbf{A}$ taken in state $(s, \theta) \in \mathbf{\Omega}$;
- payoff function $u(a, s)$;
- discount factor $\gamma \in (0, 1]$.

Since the transitions $\Pr\{\acute{\theta}|\theta\}$ of unknown state $\theta \in \mathbf{\Theta}$ are not directly observable, the learning agent must rely on every observed transition $(s, a, \acute{s})$ to infer the underlying probability distribution $B = \{B(\theta)\}_{\theta \in \mathbf{\Theta}}$, for $B(\theta) = \Pr\{\theta\}$. This distribution, also called a belief, can be updated using Bayes' theorem [25], [26], [27], [28], as

$$B_a^{s,\acute{s}}(\theta) = \frac{\Pr\{\acute{s}|a, s, \theta\} B(\theta)}{\sum_{\acute{\theta} \in \mathbf{\Theta}} \Pr\{\acute{s}|a, s, \acute{\theta}\} B(\acute{\theta})}, \tag{17}$$

where $B(\theta)$ is the prior belief, $B_a^{s,\acute{s}}(\theta)$ is an updated belief.

The RL problem consists of finding a strategy $\pi(B) \in \mathbf{A}$ (mapping from the beliefs to actions) that achieves a maximal value. In RL, the value $V^{\pi}(B)$ of strategy π represents the expected discounted sum of the payoffs received while executing it. That is,

$$V^{\pi}(B) = \sum_{t=0}^{\infty} \gamma^t u(\pi(B_t), s_t), \tag{18}$$

where B_t and s_t are, respectively, the belief and state at stage t of the learning process. A strategy π^* is optimal if its value is maximal in all belief states, i.e., $V^{\pi^*}(B) \geq V^{\pi}(B)$, for all π and B, and satisfies the Bellman optimality equation [28]:

$$V^{\pi^*}(B) = \max_{a \in \mathbf{A}} \sum_{\acute{s} \in \mathbf{S}} \Pr\{\acute{s}|a, s, B\} \Big(u(a, \acute{s}) + \gamma V^{\pi^*}\big(B_a^{s,\acute{s}}\big) \Big), \tag{19}$$

where $B_a^{s,\acute{s}}$ is an updated belief computed using (17).

Apparently, the problem of determining the players' best response strategies in a stochastic game under incomplete or private information can be cast as a POMDP [31], [32], [33]. More specifically, when applied to game $\mathbf{\Gamma}$, the POMDP for player/miner U_n, $n \in \mathbf{N}$, is represented by the tuple $(\mathbf{\Omega}^n, \mathbf{S}^n, \mathbf{A}^n, \Pr, u^n,\ \gamma)$, with elements described in the previous subsection. In our case, the strategy of the player $\pi^n(B^n, s^n) \in \mathbf{A}^n$ maps from its belief B^n and state s^n to the action a^n. Hence, the RL problem is to find an optimal strategy π^{n*} maximizing the value in (12). We can solve this problem by stochastic dynamic programming [33], [34]. In this case, we obtain a dynamic programming algorithm comprising four steps repeated at each stage of the mining/learning process. At any stage t, we store a set of $|\mathbf{S}^n \times \mathbf{A}|$ updated values $V_{t+1}^n(s^n|a^n, \mathbf{a}^{-n})$ indexed by state $s^n \in \mathbf{S}^n$ and action profile $(a^n, \mathbf{a}^{-n}) \in \mathbf{A}$. The algorithm terminates when the value V_t^n reaches a stable state, i.e., when

$$\begin{aligned} &\left\| V_{t+1}^n - V_t^n \right\| \\ &= \sqrt{\sum\nolimits_{s^n \in \mathbf{S}^n} \sum\nolimits_{a^n \in \mathbf{A}^n} \left(V_{t+1}^n\big(B_{t+1}^n, s^n|a^n\big) - V_t^n(B_t^n, s^n|a^n)\right)^2} \leq \delta, \end{aligned} \tag{20}$$

where $\delta \in [0,\ 1)$ is some infinitesimal number.

The algorithm proceeds as follows:

1) At the beginning of stage t, for the current state s_t^n and each possible transition $(s_t^n, a^n, \acute{s}^n)$, update the belief according to Bayes' theorem, as

$$B_{a^n}^{s_t^n,\acute{s}^n}(\mathbf{a}^{-n}) = \frac{\Pr\big\{\acute{s}^n|a^n, s_t^n, \mathbf{a}^{-n}\big\} B_t^n(\mathbf{a}^{-n})}{\sum_{\acute{\mathbf{a}}^{-n} \in \mathbf{A}^{-n}} \Pr\{\acute{s}^n|a^n, s_t^n, \acute{\mathbf{a}}^{-n}\} B_t^n(\acute{\mathbf{a}}^{-n})}, \tag{21}$$

for all $\mathbf{a}^{-n} \in \mathbf{A}^{-n}$.

2) Compute the value of an updated belief for the current state s_t^n from the Bayesian exploration equation [26], [33]:

$$V_t^n\Big(B_{a^n}^{s_t^n,\acute{s}^n}, \acute{s}^n|a^n\Big) = \sum_{\mathbf{a}^{-n} \in \mathbf{A}^{-n}} B_{a^n}^{s_t^n,\acute{s}^n}(\mathbf{a}^{-n}) V_t^n(\acute{\mathbf{a}}^n|a^n, \mathbf{a}^{-n}), \tag{22}$$

for all $\acute{s}^n \in \mathbf{S}^n$, $a^n \in \mathbf{A}^n$.

3) Find the optimal strategy $\pi^{n*}(B_t^n, s_t^n) = \pi_t^{n*}$ at stage t by solving the Bellman equation:

$$\pi_t^{n*} = \arg\max_{a^n \in \mathbf{A}^n} V_{t+1}^n\big(B_t^n, s_t^n|a^n\big), \tag{23}$$

where

$$\begin{aligned} V_{t+1}^n(B_t^n, s^n|a^n) &= \sum_{\acute{s}^n \in \mathbf{S}^n} \Pr\{\acute{s}^n|a^n, s^n, B_t^n\}(u^n(a^n, \acute{s}^n) \\ &+ \gamma V_t^n(B_{a^n}^{s^n,\acute{s}^n,\acute{s}^n|a^n}) = \sum_{\mathbf{a}^{-n} \in \mathbf{A}^{-n}} B_t^n(\mathbf{a}^{-n}) V_{t+1}^n(s^n|a^n, \mathbf{a}^{-n}); \end{aligned} \tag{24}$$

$$\begin{aligned} V_{t+1}^n(s^n|a^n, \mathbf{a}^{-n}) = \sum_{\acute{s}^n \in \mathbf{S}^n} \Pr\{\acute{s}^n|a^n, s^n, \mathbf{a}^{-n}\}(u^n(a^n, \acute{s}^n) \\ + \gamma V_t^n(\acute{s}^n|a^n, \mathbf{a}^{-n})), \end{aligned} \tag{25}$$

for all $s^n \in \mathbf{S}^n$, $a^n \in \mathbf{A}^n$.

4) After executing the action $a_t^n = \pi_t^{n*}$ and observing the state $\acute{s}^n = s_{t+1}^n$ resulting from this action at the end of stage t, set the next belief $B_{t+1}^n(\mathbf{a}^{-n})$ based on the last realized transition $(s_t^n, a_t^n, s_{t+1}^n)$, as

$$B_{t+1}^n(\mathbf{a}^{-n}) = B_{a_t^n}^{s_t^n, s_{t+1}^n}(\mathbf{a}^{-n}),\ \forall \mathbf{a}^{-n} \in\ \mathbf{A}^{-n}. \tag{26}$$

Lemma 3 below establishes that the proposed dynamic programming algorithm which represents the sequence of iterations $\{V_t^n\}_{t\in\mathbb{N}}$, converges to a fixed optimal value V^{n*}, i.e., the value of an optimal strategy $\pi^{n*} = \pi^{n*}(B^{n*}, s^n)$. In other words, this lemma shows that $\lim_{t\to\infty} V_t^n = V^{n*}$ and, hence, $\lim_{t\to\infty} \pi_t^{n*} = \pi^{n*}$.

Lemma 3. *The learning process defined by the sequence of iterations $\{V_t^n\}_{t\in\mathbb{N}}$ in (21), (22), (23), (24), (25), (26), converges to a fixed optimal value, i.e., the solution V^{n*} of an exact POMDP, for each $n \in \boldsymbol{N}$, with probability one as time tends to infinity.*

The proof of Lemma 3 is given in Appendix C. Based on Lemma 3, we obtain the following corollary that establishes that the learning process $\{V_t^n\}_{t\in\mathbb{N}}$ converges to a stable state in which the players' strategies form the NE defined by (14).

Corollary 1. *With probability one as time tends to infinity, the learning process represented by the sequence of iterations $\{V_t^n\}_{t\in\mathbb{N}}$ in (21) – (26), converges to a stable state in which the players' strategies are in the NE.*

The proof of Corollary 1 is provided in Appendix D.

4.3 Exact Model-Based RL by the Miners

We now discuss the implementation of the proposed dynamic programming algorithm in (21) – (26). Note that in game $\boldsymbol{\Gamma}$, the state $s_{t+1}^n = \acute{s}^n \in \mathbf{S}^n$ resulting from the action $a^n = a_t^n \in \mathbf{A}$ taken at stage t in state $s_t^n \in \mathbf{S}^n$ is represented by the pair (r_t^n, p_t) which comprises:

- reward $r_t^n \in \mathbf{R}^n$ received by player/miner U_n at the end of stage t as a result of players' actions $(a_t^n, \mathbf{a}_t^{-n}) \in \mathbf{A}$;
- ECU's cost $p_t \in \mathbf{P}$ assigned by the SP at the beginning of stage t.

Clearly, the random variables r_t^n and p_t are independent of each other. Furthermore, the action profile $(a_t^n, \mathbf{a}_t^{-n})$ has no impact on the ECU's cost p_t, since this cost is assigned by the SP before the corresponding actions are realized by players. Thus, the transition probability $\Pr\{\acute{s}^n|a^n, s^n, \mathbf{a}^{-n}\}$ in (21) takes the form

$$\Pr\{\acute{s}^n|a^n, s^n, \mathbf{a}^{-n}\} = \Pr\{\acute{r}^n|a^n, r^n, \mathbf{a}^{-n}\}\Pr\{\acute{p}|p\}, \tag{27}$$

where the probability $\Pr\{\acute{r}^n|a^n, r^n, \mathbf{a}^{-n}\}$ is equivalent to the probability $\Pr\{r_t^n = r^n|a^n, \mathbf{a}^{-n}\}$ in (1).

Combining (1), (21), and (27), we arrive at the following expression for the belief update rule in (21):

$$B_{a^n}^{s_t^n,\acute{s}^n}(\mathbf{a}^{-n}) = B_{a^n}^{r^n}(\mathbf{a}^{-n}) = \begin{cases} \frac{P_s^n(a^n,\mathbf{a}^{-n})B_t^n(\mathbf{a}^{-n})}{\psi_1}, & r^n = R + rx^n, \\ \frac{\left(1+P_o^n-P_s^n(a^n,\mathbf{a}^{-n})\right)B_t^n(\mathbf{a}^{-n})}{\psi_2}, & r^n = 0, \end{cases} \tag{28}$$

which does not depend on $\Pr\{\acute{p}|p\}$. In (28), the probabilities $P_s^n(a^n, \mathbf{a}^{-n})$ and P_o^n are defined in (2) and (3), respectively; ψ_1 and ψ_2 are normalizing factors, given by

$$\psi_1 = \sum_{\acute{\mathbf{a}}^{-n}\in\mathbf{A}^{-n}} P_s^n(a^n, \acute{\mathbf{a}}^{-n})B_t^n(\acute{\mathbf{a}}^{-n}); \tag{29}$$

$$\psi_2 = \sum_{\acute{\mathbf{a}}^{-n}\in\mathbf{A}^{-n}} \left(1 - P_s^n\left(a_t^n, \acute{\mathbf{a}}^{-n}\right)\left(1 - P_o^n\right)\right)B_t^n(\acute{\mathbf{a}}^{-n}). \tag{30}$$

The transition probability $\Pr\{\acute{s}^n|a^n, s^n, B_t^n\}$ in the Bellman equation (24) is equal to the product

$$\Pr\left\{\acute{s}^n|a^n, s^n, B_t^n\right\} = \Pr\left\{\acute{r}^n|a^n, r^n, B_t^n\right\}\Pr\{\acute{p}|p\}, \tag{31}$$

where

$$\Pr\left\{\acute{r}^n|a^n, r^n, B_t^n\right\} = \sum_{\mathbf{a}^{-n}\in\mathbf{A}^{-n}} B_t^n(\mathbf{a}^{-n})\Pr\{\acute{r}^n|a^n, r^n, \mathbf{a}^{-n}\}. \tag{32}$$

Hence, we have

$$\Pr\left\{\acute{s}^n|a^n, s^n, B_t^n\right\} = \begin{cases} \left(1 - P_0^n\right)\Pr\{\acute{p}|p\}\psi_1, & r^n = R + rx^n, \\ \Pr\{\acute{p}|p\}\psi_2, & r^n = 0. \end{cases} \tag{33}$$

To estimate the transition probability $\Pr\{\acute{p}|p\}$ in (33), we utilize a counting variable $\mu_t^n(\acute{p}, p)$ to count the number of times that each transition $(\acute{p}, p)$ has occurred and, then, approximate $\Pr\{\acute{p}|p\}$ as

$$\hat{\Pr}_t\{\acute{p}|p\} = \frac{\mu_t^n(\acute{p}, p)}{\sum_{\tilde{p}\in\mathbf{P}} \mu_t^n(\tilde{p}, p)}, \ \forall \acute{p}, p \in \mathbf{P}, \tag{34}$$

where

$$\mu_t^n(\acute{p}, p) = \mu_{t-1}^n(\acute{p}, p) + 1_{\acute{p}=p_t, p=p_{t-1}} \tag{35}$$

at the beginning of stage t. Clearly, the approximation in (34) and (35) becomes more accurate with time [35], i.e., $\lim_{t\to\infty} \hat{\Pr}_t\{\acute{p}|p\} = \Pr\{\acute{p}|p\}$.

Based on the above, we present the following RL algorithm. At stage $t = 0$, for any miner U_n, we initialize the belief B_0^n, value V_0^n, and counting variable μ_0^n. Then, at stage $t = 0, \ldots, T$ (where T is a recursion depth [35] which, in general, can be infinite, i.e., $T \to \infty$), we repeat the steps below until there are no further changes in the value V_t^n, i.e., until the condition in (20) is satisfied:

1) Estimate the transition dynamics $\hat{\Pr}\{\acute{p}|p\}$ according to (34) and (35);
2) Given current state s_t^n, update the belief $B_{a^n}^{s_t^n,\acute{s}^n}(\mathbf{a}^{-n})$ for each transition $(s_t^n, a^n, \acute{s}^n)$ by using the Bayes' rule in (28);
3) Determine the value $V_t^n(B_{a^n}^{s_t^n,\acute{s}^n}, \acute{s}^n|a^n)$, for all $\acute{s}^n \in \mathbf{S}^n$, $a^n \in \mathbf{A}^n$, using the Bayesian exploration equation in (22);
4) Find the optimal strategy π_t^{n*} by solving the Bellman equation in (23) – (25).
5) Execute the action $a_t^n = \pi_t^{n*}$, observe the resulting state $\acute{s}^n = s_{t+1}^n$, and set the next belief $B_{t+1}^n(\mathbf{a}^{-n})$ based on the last realized transition $(s_t^n, a_t^n, s_{t+1}^n)$ using (26).

It is worth noting that the above model-based RL algorithm does not require any additional exploration, since it is implicit in the computation of $V_t^n(B_{a^n}^{s_t^n,\acute{s}^n}, \acute{s}^n|a^n)$ in (22). Nevertheless, when the exploration ability is limited, e.g., because of the finite recursion depth T, it may be useful to deploy some explicit exploration method [35]. For example, we can adopt a

simple ε-greedy exploration. In this method, for a certain arbitrary small $\varepsilon \in [0,1)$, the optimal strategy $\pi_t^{n*} \in \mathbf{A}^n$ is executed with probability $1-\varepsilon$, whereas, all other non-optimal strategies $\pi_t^n \in \mathbf{A}^n \backslash \pi_t^{n*}$ are selected uniformly at random with probability ε. Furthermore, note that the size of a partially-observable state space in the game is exponential in the number of players N, because we have $|\mathbf{A}^{-n}| = |\times_{i \in \mathbf{N}\backslash\{n\}} \mathbf{A}^i| = (M+1)^{N-1}$ and $|\mathbf{S}^n| = |\mathbf{R}^n \times \mathbf{P}| = 2(K+1)$. This gives $|\mathbf{\Omega}^n| = |\mathbf{S}^n \times \mathbf{A}^{-n}| = 2(K+1)(M+1)^{N-1}$. Hence, the proposed RL algorithm based on the exact POMDP is intractable due to its exponential time complexity and, to be practically-realizable, we introduce its approximation in the following.

4.4 Approximate Model-Based RL by the Miners

A heuristic technique to approximate the solution of exact POMDP and reduce the size of the unobservable state space $\mathbf{A}^{-n}$ is a maximum a posteriori estimation (MAPE) [26]. In MAPE, player/miner U_n assumes that the unobservable actions $\mathbf{a}^{-n}$ of other players coincide with the most probable, with respect to the current player's belief B_t^n, actions $\hat{\mathbf{a}}^{-n} = \{\hat{a}^i\}_{i \in \mathbf{N}\backslash\{n\}}$. Thus, MAPE is equivalent to exact RL where the belief B_t^n of player U_n is approximated as

$$\hat{B}_t^n(\mathbf{a}^{-n}) = \begin{cases} 1, & \mathbf{a}^{-n} = \hat{\mathbf{a}}^{-n} = \arg\max_{\mathbf{a}^{-n} \in \mathbf{A}^{-n}} B_t^n(\mathbf{a}^{-n}), \\ 0, & otherwise. \end{cases} \tag{36}$$

Therefore, the estimation of values $V_{t+1}^n(B_t^n, s^n|a^n)$ for the actual player's belief B_t^n reduces to the estimation of values $\hat{V}_{t+1}^n(\hat{B}_t^n, s_t^n|a^n) = \hat{V}_{t+1}^n(s^n|a^n,\ \hat{\mathbf{a}}^{-n})$ given most probable, from the player's point of view, action profile $\hat{\mathbf{a}}^{-n}$.

As a result, we obtain the following algorithm where, at every stage t, we store the set of $|\mathbf{S}^n \times \mathbf{A}^n|$ updated value-functions $\hat{V}_{t+1}^n(s^n|a^n, \hat{\mathbf{a}}^{-n})$ indexed by the state $s^n \in \mathbf{S}^n$ and action $a^n \in \mathbf{A}^n$. At stage $t=0$, for every miner U_n, we initialize the belief B_0^n, value $\hat{V}_0^n$, and counting variable μ_0^n. Then, at stage $t = 0, \ldots, T$, we repeat the steps below until there are no further changes in the value $\hat{V}_t^n$, i.e., the condition in (20) is satisfied:

1) Update the transition dynamics $\hat{\Pr}\{\acute{p}|p\}$ according to (34) and (35);
2) Given the current belief B_t^n, determine the most probable action profile $\hat{\mathbf{a}}^{-n} = \{\hat{a}^i\}_{i \in \mathbf{N}\backslash\{n\}}$ of other players from

$$\hat{a}^i = \arg\max_{a^i \in \mathbf{A}^i} B_t^n(a^i),\ \forall i \in \mathbf{N}\backslash\{n\}; \tag{37}$$

If there are two or more most probable action profiles, one of them is chosen uniformly at random.

3) Determine a near-optimal strategy $\hat{\pi}_t^{n*}$ by solving the approximate Bellman equation

$$\begin{aligned} \hat{\pi}_t^{n*} &= \arg\max_{a^n \in \mathbf{A}^n} \hat{V}_{t+1}^n\big(\hat{B}_t^n, s_t^n|a^n\big) \\ &= \arg\max_{a^n \in \mathbf{A}^n} \hat{V}_{t+1}^n\big(s_t^n|a^n, \hat{\mathbf{a}}^{-n}\big), \end{aligned} \tag{38}$$

where

$$\begin{aligned} &\hat{V}_{t+1}^n\big(\hat{B}_t^n, s^n|a^n\big) \\ &= \max_{a^n \in \mathbf{A}^n} \sum_{\mathbf{a}^{-n} \in \mathbf{A}^{-n}} \hat{B}_t^n(\mathbf{a}^{-n}) \sum_{\acute{s}^n \in \mathbf{S}^n} \Pr\{\acute{s}^n|a^n, s^n, \mathbf{a}^{-n}\}(u^n(a^n, \acute{s}^n) \\ &\quad + \gamma \hat{V}_t^n\Big(\hat{B}_{a^n}^{s^n,\acute{s}^n}, \acute{s}^n|a^n\Big)\Big) = \hat{V}_{t+1}^n(s^n|a^n, \hat{\mathbf{a}}^{-n}) \\ &= \max_{\ a^n \in \mathbf{A}^n} \sum_{\acute{s}^n \in \mathbf{S}^n} \Pr\{\acute{s}^n|a^n, s^n, \hat{\mathbf{a}}^{-n}\}(u^n(a^n, \acute{s}^n) \\ &\quad + \gamma \hat{V}_t^n(\acute{s}^n|a^n, \hat{\mathbf{a}}^{-n})\big),\ \forall s^n \in \mathbf{S}^n,\ a^n \in \mathbf{A}^n; \end{aligned} \tag{39}$$

4) With probability $1-\varepsilon$, execute the near-optimal action $a_t^n = \hat{\pi}_t^{n*}$. With probability ε, execute another action $a_t^n \in \mathbf{A}^n \backslash \hat{\pi}_t^{n*}$. Observe the resulting state $\acute{s}^n = s_{t+1}^n$ and, for each $i \in \mathbf{N}\backslash\{n\}$, set the next belief $B_{t+1}^n(a^i)$ based on the last realized transition $(s_t^n, a_t^n, s_{t+1}^n)$ according to

$$\begin{aligned} B_{t+1}^n\big(a^i\big) &= B_{a_t^n}^{s_t^n, s_{t+1}^n}\big(a^i\big) \\ &= \frac{\Pr\big\{s_{t+1}^n|a_t^n, s_t^n, a^i\big\} B_t^n(a^i)}{\sum_{\acute{a}^i \in \mathbf{A}^i} \Pr\big\{s_{t+1}^n|a_t^n, s_t^n, \acute{a}^i\big\} B_t^n(\acute{a}^i)},\ \forall a^i \in\ \mathbf{A}^i. \end{aligned} \tag{40}$$

The above approximate RL algorithm has a polynomial time complexity, since the most complex operation in the algorithm is the search of an optimal strategy $\hat{\pi}_t^{n*}$ according to the Bellman equation in (38) and (39) that has the complexity of $\mathcal{O}(|\mathbf{A}^n|^2 \times |\mathbf{S}^n|^2) = \mathcal{O}(M^2K^2)$. Note that although the algorithm converges to a certain fixed value $\hat{V}^{n*}$, this value is not necessarily the optimal one. That is, in general, $\hat{V}^{n*} \neq V^{n*}$. Lemma 4 below derives the error bound between the fixed value $\hat{V}^{n*}$ reached by the approximate RL and the optimal value V^{n*}, i.e., a solution of the POMDP, reached by the exact algorithm.

Lemma 4. *The approximate learning process defined by the sequence of iterations $\{\hat{V}_t^n\}_{t \in \mathbb{N}}$ in (37), (38), (39), (40) converges to a fixed value $\hat{V}^{n*}$ with probability one as time tends to infinity, for all $n \in \boldsymbol{N}$. Moreover, if the payoff function is bounded, i.e., $|u^n| \leq u_{max}^n < \infty$, and there exists $\epsilon \in [0,\ 1]$, such that $B^n - \hat{B}^n{}_1 \leq \epsilon$, where $\hat{B}^n$ is given by (36), then the error between the value $\hat{V}^{n*}$ and the optimal value, i.e., the solution V^{n*} of an exact POMDP, is bounded by*

$$\left|V^{n*} - \hat{V}^{n*}\right| \leq \frac{\epsilon u_{max}^n}{1-\gamma},\ \forall n \in \mathbf{N}. \tag{41}$$

The proof of Lemma 2 is given in Appendix E. Note that based on (5), the payoff of player U_n is bounded as $u^n \in [-Pa_M,\ R + rx^n]$, which yields $u_{max}^n = \max\{Pa_M,\ R + rx^n\}$. Furthermore, we have $\epsilon = 1$, since $B^n - \hat{B}^n{}_1 \leq 1$. Hence, the error bound in (41) takes the form:

$$\left|V^{n*} - \hat{V}^{n*}\right| \leq \frac{\max\{Pa_M,\ R + rx^n\}}{1-\gamma},\ \forall n \in \mathbf{N}. \tag{42}$$

5 Interactions Between the Service Provider and Miners

5.1 Exact Model-Based RL by the SP

Recall that the stochastic optimization problem of the SP in (9) depends on the random, i.e., unknown before the price per ECU is assigned, but observable after the price is assigned, environmental state. This state is determined by the miners' best response strategies and rewards received by playing these strategies. To estimate the miners' best responses before setting its price, the SP requires at least the following information from each miner U_n: i) initial belief B_0^n, ii) initial values V_0^n, iii) initial counting variables μ_0^n. Since this information is inaccessible to the SP, the direct estimation of the miners' best responses is infeasible. Instead, the SP can learn these strategies via direct observations performed during repeated interactions with the miners. Hence, in the following, we formulate the optimization problem of the SP as an MDP with fully-observable random state and unknown state transitions. Based on the MDP model, we derive the RL process that will allow the SP to dynamically adjust the ECU's price in order to maximize the long-term expected payoff from its MEC services.

In particular, the MDP for the problem in (9) is defined by the tuple $(\boldsymbol{\Omega}, \mathbf{P}, \Pr, u,\ \gamma)$ comprising the elements below:

- $\boldsymbol{\Omega} = \boldsymbol{\Sigma} \times \mathbf{A} \times \mathbf{R}$ is a fully-observable finite discrete state space of the SP represented by the Cartesian product of the set $\boldsymbol{\Sigma} = \times_{n\in\mathbf{N}}\boldsymbol{\Sigma}^n$ of possible miners' budgets, set $\mathbf{A}$ of miners' action profiles, and set $\mathbf{R} = \times_{n\in\mathbf{N}}\mathbf{R}^n$ of miners' rewards, i.e., the state at stage t is $\boldsymbol{\omega}_t = (\boldsymbol{\sigma}_t, \mathbf{a}_t, \mathbf{r}_t)$, where $\boldsymbol{\sigma}_t = \{\sigma_t^n\}_{n\in\mathbf{N}}$ and $\mathbf{r}_t = \{r_t^n\}_{n\in\mathbf{N}}$;
- $\mathbf{P}$ is the SP's action space;
- $\Pr\{\acute{\boldsymbol{\omega}}|p, \boldsymbol{\omega}\}$ is the transition dynamics, i.e., the probability of transiting to state $\acute{\boldsymbol{\omega}} \in \boldsymbol{\Omega}$ given the action $p \in \mathbf{P}$ taken in state $\boldsymbol{\omega} \in \boldsymbol{\Omega}$;
- $u(p, \boldsymbol{\omega}) = u(p|\mathbf{a})$ is the SP's payoff function;
- $\gamma \in (0, 1]$ is a discount factor.

The RL problem is to determine an optimal strategy $\pi(\boldsymbol{\omega}) \in \mathbf{P}$ mapping from the state space $\boldsymbol{\Omega}$ to the action space $\mathbf{P}$, which maximizes the value-function, given by

$$V(\boldsymbol{\omega}) = \sum_{t=0}^{\infty} \gamma^t u(\pi(\boldsymbol{\omega}_t), \boldsymbol{\omega}_t), \tag{43}$$

subject to the constraint in (8). Apparently, this problem can be solved by dynamic programming [34]. The Bellman optimality equation for the problem takes the form

$$\begin{aligned} V^*(\boldsymbol{\omega}) &= \max_{p\in\mathbf{P}} \sum_{\acute{\boldsymbol{\omega}}\in\boldsymbol{\Omega}} \Pr\{\acute{\boldsymbol{\omega}}|p, \boldsymbol{\omega}\}(u(p, \acute{\boldsymbol{\omega}}) + \gamma V^*(\acute{\boldsymbol{\omega}})) \\ &= \max_{p\in\mathbf{P}} \sum_{\acute{\boldsymbol{\sigma}}\in\boldsymbol{\Sigma}} \sum_{\acute{\mathbf{a}}\in\mathbf{A}} \sum_{\acute{\mathbf{r}}\in\mathbf{R}} \Pr\{\acute{\boldsymbol{\sigma}}, \acute{\mathbf{a}}, \acute{\mathbf{r}}|p, \mathbf{a}, \mathbf{r}\}(u(p, \acute{\mathbf{a}}) \\ &\quad + \gamma V^*(\acute{\boldsymbol{\sigma}}, \acute{\mathbf{a}}, \acute{\mathbf{r}})) = \max_{p\in\mathbf{P}} \sum_{\acute{\mathbf{a}}\in\mathbf{A}} \sum_{\acute{\mathbf{r}}\in\mathbf{R}} \Pr\{\acute{\mathbf{a}}, \acute{\mathbf{r}}|p, \mathbf{a}, \mathbf{r}\}(u(p, \acute{\mathbf{a}}) \\ &\quad + \gamma V^*(\boldsymbol{\sigma} - p\mathbf{a} + \mathbf{r}, \acute{\mathbf{a}}, \acute{\mathbf{r}})). \end{aligned} \tag{44}$$

To estimate the transition dynamics $\Pr\{\acute{\mathbf{a}}, \acute{\mathbf{r}}|p, \mathbf{a}, \mathbf{r}\}$ in (44), note that according to (1), at any stage t, the miners' rewards $\mathbf{r}_t \in \mathbf{R}$ depend only on their action profile $\mathbf{a}_t \in \mathbf{A}$. Hence, the probability $\Pr\{\acute{\mathbf{a}}, \acute{\mathbf{r}}|p, \mathbf{a}, \mathbf{r}\}$ can be expressed as

$$\Pr\{\acute{\mathbf{a}}, \acute{\mathbf{r}}|p, \mathbf{a}, \mathbf{r}\} = \Pr\{\acute{\mathbf{a}}|p, \mathbf{a}, \mathbf{r}\}\Pr\{\acute{\mathbf{r}}|\acute{\mathbf{a}}\}, \tag{45}$$

In (45), the joint probability $\Pr\{\acute{\mathbf{r}}|\acute{\mathbf{a}}\} = \Pr\{\acute{r}^1, \ldots, \acute{r}^N|\acute{\mathbf{a}}\}$ is represented by the product of marginal probabilities, i.e., $\prod_{n\in\mathbf{N}} \Pr\{\acute{r}^n|\acute{\mathbf{a}}\}$. Clearly, at any stage t, a marginal probability $\Pr\{\acute{r}^n|\acute{\mathbf{a}}\}$ is equivalent to the probability $\Pr\{r_t^n = r^n|\mathbf{a}_t\}$ in (1). On the other hand, the probability $\Pr\{\acute{\mathbf{a}}|p, \mathbf{a}, \mathbf{r}\}$ is calculated similarly to the probability $\Pr\{\acute{p}|p\}$. That is, at the beginning of stage t, we compute

$$\hat{\Pr}_t\{\acute{\mathbf{a}}|p, \mathbf{a}, \mathbf{r}\} = \frac{\mu_t(\acute{\mathbf{a}}, p, \mathbf{a}, \mathbf{r})}{\sum_{\tilde{\mathbf{a}}\in\boldsymbol{A}} \mu_t(\tilde{\mathbf{a}}, p, \mathbf{a}, \mathbf{r})},\ \forall \acute{\mathbf{a}}, \mathbf{a} \in \mathbf{A}, p \in \mathbf{P}, \tag{46}$$

where

$$\mu_t(\acute{\mathbf{a}}, p, \mathbf{a}, \mathbf{r}) = \mu_{t-1}(\acute{\mathbf{a}}, p, \mathbf{a}, \mathbf{r}) + 1_{\acute{\mathbf{a}}=\mathbf{a}_t, p=p_t, \mathbf{a}=\mathbf{a}_{t-1}, \mathbf{r}=\mathbf{r}_{t-1}}. \tag{47}$$

Based on the above MDP model, we obtain the algorithm, i.e., the sequence of iterations $\{V_t\}_{t\in\mathbb{N}}$, where at every stage t, given the current state $\boldsymbol{\omega}_t = (\boldsymbol{\sigma}_t, \mathbf{a}_t, \mathbf{r}_t)$, we determine the optimal strategy $\pi^*(\boldsymbol{\omega}_t) = \pi_t^*$ by solving the Bellman equation:

$$\pi_t^* = \arg\max_{p\in\mathbf{P}} V_{t+1}(\boldsymbol{\sigma}_t, \mathbf{a}_t, \mathbf{r}_t, p), \tag{48}$$

where

$$\begin{aligned} V_{t+1}(\boldsymbol{\sigma}, \mathbf{a}, \mathbf{r}, p) &= \sum_{\acute{\mathbf{a}}\in\mathbf{A}} \sum_{\acute{\mathbf{r}}\in\mathbf{R}} \Pr\{\acute{\mathbf{a}}, \acute{\mathbf{r}}|p, \mathbf{a}, \mathbf{r}\}(u(p, \acute{\mathbf{a}}) \\ &\quad + \gamma V_t(\boldsymbol{\sigma} - p\mathbf{a} + \mathbf{r}, \acute{\mathbf{a}}, \acute{\mathbf{r}}, p)) = \sum_{\acute{\mathbf{r}}\in\mathbf{R}} \Pr\{\acute{\mathbf{r}}|\acute{\mathbf{a}}\}(u(p, \acute{\mathbf{a}}) \\ &\quad + \gamma \sum_{\acute{\mathbf{a}}\in\mathbf{A}} \Pr\{\acute{\mathbf{a}}|p, \mathbf{a}, \mathbf{r}\} V_t(\boldsymbol{\sigma} - p\mathbf{a} + \mathbf{r}, \acute{\mathbf{a}}, \acute{\mathbf{r}}, p)), \end{aligned} \tag{49}$$

for all $\boldsymbol{\sigma} \in \boldsymbol{\Sigma}$, $\mathbf{a} \in \mathbf{A}$, $\mathbf{r} \in \mathbf{R}$, $p \in \mathbf{P}$. In the algorithm, at any stage t, we store a set of $|\boldsymbol{\Omega} \times \mathbf{P}|$ updated values $V_{t+1}(\boldsymbol{\omega}, p)$ indexed by state $\boldsymbol{\omega} \in \boldsymbol{\Omega}$ and action $p \in \mathbf{P}$. The algorithm comprises three steps which are repeated until there are no further changes in the value V_t, i.e., until:

$$\|V_{t+1} - V_t\| = \sqrt{\sum_{\boldsymbol{\omega}\in\boldsymbol{\Omega}} \sum_{p\in\mathbf{P}} \left(V_{t+1}(\boldsymbol{\omega}, p) - V_t(\boldsymbol{\omega}, p)\right)^2} \leq \delta. \tag{50}$$

The algorithm proceeds as follows. At stage $t = 0$, we initialize the values V_0 and counting variables μ_0. Then, at stage $t = 0, \ldots, T$, we repeat the steps below:

1) Observe the current state $\boldsymbol{\omega}_t = (\boldsymbol{\sigma}_t, \mathbf{a}_t, \mathbf{r}_t) \in \boldsymbol{\Omega}$ and update the transition dynamics $\hat{\Pr}_t\{\acute{\mathbf{a}}|p, \mathbf{a}, \mathbf{r}\}$ using (46) and (47);
2) Find the current optimal strategy π_t^* according to the Bellman equation in (48) and (49).
3) With probability $1 - \varepsilon$, execute the action $p_t = \pi_t^*$. With probability ε, execute another action $p_t \in \mathbf{P}\backslash\pi_t^*$.

The convergence of the above exact model-based RL algorithm to a fixed optimal value V^* that represents the solution of an exact MDP which is equivalent to the solution of the optimization problem in (9) is established in Lemma 5 below.

Lemma 5. *The learning process defined by the sequence of iterations $\{V_t\}_{t\in\mathbb{N}}$ in (48) and (49), converges to a fixed optimal value, i.e., the solution V^* of an exact MDP, with probability one as time tends to infinity.*

The proof of Lemma 5 is provided in Appendix F.

5.2 RL With Function Approximation by the SP

Note that the size of state space $\boldsymbol{\Omega}$ in our MDP model is exponential in the number of miners N. In particular, $|\boldsymbol{\Omega}| = |\times_{n\in\mathbf{N}}(\boldsymbol{\Sigma}^n \times \mathbf{A}^n \times \mathbf{R}^n)| = 2^N(P+1)^N(M+1)^N$. Thus, the exact RL algorithm becomes intractable if the number of miners is large, as it requires the computation and storage of $|\boldsymbol{\Omega}\times\mathbf{P}| = 2^N(P+1)^{N+1}(M+1)^N$ value-functions at every stage of learning process. A common approach to deal with large or continuous state spaces of MDP model is the function approximation [27], [35], [36], [37], [38], where the value-function $V_t(\boldsymbol{\omega}, p)$ is approximated as

$$\hat{V}(\boldsymbol{\omega}, p, \boldsymbol{\theta}) = \sum_{i=1}^{F} \theta^i \varphi^i(\boldsymbol{\omega}, p), \tag{51}$$

where $\boldsymbol{\theta}\in\mathbb{R}^F$ is the real-value vector of weights; $\varphi^i : \boldsymbol{\Omega}\times\mathbf{P}\to\mathbb{R}$ is monotonically-increasing, non-constant, continuous, and bounded function, sometimes called an activation function. Some common activation functions are the threshold function, piece-wise linear function, or sigmoid function [38], [39]. As such, an approximation in (51) represents a feed-forward neural network [38] with the output $\hat{V}(\boldsymbol{\omega}, p, \boldsymbol{\theta})$ and a single hidden layer that comprises F neurons, numbered $1,\ldots,F$. A hidden neuron $i\in\{1,\ldots,F\}$ has the input p, the output $\varphi^i(\boldsymbol{\omega}, p)$, and two synaptic weights, $\boldsymbol{\omega}$. and θ^i, where $\boldsymbol{\omega}$ is the weight of the synapse that goes from input p and θ^i is the weight of the synapse that connects neuron i with the output $\hat{V}(\boldsymbol{\omega}, p, \boldsymbol{\theta})$.

At any stage t of the learning process, the SP's strategy $\pi(\boldsymbol{\theta}_t)\in\mathbf{P}$ maps from the current weights $\boldsymbol{\theta}_t$ to the action p_t. An optimal strategy $\pi^*(\boldsymbol{\theta}_t) = \pi_t^*$ maximizes the approximated value, i.e.,

$$\pi_t^* = \arg\max_{p\in\mathbf{P}} \hat{V}(\boldsymbol{\omega}_t, p, \boldsymbol{\theta}_t). \tag{52}$$

The RL problem is to find the weights $\boldsymbol{\theta}^*$ which minimize the loss function, given by [35], [40], [41]

$$L(\boldsymbol{\theta}) = \mathrm{E}\Big\{ \left(\hat{V}^* - \hat{V}\right)^2 \Big\} = \mathrm{E}\Bigg\{ \Bigg(\max_{p\in\mathbf{P}} \sum_{\acute{\boldsymbol{\omega}}\in\boldsymbol{\Omega}} \Pr\{\acute{\boldsymbol{\omega}}|p,\boldsymbol{\omega}\}(u(p,\acute{\boldsymbol{\omega}}) + \gamma\hat{V}(\acute{\boldsymbol{\omega}}, p, \boldsymbol{\theta})) - \hat{V}(\boldsymbol{\omega}, \pi(\boldsymbol{\theta}), \boldsymbol{\theta}) \Bigg)^2 \Bigg\}, \tag{53}$$

where a fixed optimal value $\hat{V}^*$ is such that $|\hat{V}^* - V^*| < \epsilon$, for some infinitesimal $\epsilon\in[0,1)$.

Apparently, the above problem can be solved by stochastic gradient descent [37], [40], [41]. That is, at every stage t, we minimize the current loss L_t, given by

$$L_t = \big(y_t - \hat{V}(\boldsymbol{\omega}_{t-1}, p, \boldsymbol{\theta}_t)\big)^2, \tag{54}$$

where

$$y_t = \max_{p\in\mathbf{P}} \big(u(p, \boldsymbol{\omega}_t) + \gamma\hat{V}(\boldsymbol{\omega}_t, p, \boldsymbol{\theta}_{t-1})\big) \tag{55}$$

is an approximation target at stage t. After differentiating the loss L_t with respect to weights $\boldsymbol{\theta}_t$, we obtain the gradient:

$$\nabla_{\boldsymbol{\theta}_t} L_t = \big(\hat{V}(\boldsymbol{\omega}_{t-1}, p, \boldsymbol{\theta}_t) - y_t\big)\nabla_{\boldsymbol{\theta}_t}\hat{V}(\boldsymbol{\omega}_{t-1}, p, \boldsymbol{\theta}_t). \tag{56}$$

Then, the loss can be minimized by updating the weights in the direction of the negative gradient, i.e.,

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha_t \nabla_{\boldsymbol{\theta}_t} L_t, \tag{57}$$

for

$$\sum_{t=1}^{\infty} \alpha_t = \infty \text{ and } \sum_{t=1}^{\infty} \alpha_t^2 < \infty. \tag{58}$$

Based on the above, we arrive at the following RL algorithm where, at every stage t, we store the past weights $\boldsymbol{\theta}_{t-1}$ and state $\boldsymbol{\omega}_{t-1}$. At stage $t=0$, for each miner U_n, we set the value of F and initialize the weights $\boldsymbol{\theta}_0$. At stage $t = 1,\ldots,T$, we repeat the steps below until the loss is minimized, i.e., until $L(\boldsymbol{\theta}) < \delta$:

1) Given the past weights $\boldsymbol{\theta}_{t-1}$, observe the current state $\boldsymbol{\omega}_t\in\boldsymbol{\Omega}$ and calculate the approximation target y_t using (55);
2) Given the current state $\boldsymbol{\omega}_t$ and weights $\boldsymbol{\theta}_t$, determine the optimal strategy π_t^* according to (52).
3) With probability $1-\varepsilon$, execute the action $p_t = \pi_t^*$. With probability ε, execute another action $p_t\in\mathbf{P}\backslash\pi_t^*$.
4) Given the target y_t and past state $\boldsymbol{\omega}_{t-1}$, compute the gradient $\nabla_{\boldsymbol{\theta}_t} L_t$ and next weights $\boldsymbol{\theta}_{t+1}$ based on (56) and (57), respectively.

The proposed algorithm has a polynomial time complexity, because the most complex operations in the algorithm are related to the gradient estimation in (56) and weight updates in (57) of the complexity $\mathcal{O}(F|\mathbf{P}-1|) = \mathcal{O}(FP)$. The convergence of the proposed algorithm to the fixed optimal value $\hat{V}^*$ defined in (53), is established in Lemma 6 below.

Lemma 6. *The learning process defined by the sequence of approximations $\{\hat{V}\}_{t\in\mathbb{N}}$ in (52) – (57), converges to a fixed optimal value $\hat{V}^*$, for all $n\in\boldsymbol{N}$, with probability one as time tends to infinity.*

The proof of Lemma 6 is given in Appendix G. Note that the proposed algorithm belongs to the family of unsupervised deep Q-learning [40], [41], [42], [43] techniques. Similar to most modern deep learning models [39], it uses an artificial neural network to approximate the solution of exact MDP and, at each learning stage, updates the weights of the neural network to minimize the loss function. Unlike supervised learning methods based on feature engineering [39], the algorithm does not require any labeled data for training, which means that it can quickly adjust to dynamic environmental changes [39], [40]. The "deepness" of learning in the algorithm is determined by the number of hidden neuron layers (in our case, one layer) through which the input is mapped to the output. That is, the larger is the number of hidden layers, the "deeper" and the more accurate is the RL

model. However, the improved accuracy comes with the tradeoff in the complexity [38], [39].

Now, if we combine the proposed exact RL algorithms for the SP and miners, we will obtain the exact "hierarchical" learning procedure. In this procedure, at any stage $t = 0, \ldots, T$, the SP assigns the ECU's price p_t and updates the values V_{t+1} by applying the exact RL algorithm presented in this section. Then, given the ECU's price p_t assigned by the SP, every miner U_n takes the action a_t^n and updates the values V_{t+1}^n using the exact RL algorithm in Section 4. As such, the exact hierarchical learning procedures can be represented by the sequence of iterations $\{V,\ \{V^n\}_{n\in\mathbf{N}}\}_{t\in\mathbb{N}}$. The convergence of this sequence to fixed optimal values $\{V^*,\ \{V^{n*}\}_{n\in\mathbf{N}}\}_{t\in\mathbb{N}}$ is based on the convergence of the exact RL algorithms for the SP and miners established in Lemmas 3 and 5, respectively. Similarly, by combining the approximate RL algorithms, we obtain the approximate "hierarchical" learning procedure represented by the sequence of iterations $\{\hat{V}, \{\hat{V}^n\}_{n\in\mathbf{N}}\}_{t\in\mathbb{N}}$. The convergence of this procedure to fixed sub-optimal values $\{\hat{V}^*, \{\hat{V}^{n*}\}_{n\in\mathbf{N}}\}_{t\in\mathbb{N}}$ follows directly from the convergence of approximate RL algorithms for the SP and miners proved in Lemmas 4 and 6, respectively.

6 Performance Evaluation

6.1 Simulation Settings

We simulate a public blockchain application in the long-term evolution advanced (LTE-A) time division duplex (TDD) based MEC network. The simulation model of the network is implemented with the OPNET development package [44]. In the model, the BSs are represented by one macro-cell LTE evolved NodeB (eNB) labeled as BS_1, one micro-eNB labeled as BS_2, and $M-2$ pico-eNBs labeled as $\mathrm{BS}_3, \ldots, \mathrm{BS}_M$. The number of pico-eNBs varies from 1 (by default) to 5, i.e., the number of BSs varies from $M = 3$ (by default) to $M = 7$. The network area coincides with the macro-cell area, i.e., BS_1 is placed in the center of the network. The small-cell BSs are placed randomly in the network area. Cell radiuses of the macro-, micro and pico-eNBs are equal to 2000, 500 and 50 m, respectively. The model considers inter-cell interference (ICI). In particular, the bandwidth of macro-cell BS_1, set as $\mathcal{B}_1 = \mathcal{B} = 20$ MHz, is the same as the network bandwidth $\mathcal{B}$. The spectrum bands of small-cell BSs, set as $\mathcal{B}_2 = 10$ and $\mathcal{B}_3 = \ldots = \mathcal{B}_M = 5$ MHz, overlap fully with the spectrum band of BS_1 and partially with each other. Every BS_m allocates bandwidth $b^n = x^n\mathcal{B}_m/(\sum_{i\in\mathbf{N}_m} x^i)$ to its associated miner U_n, $n \in \mathbf{N}_m$, in proportion to its selected number of transactions x^n. The transmit power of every miner U_n, $n \in \mathbf{N}$, is equal to $p^n = 23$ dBm. The computing resources of the macro-, micro and pico-eNBs are limited to $\mathcal{A}_1 = 500$, $\mathcal{A}_2 = 100$ and $\mathcal{A}_3 = \ldots = \mathcal{A}_M = 10$ ECUs, respectively. The maximal hash rate of miner U_n, $n \in \mathbf{N}_m$, associated with BS_m is set as $a_n \leq \mathcal{A}_m/N_m$. By default, the numbers of miners associated with the macro-, micro and pico-eNBs are given by $N_1 = 50$, $N_2 = 20$ and $N_3 = \ldots = N_M = 10$, respectively. As such, the total number of miners varies from $N = 80$ (by default) to $N = 120$. All other parameters of LTE-A model (e.g., antenna gains, noise, shadowing, path loss, etc.) are set based on the 3rd Generation Partnership Project (3GPP) specifications [45].

We also consider a dynamic and random scenario in which the numbers of miners can change at any mining stage. In this scenario, the arrivals of miners at every BS follow the Poisson distribution with the mean arrival rate λ miners/hour. The time during which the miners remain in the system is distributed exponentially with the expected value of $1/\mu$ hours, where μ is the mean departure rate in miners/hour. The miners' mobility is simulated according to a random waypoint model [46]. In the model, at the end of stage t, each miner U_n randomly selects its target l_{t+1}^n within a distance of 10 m from the current location l_t^n (i.e., $\|l_{t+1}^n - l_t^n\| \in [0, 10]$). If $l_{t+1}^n \neq l_t^n$, the miner moves towards its target with a random velocity $v_{t+1}^n \in (0, v_{\max}]$, where $v_{\max} = 10$ m/s is a typical moving vehicle speed. Otherwise, if $l_{t+1}^n = l_t^n$, the miner remains in its current location. The next stage $t+1$ starts after all miners arrive at their target locations where they will remain stationary until the end of stage $t+1$ (recall that miners' locations are fixed during one mining stage). All miners operate in a typical urban environment. The miners' budgets and all payments in the blockchain are counted in the units of a digital currency, i.e., bitcoin. Similar to [9], [13], [24], the block size x^n follows a normal distribution with the mean of 200 and a variance of 5; a fixed reward part is $R = 300$; the reward factor is $r = 1$; $\xi T_b = 0.5$ and $P = 10R$. The initial miners' budgets follow a Poisson distribution with a default expected value of $7R$.

According to [28], [35], a discount factor is set as $\gamma = 0.9$; the learning rate of stochastic gradient descent is set as $\alpha_t = 1/(t+0.5)^{0.7}$; the initial values of counting variables for the SP and miner U_n are equal to $\mu_0(\hat{\mathbf{a}}, p, \mathbf{a}, \mathbf{r}) = 1$ and $\mu_0^n(\acute{p}, p) = 1$, respectively; the initial weights are set as $\boldsymbol{\theta}_0 = \{\theta_0^i\}_{i\in\{1,\ldots,F\}}$, for $F = 100$ and $\theta_0^i = 0.5$; the initial value-functions for the SP and the miners are set assuming that their future actions and states are the same as the initial ones, i.e.,

$$V_0(\boldsymbol{\sigma}, \mathbf{a}, \mathbf{r}, p) = \sum_{t=0}^{\infty} \gamma^t u(p|\mathbf{a}) = \frac{u(p|\mathbf{a})}{1-\gamma} = \frac{p - \xi T_b}{1-\gamma} \sum\nolimits_{n\in\mathbf{N}} a^n; \tag{59}$$

$$V_0^n(s^n|a^n, \mathbf{a}^{-n}) = \sum_{r^n\in\mathbf{R}^n} \Pr\{r_0^n = r^n|a^n, \mathbf{a}^{-n}\} u^n(a^n, r^n|p_0). \tag{60}$$

In the following, we evaluate the performance of the proposed exact and approximate RL, denoted as "Exact RL" and "Approx RL", respectively, realized according to a hierarchical learning procedure presented in Section 5. The performance of these algorithms is benchmarked with the performance of models below:

1) Stackelberg game under complete information denoted as "Complete Info" where at every stage t, each miner U_n has a full knowledge of the current and past actions $\mathbf{a}_t^{-n}, \mathbf{a}_{t-1}^{-n}, \ldots, \mathbf{a}_0^{-n}$ of other miners. Thus, no miner U_n needs to learn the actions of its opponents. Instead, it takes a best response action

$$a_t^{n*} = \max_{a_t^n\in\mathbf{A}^n} \sum_{r^n\in\mathbf{R}^n} \Pr\{r_t^n = r^n|a_t^n, \mathbf{a}_t^{-n}\} u^n(a_t^n, r^n|p_t), \tag{61}$$

that maximizes its expected payoff. The SP sets the current optimal ECU's price p_t^* based on anticipat-ed best responses

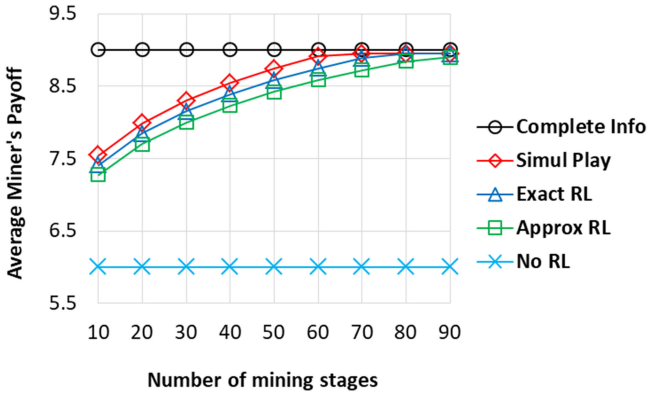

Fig. 3. Time dynamics of average miner's payoff in different models.

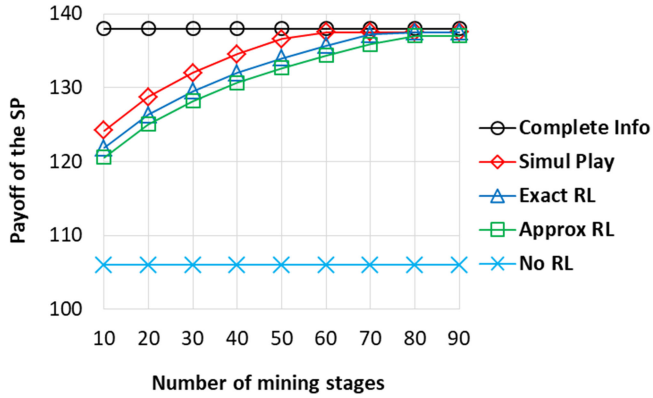

Fig. 4. Time dynamics of the SP's payoff in different models.

of miners, i.e., the NE $\mathbf{a}_t^*$. In particular, the SP decides on the price by solving a stochastic optimization problem in (9) through the Bellman equation. Given known best responses of miners, the Bellman equation takes the form:

$$p_t^* = \arg\max_{p\in\mathbf{P}} V_{t+1}(\boldsymbol{\sigma}_t, \mathbf{a}_t, \mathbf{r}_t, p), \tag{62}$$

$$V_{t+1}(\boldsymbol{\sigma}, \mathbf{a}, \mathbf{r}, p) = \sum_{\acute{\mathbf{r}}\in\mathbf{R}} \Pr\{\acute{\mathbf{r}}|\mathbf{a}_t^*\}\big(u(p, \mathbf{a}_t^*) + \gamma V_t\big(\boldsymbol{\sigma} - p\mathbf{a} + \mathbf{r}, \mathbf{a}_t^*, \acute{\mathbf{r}}, p\big)\big). \tag{63}$$

Such a Complete Info model is similar to those in [8], [9], except that here we take into account the constrained miners' budgets, whereas in [8], [9], the miners' budgets are unlimited. Since the model is rather unrealistic, we adopt it only for bench-marking the proposed algorithms.

2) Simultaneous play [47] Stackelberg game denoted as "Simul Play" where at the beginning of stage t, no miner U_n knows the immediate actions $\mathbf{a}_t^{-n}$ of other miners, although it can observe these actions after they are executed, i.e., at the end of stage t. As such, the miner's decision process is modeled as an MDP defined by the tuple $(\mathbf{S}^n, \mathbf{A}^n, \Pr, u^n, \gamma)$, where a stochastic state space $\mathbf{S}^n = \mathbf{R}^n \times \mathbf{P} \times \mathbf{A}^{-n}$ is fully-observable. The miner's best response a_t^{n*} is found from the Bellman equation:

$$a_t^{n*} = \arg\max_{a^n\in\mathbf{A}^n} V_t^n\big(s_{t-1}^n|a^n\big), \tag{64}$$

$$V_t^n(s^n|a^n) = \sum_{\acute{s}^n\in\mathbf{S}^n} \Pr\{\acute{s}^n|a^n, s^n\}(u^n(a^n, \acute{s}^n) + \gamma V_{t-1}^n(\acute{s}^n|a^n)\big). \tag{65}$$

There are two ways in which the SP can assign the price in this game: i) directly estimate the miners' best responses $\mathbf{a}_t^*$; ii) learn the best responses. The first approach is complex, as it requires solving the Bellman equation in (64) and (65) for every miner U_n. Therefore, we deploy the second approach in which the SP learns the best responses through the RL algorithm presented in Section 5. Note that this model is more realistic than Complete Info, as no miner knows immediate actions of its opponents. However, in the model, the actions of miners are fully-observable by other miners. This assumption is infeasible in the blockchain network, as miners compete against each other to obtain the reward and, hence, they are not interested in revealing their hash rate decisions to other miners.

3) Stackelberg game under private information about miners' actions with no RL involved denoted as "No RL". Here, the assumptions of the game are the same as in this paper, but neither the SP nor the miners use RL. Each miner U_n operates under its initial belief B_0^n about actions of other miners and at any stage t, it selects the best response

$$a_t^{n*} = \max_{a^n\in\mathbf{A}^n} \sum_{r^n\in\mathbf{R}^n} \Pr\{r^n|a^n, B_0^n\}\ u^n(a^n, r^n|p_t), \tag{66}$$

maximizing its expected payoff. Given $B_0^1, \ldots, B_0^N$, the SP can compute miners' best responses $\mathbf{a}_t^*$ and assign the optimal price p_t^* by solving the problem in (9) with the Bellman equation in (62) and (63).

6.2 Simulation Results

Figs. 3 and 4 show the time dynamics of the average payoffs of a miner and the SP, respectively. Note that the graphs for the Complete Info model show the average payoff of the miner at the NE and the SP's payoff given the optimal price. Observe that in Simul Play, Exact RL and Approx RL, payoffs increase consistently with time, converging to the stable close-to-optimal levels after about 70, 80, and 90 iterations, respectively. Moreover, Simul Play converges faster than Exact and Approx RL. The reason is that in Simul Play, the state s^n of miner U_n has an observable transition $\Pr\{\acute{s}^n|a^n, s^n\}$. On the contrary, in Exact RL and Approx RL, in addition to the state s^n, there is also an unobservable state $\mathbf{a}^{-n}$ which is estimated based on the observable transitions by updating miners' beliefs with the Bayes' rule. Clearly, RL in POMDPs takes a longer time (i.e., number of stages) than that in MDPs, because it requires convergence of the beliefs before convergence of the best responses. Hence, Simul Play converges slower than Exact or Approx RL. Finally, note that Exact RL converges faster than Approx RL, since the latter is based on the approximated solution. Therefore, to reach a stable state, it requires more exploration.

Fig. 5 shows the convergence time of a hierarchical learning algorithm and its components, i.e., RL for the SP and RL for the miners, depending on the number of miners N.

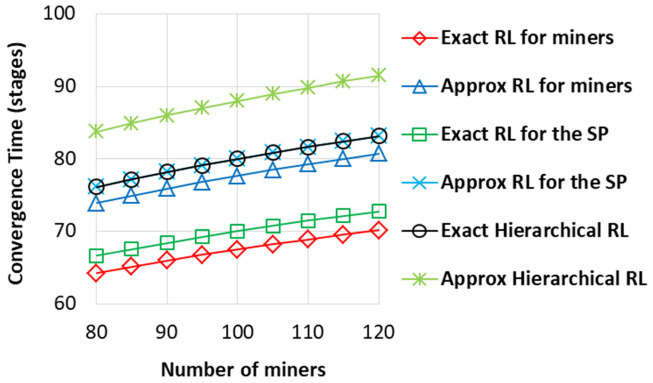

Fig. 5. Convergence time of hierarchical RL and its components as a function of the number of miners.

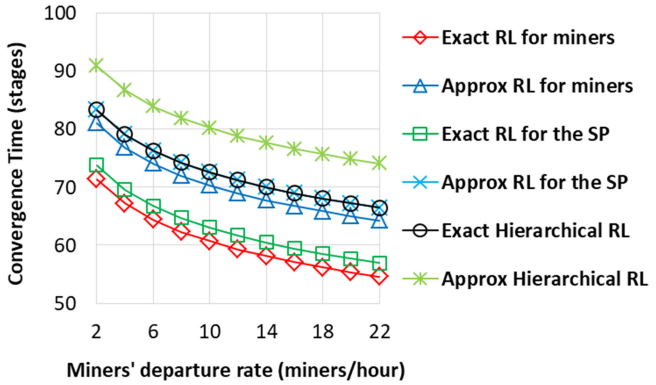

Fig. 7. Convergence time of hierarchical RL and its components as a function of miners' mean departure rate per BS.

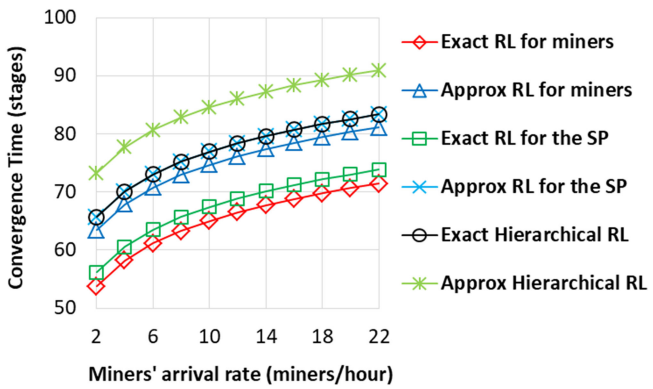

Fig. 6. Convergence time of hierarchical RL and its components as a function of miners' mean arrival rate per BS.

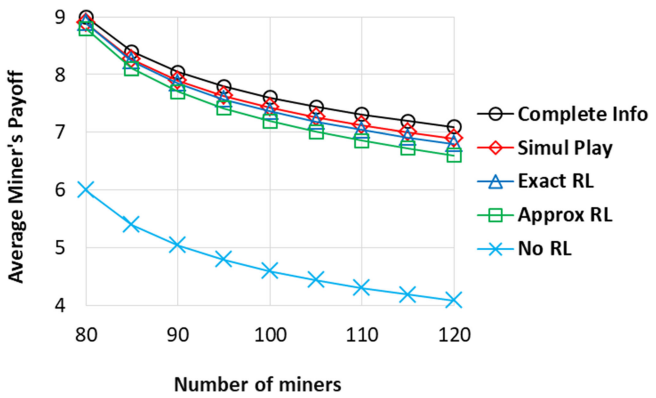

Fig. 8. Average miner's payoff as a function of the number of miners.

Observe that the convergence time increases linearly with the number of miners. Note that the algorithm for the SP converges after the algorithm for the miners. This can be explained as follows. The payoff of the SP is defined by its ability to accurately predict miners' best responses. On the other hand, the best responses depend on the current miners' beliefs about unobservable actions. Thus, when the beliefs are fluctuating, i.e., the RL process for miners has not reached its stable state, the SP's prediction accuracy is low and, consequently, its payoff is far from the optimal one. However, when the beliefs are stable, the SP's prediction accuracy increases, and its payoffs approaches the optimal one. Therefore, the algorithm for the SP converges after the algorithm for the miners.

Figs. 6 and 7 demonstrate the convergence time in a dynamic scenario in which the number of miners N can randomly change at any mining stage. In particular, Fig. 6 shows the results in the case in which the miners' mean arrival rate per BS varies from $\lambda = 2$ to $\lambda = 22$ miners/hour and the miners' mean departure rate per BS is fixed to $\mu = 2$ miners/hour. Fig. 7 shows the results in the case when the miners' mean departure rate per BS varies from $\mu = 2$ to $\mu = 22$ miners/hour and the miners' mean arrival rate per BS is fixed to $\lambda = 22$ miners/hour. Thus, in both cases, we have $\lambda \geq \mu$. As such, the expected number of miners is growing at each stage. Note that by increasing λ or decreasing μ, we increase not only the expectation of the number of miners but also its standard deviation. That is, deviations in the number of miners are higher for the larger values of λ or smaller values of μ. Indeed, from Figs. 6 and 7, the convergence time increases with λ and decreases with μ. However, such a growth rate is nondeterministic, i.e., has a non-zero standard deviation that increases with the ratio λ/μ. Furthermore, observe that unlike Fig. 5 showing a linear dependency of the convergence time from the number of miners, in Figs. 6 and 7, this dependency is concave. Hence, the added uncertainty has a positive impact on the convergence time of RL algorithms. The reason is that additional randomness increases the exploration abilities of learners (i.e., the SP and miners). As such, the learners are forced to select unexplored options that can result in the better payoffs compared to those that have been already determined as the optimal ones, which improves the performance of RL.

Figs. 8 and 9 show the impact of the number of BSs M and the associated number of miners N on the average payoffs of a miner and the SP, respectively, in different models after convergence. The results in these figures are collected for the number of BSs varying from $M = 3$ to $M = 7$ and the associated number of miners varying from $N = 80$ to $N = 120$. Figs. 10, 12, 13 demonstrate the average payoffs of a miner and the SP, respectively, in a dynamic and random scenario. In particular, Figs. 10 and 11 show the results in the case in which the miners' mean arrival rate per BS varies from $\lambda = 2$ to $\lambda = 22$ miners/ hour and the miners' mean departure rate per BS is fixed to $\mu = 2$ miners/hour. Figs. 12 and 13 show the results in the case when the mean departure rate per BS varies from $\mu = 2$ to $\mu = 22$ miners/hour and the mean arrival rate per BS is fixed to $\lambda = 22$ miners/hour. From the figures, the

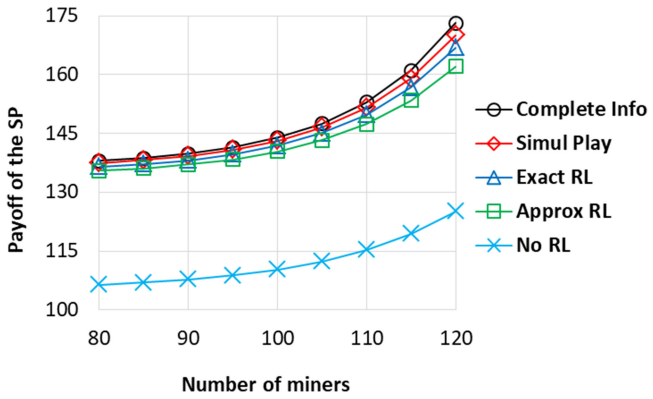

Fig. 9. Payoff of the SP as a function of the number of miners.

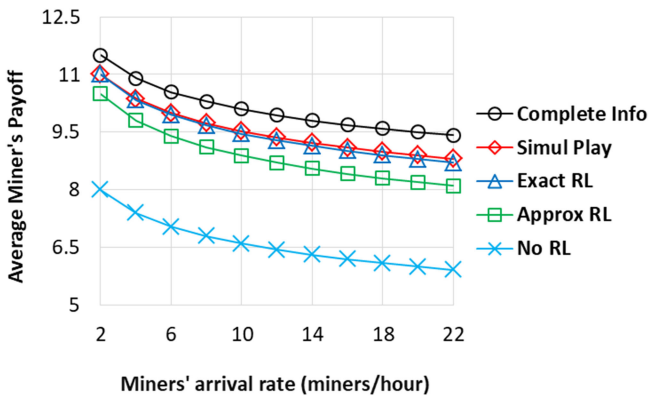

Fig. 10. Average miner's payoff as a function of miners' mean arrival rate per BS.

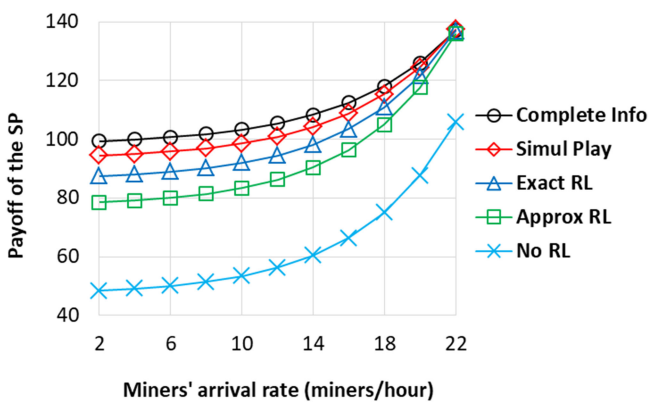

Fig. 11. Payoff of the SP as a function of miners' mean arrival rate per BS.

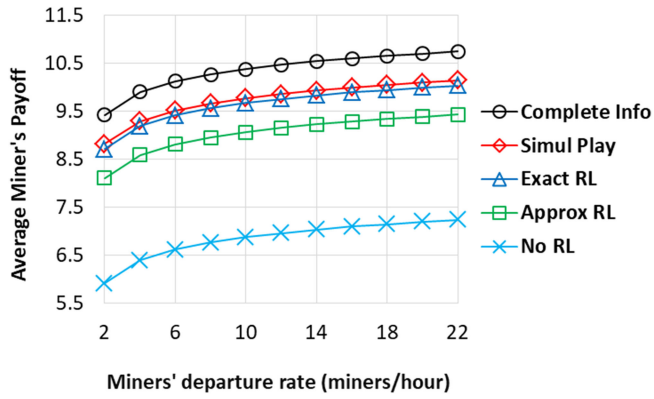

Fig. 12. Average miner's payoff as a function of miners' mean departure rate per BS.

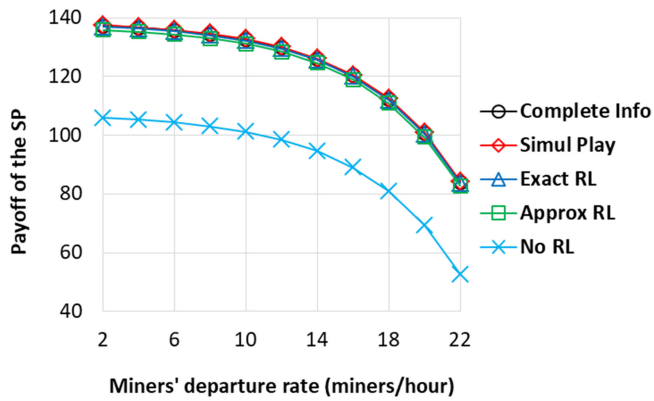

Fig. 13. Payoff of the SP as a function of miners' mean departure rate per BS.

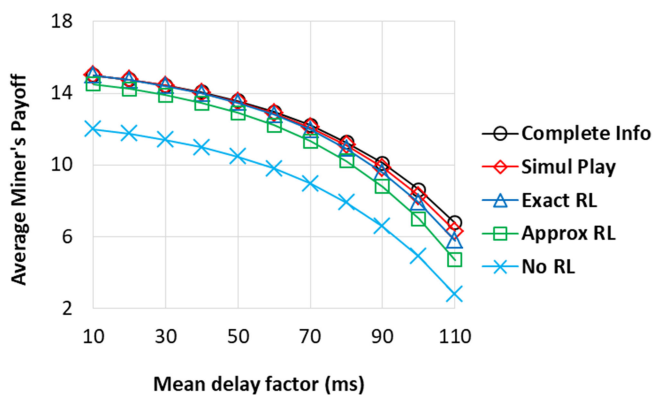

Fig. 14. Average miner's payoff depending on the mean delay factor.

average miner's payoff decreases with N and λ/μ. On the contrary, the SP's payoff increases with N and λ/μ. The reason is that when N and λ/μ are increasing, the chances of winning a mining reward are reducing, which means that there is a growing number of miners that pay for MEC services without receiving a reward. This results in the decreasing miners' payoffs and increasing SP's payoff. Furthermore, observe that the uncertainty added in a dynamic scenario evaluated in Figs. 10, 12, 13 has a positive impact on the algorithms' performance because of the reason mentioned before. That is, the additional randomness increases the exploration abilities of learners and, hence, may result in the increased payoffs of the miners and the SP.

Figs. 14 and 15 show the impact of the mean delay factor, i.e., network delay, on the average miner's payoff and payoff of the SP, respectively. The figures present the results of simulations in which the delay factor is modeled according to the exponential distribution (as in [48]) with the fixed expected value varying from 10 to 110 ms. Observe the average miner's payoff decreases exponentially with the mean delay factor. The reason is that growing network delays increase the orphaning probability (see equation (3)). On the other hand, when the orphaning probability increases, the probability of winning a mining reward reduces (see equation (1)). Thus, given the same hash rates, miners' chances to win a reward reduce when the network delay increases. As

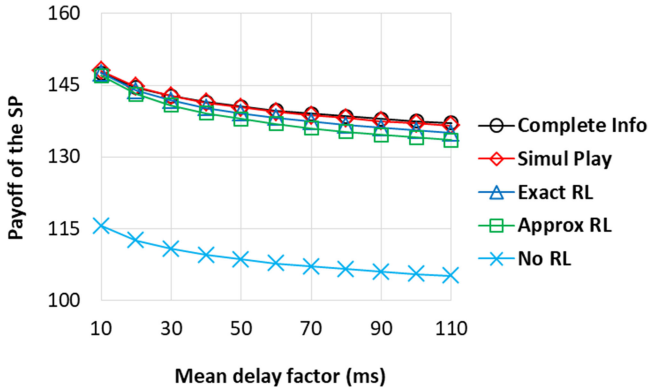

Fig. 15. Payoff of the SP depending on the mean delay factor.

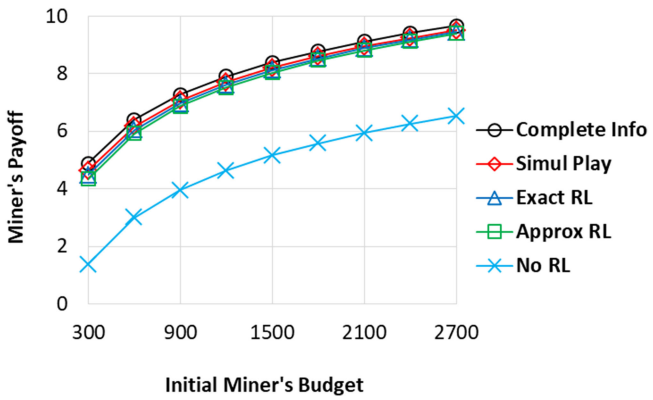

Fig. 16. Miner's payoff depending on the initial miner's budget.

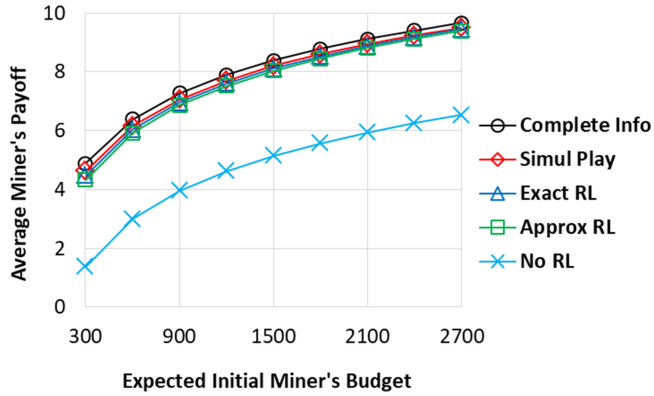

Fig. 17. Average miner's payoff depending on the expected miner's budget.

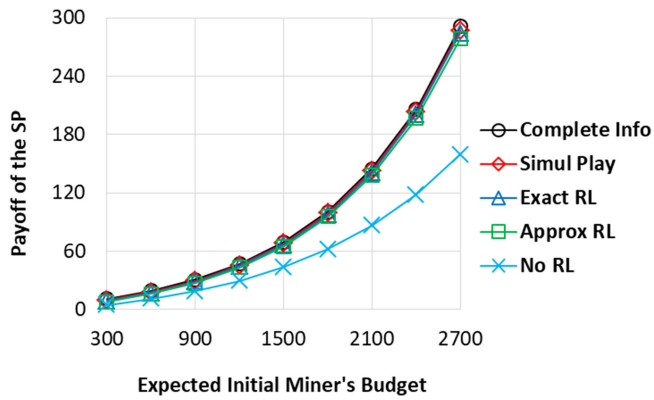

Fig. 18. Payoff of the SP depending on the expected miner's budget.

such, there is a growing number of miners that pay for MEC services without receiving a reward. This leads to decreased miners' payoffs. On the other hand, the impact of the network delay on the SP's payoff is smaller than that for the miners. In particular, the average SP's payoff decreases logarithmically with the mean delay factor. The reason is that the delay do not affect the SP's payoff directly, since miners pay for their hash rates, and not the block throughputs. Yet, the SP's payoff is affected by the delay factor indirectly. To see why, recall that the miners have limited budgets. That is, the SP can only assign such a price which is affordable by the miners. However, when the miner's chances of winning a reward reduce, its expected budget also reduces. As a result, the SP must assign a lower price to miners to assure that they can afford to pay such a price. This leads to a decreased SP's payoff.

The impact of the initial miner's budget on its payoff in different models after convergence is shown in Fig. 16. Observe that the miner's payoff increases with its initial budget. This is due to the fact that when the budget is small, the miner cannot afford to buy a higher hash rate, which decreases its winning probability and results in a decreased payoff. Figs. 17 and 18 present the average miner's payoffs and payoff of the SP, respectively, depending on the expected (rather than deterministic) initial miner's budget after convergence. Observe the average miner's payoff decreases, whereas, the average payoff of the SP increases with the expected budget. Such results can be explained as follows. When the expected initial miner's budget is low, the SP assigns a lower price per ECU which reduces the SP's payoff but increases the payoffs of miners. To summarize, all presented simulation results show that after convergence, the proposed algorithms exhibit the performance close to that achieved in the optimal Complete Info model.

7 Conclusion

We have proposed a novel hierarchical learning framework for a stochastic Stackelberg game under private information. The game models the interactions between the SP and miners in a public blockchain network implemented in the mobile edge computing system. Within the framework, we have developed the exact and approximate reinforcement learning algorithms based on the fully- and partially-observable MDPs for the decision making of the SP and miners. We have shown that the proposed exact and approximate learning procedures converge to stable states where the miners' hash rate decisions are the best responses to the optimal price of edge computing services.

The proposed framework can be easily extended to include multiple SPs, i.e., when the miners are free to select their associated SPs. In this case, each SP (i.e., the leader) assigns the price per unit hash rate by considering not only budget constraints of its associated miners (i.e., the followers) but also the fact that if the assigned price is too high, its followers can switch to another SP (which will reduce its payoff). As such, the long-term payoff of the SP will also depend on the price assignments (i.e., actions) of other SPs. That is, not only the miners compete against each other, but also the SPs. Then, the stochastic state in the MDP of the SP will include the actions of other SPs in addition to actions,

rewards, and initial budgets of miners. On the other hand, the actions in partially-observable MDPs of the miners will be represented by the miners' hash rates and SP selections.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China Project No. 61950410603, Singapore NRF National Satellite of Excellence, Design Science and Technology for Secure Critical Infrastructure NSoE DeST-SCI2019-0007, A*STAR-NTU-SUTD Joint Research Grant Call on Artificial Intelligence for the Future of Manufacturing RGANS1906, WASP/NTU M4082187 (4080), Singapore MOE Tier 1 2017-T1-002-007 RG122/17, MOE Tier 2 MOE2014-T2-2-015 ARC4/15, Singapore NRF2015-NRF-ISF001-2277, and Singapore EMA Energy Resilience NRF2017EWT-EP003-041.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008, [Online]. Available: http://bitcoin.org/bitcoin.pdf.
[2] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.
[3] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 2084–2123, 3Q 2016.
[4] K. Yeow, A. Gani, R. W. Ahmad, J. J. P. C. Rodrigues, and K. Ko, "Decentralized consensus for edge-centric internet of things: A review, taxonomy, and research issues," *IEEE Access*, vol. 6, pp. 1513–1524, 2018.
[5] W. Wang *et al.*, "A survey on consensus mechanisms and mining management in blockchain networks," 2018, *arXiv:1805.02707*.
[6] N. Herbaut and N. Negru, "A model for collaborative blockchain-based video delivery relying on advanced network services chains," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 70–76, Sep. 2017.
[7] J. Kang *et al.*, "Enabling localized peer-to-peer electricity trading among plug-in hybrid electric vehicles using consortium blockchains," *IEEE Trans. Ind. Informat.*, vol. 13, no. 6, pp. 3154–3164, Dec. 2017.
[8] Z. Xiong *et al.*, "When mobile blockchain meets edge computing,", 2017, *arXiv:1711.05938*.
[9] Z. Xiong *et al.*, "Cloud/fog computing resource management and pricing for blockchain networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4585–4600, Jun. 2019.
[10] A. Asheralieva, "Optimal computational offloading and content caching in wireless heterogeneous mobile edge computing systems with hopfield neural networks," *IEEE Trans. Emerging Topics Comput. Intell.*, to be published, doi: 10.1109/TETCI.2019.2892733.
[11] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.
[12] R. Yang, F. R. Yu, P. Si, Z. Yang, and Y. Zhang, "Integrated Blockchain and edge computing systems: A survey, some research issues and challenges," *IEEE Commun. Surveys. Tuts.*, vol. 21, no. 2, pp. 1508–1532, 2Q 2019.
[13] N. Houy, "The Bitcoin mining game," *Ledger*, vol. 1, pp. 53–68, Dec. 2016, Accessed: Dec. 24, 2019. [Online]. Available: https://ledgerjournal.org/ojs/index.php/ledger/article/view/13
[14] N. Fotiou and G. C. Polyzos, "Decentralized name-based security for content distribution using blockchains," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2016, pp. 415–420.
[15] T. Jin *et al.*, "Blockndn: A bitcoin blockchain decentralized system over named data networking," in *Proc. IEEE Int. Conf. Ubiquitous Future Netw.*, 2017, pp. 75–80.
[16] N. Herbaut and N. Negru, "A model for collaborative blockchain-based video delivery relying on advanced network services chains," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 70–76, Sep. 2017.
[17] W. Wang *et al.*, "Decentralized caching for content delivery based on blockchain: A game theoretic perspective," in *Proc. IEEE Int. Conf. Commun.*, 2018, pp. 1–6.
[18] K. Kotobi and S. G. Bilén, "Blockchain-enabled spectrum access in cognitive radio networks," in *Proc. IEEE Wireless Telecommun. Symp.*, 2017, pp. 1–6.
[19] A. Lei, H. Cruickshank, Y. Cao, P. Asuquo, C. P. A. Ogah, and Z. Sun, "Blockchain-based dynamic key management for heterogeneous intelligent transportation systems," *IEEE Internet Things J.* vol. 4, no. 6, pp. 1832–1843, Dec. 2017.
[20] J. Kang, Z. Xiong, D, Niyato, D. Ye, D. I. Kim, and J. Zhao, "Toward secure blockchain-enabled internet of vehicles: Optimizing consensus management using reputation and contract theory," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2906–2920, Mar. 2019.
[21] W. Wang *et al.*, "A survey on consensus mechanisms and mining strategy management in blockchain networks," in *IEEE Access*, vol. 7, pp. 22328–22370, Jan. 2019.
[22] Y. Jiao *et al.*, "Social welfare maximization auction in edge computing resource allocation for mobile blockchain," in *Proc. IEEE Int. Conf. Commun.*, 2018, pp. 1–6.
[23] N. C. Luong *et al.*, "Optimal auction for edge computing resource management in mobile blockchain networks: A deep learning approach," in *Proc. IEEE Int. Conf. Commun.*, 2018, pp. 1–6.
[24] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *Proc. IEEE P2P*, 2013, pp. 1–10.
[25] S. V. Albrecht, J. W. Crandall, and S. Ramamoorthy, "Belief and truth in hypothesised behaviours," *Artif. Intell.*, vol. 235, no. 2016, pp. 63–94, 2016.
[26] A. Asheralieva, "Bayesian reinforcement learning-based coalition formation for distributed resource sharing by device-to-device users in heterogeneous cellular networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 5016–5032, Aug. 2017.
[27] A. Asheralieva and D. Niyato, "Hierarchical game-theoretic and reinforcement learning framework for computational offloading in UAV-enabled mobile edge computing networks with multiple service providers," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8753–8769, Oct. 2019.
[28] M. Ghavamzadeh *et al.*, "Bayesian reinforcement learning: A survey," *Foundations Trends Mach. Learn.*, vol. 8, no. 5-6, 2016, pp. 359–483.
[29] A. K. Dixit *et al.*, *Investment Under Uncertainty*, Princeton, NJ, USA: Princeton university press, 1994.
[30] R. C. Merton, "An intertemporal capital asset pricing model," *Econometrica: J. Econometric Soc.*, 1973, pp. 867–887.
[31] M. J. Osborne and A. Rubenstein. *A Course in Game Theory*. Cambridge, MA, USA: MIT Press, 1994.
[32] S. Sorin, "Stochastic games with incomplete information," *Stochastic Games and Applications*. Berlin, Germany: Springer, 2003, pp. 375–395.
[33] D. Rosenberg, E. Solan, and N. Vieille, "Stochastic games with a single controller and incomplete information," *SIAM J. Control Optim.*, vol. 43, no. 1, pp. 86-1–10, 2004.
[34] P. Poupart *et al.*, "An analytic solution to discrete Bayesian reinforcement learning," in *Proc. ACM Int. Conf. Mach. Learn.*, 2006, pp. 697–704.
[35] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.
[36] M. Geist and O. Pietquin, "A brief survey of parametric value function approximation," *Rapport interne*, Supélec, Sep. 2010.
[37] M. Irodova and R. H. Sloan, "Reinforcement learning and function approximation," in *Proc. FLAIRS Conference*, 2005, pp. 455–460.
[38] F. S. Melo, S. P. Meyn, and M. I. Ribeiro, "An analysis of reinforcement learning with function approximation," in *Proc. ACM Int. Conf. Mach. learn.*, 2008, pp. 664–671.
[39] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, 2015.
[40] L. Deng and D. Yu, "Deep learning: Methods and applications," *Foundactions Trends Signal Process.*, vol. 7, no. 3–4, pp 197–387, 2014.
[41] S. Gu *et al.*, "Continuous deep q-learning with model-based acceleration," in *Proc. ACM Int. Conf. Mach. Learn.*, Jun. 2011, pp. 2829–2838.
[42] T. Hester *et al.*, "Deep q-learning from demonstrations," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 3223–3230.
[43] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.
[44] OPNET simulation and development tool, Accessed: Dec. 24, 2019. [Online]. Available: http://www.opnet.com.
[45] Evolved universal terrestrial radio access (E-UTRA) and evolved universal terrestrial radio access network (E-UTRAN); overall description; Stage 2, *3GPP TS 36.300 (Release 13)*, 2016.
[46] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Commun. Mobile Comput.*, vol. 2, no. 5, pp. 1–27, Sep. 2002.

[47] T. Imai, "Essays in revealed preference theory and behavioral economics," PhD diss., Division Humanities Social Sci., CA Inst. Technol., 2016.
[48] G. Fanti *et al.*, "Barracuda: The power of ℓ-polling in proof-of-stake blockchains," in *Proc. ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2019, pp. 351–360.
[49] Z. Han *et al.*, *Game Theory in Wireless and Communication Networks: Theory, Models, and Applications*. Cambridge, UK: Cambridge University Press, 2012.
[50] M. Schatzman, *Numerical Analysis: A Mathematical Introduction*. 1st ed. Oxford, UK: Oxford University Press, 2002.

Alia Asheralieva received the BS degree from Kyrgyz Technical University, Bishkek, Kyrgyzstan, in 2004, the ME degree from the Asian Institute of Technology, Bangkok, Thailand, in 2007, and the PhD degree from the University of Newcastle, Callaghan NSW, Australia, in 2014. In 2015 and 2016, she was a research assistant professor in the Graduate School of Information Science and Technology at Hokkaido University, Sapporo, Japan. From 2017, she was a postdoctoral research fellow in the Wireless Networks and Decision Systems Group of the Information Systems Technology and Design Pillar, Singapore University of Technology and Design. She is currently an assistant professor with the Department of Computer Science and Engineering of the Southern University of Science and Technology in Shenzhen, China. Her main research interests include many areas of communications and networking, including cognitive radio networks, heterogeneous networks, device-to-device and Internet of Things communications, cloud/edge/fog computing, cross-layer resource allocation and optimization, congestion control and routing, game theory, computational and artificial intelligence for wireless networks, queuing theory, simulation and network modelling, QoS, and performance evaluation.

Dusit Niyato (M'09–SM'15–F'17) received the BEng degree from the King Mongkut's Institute of Technology Ladkrabang, Thailand, in 1999, and the PhD degree in electrical and computer engineering from the University of Manitoba, Canada, in 2008. He is currently a professor with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His research interests include energy harvesting for wireless communication, Internet of Things, and sensor networks.