Bayesian Reinforcement Learning and Bayesian Deep Learning for Blockchains With Mobile Edge Computing

Alia Asheralieva and Dusit Niyato, *Fellow, IEEE*

***Abstract*—We present a novel game-theoretic, Bayesian reinforce-ment learning (RL) and deep learning (DL) framework to represent interactions of miners in public and consortium blockchains with mobile edge computing (MEC). Within the framework, we formulate a stochastic game played by miners under incomplete information. Each miner can offload its block operations to one of the base stations (BSs) equipped with the MEC server. The miners select their offloading BSs and block processing rates simultaneously and independently, without informing other miners about their actions. As such, no miner knows the past and current actions of others and, hence, constructs its belief about these actions. Accordingly, we devise a Bayesian RL algorithm based on the partially-observable Markov decision process for miner's decision making that allows each miner to dynamically adjust its strategy and update its beliefs through repeated interactions with each other and with the mobile environment. We also propose a novel unsupervised Bayesian deep learning algorithm where the uncertainties about unobservable states are approximated with Bayesian neural networks. We show that the proposed Bayesian RL and DL algorithms converge to the stable states where the miners' actions and beliefs form the perfect Bayesian equilibrium (PBE) and myopic PBE, respectively.**

***Index Terms*—Bayesian methods, blockchains, deep learning, game theory, incomplete information, machine learning, mobile edge computing, partially-observable Markov decision process, reinforcement learning, resource management.**

Manuscript received July 21, 2019; revised January 18, 2020; accepted May 4, 2020. Date of publication May 14, 2020; date of current version March 8, 2021. This work was supported in part by the National Natural Science Foundation of China (NSFC) Project No. 61950410603, National Research Foundation (NRF), Singapore, under Singapore Energy Market Authority (EMA), Energy Resilience, NRF2017EWT-EP003-041, Singapore NRF2015-NRF-ISF001-2277, 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 on Artificial Intelligence for the Future of Manufacturing RGANS1906, WASP/NTU M4082187 (4080), Singapore MOE Tier 2 MOE2014-T2-2-015 ARC4/15, and MOE Tier 1 2017-T1-002-007 RG122/17. The associate editor coordinating the review of This article and approving it for publication was K. Zeng. *(Corresponding author: Alia Asheralieva.)*

Alia Asheralieva is with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China (e-mail: aasheralieva@gmail.com).

Dusit Niyato is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798 (e-mail: dniyato@ntu.edu.sg).

This paper has supplementary downloadable material available at https://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/TCCN.2020.2994366

I. Introduction

BLOCKCHAIN is a distributed ledger technology in which data is processed and stored with in-build robustness, as it cannot be controlled by a single entity and is able to avoid a single failure point. Public blockchains are also characterized by their open access, transparency and disintermediation [1]–[3]. In the blockchain, data is organized in the form of blocks, e.g., transaction records, which represent a linked list structure and preserve logical relations in the stored data. The copies of blocks are distributed across an entire blockchain network that comprises all blockchain users. This guarantees improved data integrity and security comparing to centralized ledgers. The process by which the blocks are processed, verified and added to a blockchain is referred to as a mining [1]. During mining, a blockchain user or miner executes a compute-intensive task, e.g., block processing or validation. The block transaction is confirmed and stored in a blockchain only if its output reaches a consensus that can be based on such protocols as proof-of-work (PoW), proof-of-stake (PoS), or practical Byzantine fault tolerance (pBFT) [4], [5].

Although blockchains have already been adopted in certain distributed system scenarios, e.g., content delivery networks [6] and smart-grids [7], they are still not widely utilized in the Internet of Things (IoT) and other mobile systems. The main reason is that the users' mobile terminals and IoT devices have scarce computing resources. Thus, they cannot perform many compute-intensive block tasks. To deal with this issue, MEC concept has been proposed to facilitate future IoT and mobile applications [3], [8], [9]. In the MEC, virtual cloud computing capabilities are extended at the network "edges" through MEC servers installed at BSs, so that the end-users can offload their tasks via the BSs in proximity [10]–[12]. Compared to cloud computing where all data are transmitted to, and processed at a centralized cloud (far from end-users) which leads to heavy backhaul load, high propagation delays, and increased chances of being manipulated on a way to the cloud, MEC can provide reduced backhaul traffic, congestion, energy consumption and latency, and improved security, as data are managed locally [3], [12]. Nevertheless, to allow a successful realization of the blockchain with MEC, it is important to study the profitability and practicality of the system from the perspective of its users, i.e., miners.

In particular, due to its nature, blockchain system operates in a distributed manner – every miner makes an independent decision on how to handle its block task aiming to maximize its individual payoff. The payoff is the difference between the reward that the miner receives for its task and the cost of computing power spent on running the task. The probability that the miner wins a reward depends on: i) computing powers of other miners; ii) orphaning probability for the miner's task, i.e., probability that the task is discarded due to long latency. Accordingly, to make a decision that maximizes it payoff, the miner must know exact computing powers of other miners and the orphaning probability for its task, which is unrealistic in blockchains with MEC due to the following critical issues:

- In the MEC, information exchange among miners leads to additional signaling that increases delay/energy overheads, especially in public and consortium blockchains where the number of miners and, hence, amount of communications is very large. As such, miners must decide on their computing power simultaneously and independently, without revealing their decisions to other miners [1], [3], [13]. Therefore, the computing power selected by a miner represents its private information unobservable by other miners.
- In the MEC, the orphaning probability for the miner's task is inversely proportional to the rate of a channel between the miner and its associated BS [3], [14], which depends on stochastic parameters of wireless environment (e.g., miners' locations, channel quality and co-channel interference). As a result, to estimate the orphaning probability, the miner must be able to predict the environmental parameters and adjust its task-handling decision in response to their changes.

Consequently, in this paper, we formulate the novel game-theoretic, Bayesian RL and DL framework which enable us to address the above critical issues in order to allow all miners to decide how to handle their tasks more efficiently, i.e., with maximal payoffs, and, thus, ensure the system practicality and profitability of the blockchain with MEC from the perspective of miners. We use a non-cooperative game theory, because it is the most suitable tool to model the interactions of miners – it reflects a distributed blockchain nature by assuming that all miners act as independent players aiming to maximize their individual (rather than socially-optimal) payoffs. On the other hand, by adopting learning, miners can predict changes in the random and unobservable parameters via repeated interactions with the other miners and with the environment. The main contributions of the paper are as follows:

1) We present a new system model of public and consortium blockchain applications in the MEC network formed by the set of BSs, each of which is equipped with the MEC server. Unlike prior blockchain systems (e.g., [8], [9]), the model takes into account the dynamic MEC features. In the model, each miner can select any BS in its transmission range to offload a block task and decide on its block processing rate (BPR), i.e., computing power, at a selected BS. Since the miner is rational, it aims at maximizing its expected payoff, i.e., difference between its expected reward and cost of BPR at the offloading BS.
2) We show that the interactions of miners can be described by the stochastic game with incomplete information. The game is stochastic, because the payoffs of its players/miners depend on the random environmental state with unknown state transitions. Moreover, since no miner can observe the task-handling decisions or actions of other miners, the game is played under incomplete information. To model the miner's decision process, we adopt a partially-observable Markov decision process (POMDP), the solution of which represents a perfect Bayesian equilibrium (PBE) where the strategies of the miners maximize their expected long-term payoffs. Based on the POMDP, we develop the "on-line" Bayesian RL (BRL) algorithm that enables all the miners to update their beliefs about unobservable actions and reach a PBE.
3) To reduce exponential complexity of a BRL algorithm, we formulate a novel Bayesian DL (BDL) algorithm of the polynomial complexity, in which the uncertainties about unobservable states of POMDP are modeled by a Bayesian neural network (BNN) – neural network (NN) with random weights distributed according to a predefined prior. The BDL algorithm is unsupervised – it enables a self-organized on-line learning that does not require any prior "off-line" training and/or pre-existing training datasets. We prove that the BDL algorithm converges to the state where the miners' strategies form a myopic PBE (MPBE) – a PBE with short-sighted players aiming to maximize expected stage payoffs.

The rest of the paper is organized as follows. In Section II, we review related research on mobile blockchains. In Section III, we present the system model of a blockchain with MEC. In Section IV, we define a stochastic game and design a BRL algorithm for the miners. In Section V, we formulate a BDL algorithm. In Section VI, we evaluate the performance of the proposed framework. The list of acronyms used in the paper in the alphabetical order is provided in Appendix A in the supplementary material.

II. Related Work

The existing works on mobile blockchains can be arbitrarily classified into three groups: i) blockchain-based protocols for mobile and IoT applications (e.g., [14]–[27]); ii) consensus algorithms for mobile blockchains (e.g., [7], [28]–[35]); iii) pricing and resource management in mobile blockchains (e.g., [8], [9], [36]–[39]). The first largest group adopts the concept of blockchains in developing data communication and control protocols to support various mobile/IoT applications, e.g., spectrum sharing [15]–[19], network virtualization [20]–[22], or data and resource management [14], [23]–[27], by taking advantage of such useful blockchain features as traceability, decentralization, immutability and privacy. E.g., in cognitive radio networks, blockchains can address spectrum scarcity and increase utilization in a distributed, secure and transparent

way [15]–[19]. In this case, blockchains serve as a middle layer between primary and secondary spectrum users to provide system verification and access, and perform spectrum sharing and trading. In network virtualization, blockchains can be used to slice the virtual networks, so that the resource owners can sublease resources to virtual network operators [20]–[22]. The participants of each slice are managed by the respective block slice, which provides auditability, monitoring of operations and data access. Blockchains can also be applied for data/ resource management in mobile/IoT networks, in which case they act as distributed platforms for authentication and design of smart contracts [14], [25]–[27].

The second group (e.g., [7], [28]–[35]) explores consensus algorithms and protocols applicable to mobile/IoT blockchain systems. The challenge here is the scalability measured w.r.t. three metrics: i) throughput – number of transactions verified per time unit; ii) security – number of malicious peers the system can tolerate; iii) storage – size of blockchain that can be handled by the miners with limited storage capabilities. Scalability means that all metrics can improve or, at least, not deteriorate as the number of miners increases [4], [5]. Poor scalability is the biggest issue of current blockchain systems, e.g., Ethereum and Bitcoin, that are based on classical PoW and PoS protocols. Although these systems can tolerate up to 49% of malicious miners, they have low throughput and storage efficiency [28]–[30]. Thus, several alternatives, e.g., delegated PoS (DPoS), pBFT, sharding, and their variations [7], [28]–[35], have been proposed to increase throughput and storage, without compromising security.

The third group (e.g., [8], [9], [36]–[39]) study pricing and resource management in blockchains with MEC, where miners offload their tasks to the BSs equipped with MEC servers. The focus here is on the mining under PoW consensus which leads to a competition among miners to win a reward. Authors in [8], [9], [36] and [37] analyze the interplay between miners and MEC service providers (SPs), and propose the methods to maximize the miners' payoffs and the revenues of the SPs by adopting Stackelberg game [8], [9] and auction theory [36], [36]. Authors in [38] and [39] propose the optimization-based approaches for resource allocation, offloading and block size adaptation with the objective to maximize the average miner's reward. As such, the studies in [8], [9], [36]–[39] explore the profitability of blockchains with MEC for the miners and offer various mechanisms to maximize miners' payoffs. However, they have several limitations which are listed below:

- The works in [8], [9], [36]–[39] ignore dynamics of the mobile environment, e.g., time-varying locations of miners, co-channel interference and channel quality. In particular, games/actions in [8], [9], [36] and [37] are played in a static deterministic environment resulting in short-term solutions; optimization problems in [38] and [39] assume that all the wireless parameters remain fixed. Hence, the applicability of these works to practical blockchains operating in highly-dynamic MEC networks is unclear.
- The main assumption of works in [8], [9], [36]–[39] is the complete information about the miners' actions and other system parameters. This is rather unrealistic, since in PoW blockchains, miners take their actions simultaneously and independently, and compete against each other to win a reward [1], [3], [13]. Thus, the action of each miner is the private information of this miner unknown to other miners.

In summary, the framework proposed in this paper belongs to the third group of studies in [8], [9], [36]–[39]. Similar to these studies, we analyze the practicality of blockchains with MEC for the miners, but adopt more relevant to the mobile environment considerations:

1) The presented system model addresses the randomness of a wireless environment which is not considered in existing research. The formulated game model is stochastic, i.e., it is played in a dynamic environment and returns both the long-term and short-term solutions represented by the PBE and MPBE, respectively.
2) In our framework, information about the miners' actions is incomplete. To deal with incomplete information, we adopt the BRL and BDL algorithms which enable each miner to form and update its belief about unobservable actions of other miners. As such, unlike many existing learning-based methods which show the algorithm convergence to some fixed (but not always optimal) value [40], [41], we prove that our BRL and BDL algorithms converge to the PBE and MPBE, i.e., the long-term and short-term solutions of the miners' game, respectively.
3) Although our framework is originally developed for PoW protocols, it can be applied (with slight modifications) with other protocols, such as PoS or pBFT.

III. Analytical Model of a Blockchain With MEC

A. System Model

Consider a blockchain application implemented in the MEC network, as shown in Figure 1. The network includes a set $\mathbf{M} = \{1, \ldots, M\}$ of macro- and small-cell BSs labeled as $\text{BS}_1, \ldots, \text{BS}_M$, each of which is equipped with the MEC server, and a set $\mathbf{N} = \{M+1, \ldots, M+N\}$ of blockchain miners labeled as $\text{U}_{M+1}, \ldots, \text{U}_{M+N}$, for notation consistency. In the system, miners perform the range blockchain operations: i) process data in the form of blocks, e.g., transaction records; ii) verify generated blocks; iii) store verified blocks. The BSs provide offloading and communication services to the miners. That is, any miner $\text{U}_n, n \in \mathbf{N}$, can offload its compute-intensive block task, e.g., block processing or verification, from its mobile device to one of the BSs in its transmission range. We denote by $\mathbf{M}^n \subseteq \mathbf{M}$ the subset of BSs in the range of miner U_n. Every BS_m, $m \in \mathbf{M}$, operates on an orthogonal spectrum of the bandwidth B^m that may overlap with the spectra of other BSs. Without loss of generality, BSs can be privately-owned, or controlled by different SPs. Each BS sells its computing resources measured in terms of edge computing units (ECUs) to the miners. The computing resources of the BS are limited by the maximal number of ECUs supported by its MEC server. We denote by $\mathcal{X}^m$ and c^m the maximal number of supported ECUs and the cost per

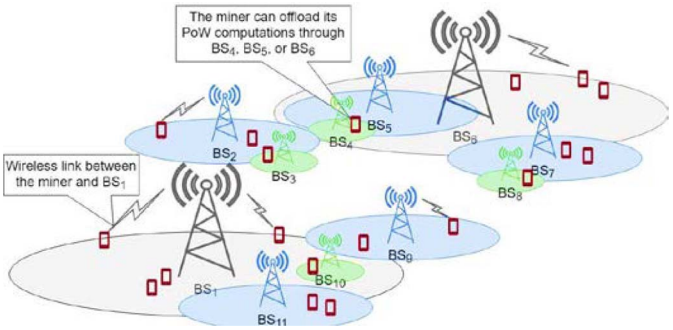

Fig. 1. Blockchain application in the MEC network with 11 BSs, macro-cell BS_1 and BS_6, micro-cell BS_2, BS_5, BS_7, BS_9 and BS_{11}, and femto-cell BS_3, BS_4, BS_8 and BS_{10}. Each miner can offload its compute-intensive task (e.g., block processing or verification) to any BS in its transmission range.

ECU at BS_m, respectively. The list of main notations used in the paper is provided in Appendix B in the supplementary material.

The process of blockchain mining consists of the number of stages, denoted as $t = 0, \ldots, T$, during which every subsequent block in the chain is processed. Within each stage, all network parameters (e.g., the numbers of miners and BSs, locations of miners, channel and computing resource allocations) remain constant. As a result, the connectivity of the MEC network is preserved during one mining stage, i.e., there are no handovers or access blockages for miners. However, the parameters can change as we head to the next stage. The BSs and miners can join the blockchain system at any mining stage and remain in the system for an indefinitely-long time (even if the duration of their stay in the system may be finite, it is unknown). That is, the total number of stages is $T \to \infty$. In order to be admitted into a blockchain system, a BS should register its operating bandwidth, the maximal number of supported ECUs, and the cost per its ECU. The miner should submit a digital signature for its identification in the system. If admitted during stage t, the BS and the miner are assigned with unique labels $m \notin \mathbf{M}$ and $n \notin \mathbf{N}$, and the sets of BSs and miners are updated as $\mathbf{M} \leftarrow \mathbf{M} \cup \{m\}$ and $\mathbf{N} \leftarrow \mathbf{N} \cup \{n\}$ at the beginning of the next mining stage $t+1$. Afterwards, each miner U_n can determine the subset $\mathbf{M}^n$ of BSs in its range by adopting conventional pilot signals [44], [45] and, then, starts participating in block mining. On the other hand, any BS_m and each miner U_n that are going to leave at the next stage $t+1$ must notify the system about their leave before the start of stage $t+1$ (note that neither the BSs and nor the miners can leave the system until the current stage t concludes). In this case, at the beginning of stage $t+1$, the sets of BSs and miners are updated as $\mathbf{M} \leftarrow \mathbf{M} \backslash \{m\}$ and $\mathbf{N} \leftarrow \mathbf{N} \backslash \{n\}$. As such, the sets $\mathbf{M}$ and $\mathbf{N}$ are fixed during one stage, but can change as we move to the next stage. The information about the sets $\mathbf{M}$ and $\mathbf{N}$ is available publicly, i.e., all BSs and all miners are aware about the updates in the sets $\mathbf{M}$ and $\mathbf{N}$.

B. Mining Process

Recall that during mining, each miner U_n must perform a compute-intensive block task which can be offloaded to one of the BSs in the transmission range of the miner. As such, at any stage t, the miner must select one BS to offload its task and the exact BPR with which the task is offloaded. Let $b_t^n \in \mathbf{M}^n$, such that $b_t^n \in \mathbb{R}$, where $\mathbb{R}$ is the one-dimensional real vector space, be a one-dimensional BS decision of miner U_n at stage t. Let $x_t^n \in \mathbf{X}^n$, such that $x_t^n \in \mathbb{R}$, be a one-dimensional BPR decision of miner U_n at stage t, where $\mathbf{X}^n = \{1, \ldots, X^n\}$ is a finite discrete set containing all possible BPR decisions of miner U_n, and X^n is the maximal BPR that can be selected by miner U_n. The maximal BPR is constrained by the maximal computing power of the miner's BS. As such, at any stage t, we must have

$$\sum_{n \in \mathbf{N}} \mathbf{1}_{b_t^n = m} X^n \leq \mathcal{X}^m, \ \forall m \in \mathbf{M}, \tag{1a}$$

where $\mathbf{1}_y = 1$, if y is true and 0, otherwise. For example, BS_m can restrict the maximal BPR of any miner U_n associated with this BS to stay within

$$X^n \leq \left\lfloor \sum_{m \in \mathbf{M}^n} \mathbf{1}_{b_t^n = m} \mathcal{X}^m \Big/ \left(\sum_{i \in \mathbf{N}} \mathbf{1}_{b_t^i = m} \right) \right\rfloor, \ \forall n \in \mathbf{N}, \tag{1b}$$

where $\lfloor \cdot \rfloor$ is the floor function. Upon deciding on its BS/BPR at stage t, miner U_n sends a connection request to the selected BS, which allocates a wireless channel of a certain bandwidth β^n to the miner. Then, miners start simultaneously mining the next block in the chain. During mining, the miner executes the following operations: 1) processes the block by offloading all related computations to the selected BS; 2) broadcasts the produced output to other miners via a selected BS; 3) verifies accuracy of outputs generated by other miners; 4) broadcasts verification results to other miners. Any miner which is the first to generate the output that reaches a consensus obtains a mining reward. Assuming that the PoW protocol is adopted, the consensus on the generated output is reached only if all miners agree that the output is correct [4], [5]. However, other options are also possible. For example, the consensus can be reached based on weighted voting in which the weight of the miner's vote is proportional to its stake submitted by the miner upon its registration in the system, as in the PoS protocol [4], [5]. Alternatively, only some selected group of miners can be allowed to validate the output. The group can be selected from the miners with the highest stakes, as in the DPoS protocol, or from the miners with the highest reputations, as in the pBFT protocol [5].

As such, the reward r_t^n of miner U_n resulting from its BS/BPR decision (b_t^n, x_t^n) at stage t is defined by

$$r_t^n = \begin{cases} R + r\tau^n > 0, & \text{output is the first to reach consensus,} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

That is, a positive reward $r_t^n = R + r\tau^n > 0$ is obtained only if output generated by the miner is the first to reach consensus. Otherwise, $r_t^n = 0$. As such, we have $r_t^n \in \mathbf{R}^n$, where $\mathbf{R}^n = \{0,\ R + r\tau^n\}$ is a finite discrete set of possible rewards of the miner. A positive reward $R + r\tau^n$ includes a fixed part $R > 0$ and a variable part $r\tau^n$. A variable part is the product of some given reward factor $r > 0$ and a block size τ^n, i.e., the number

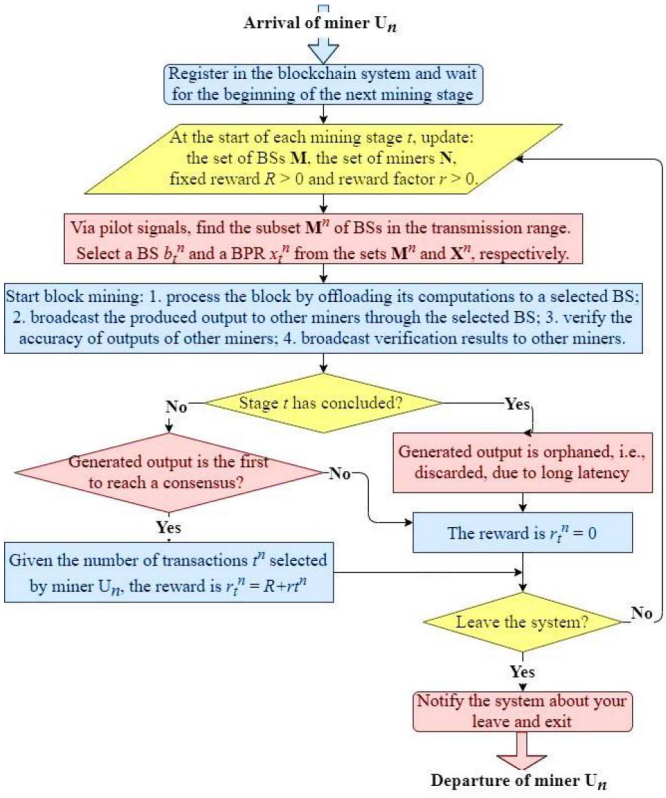

Fig. 2. Flowchart of the block mining performed by each miner U_n.

of transactions per block selected by miner U_n. The block size does not affect the complexity or the cost of its mining. Nonetheless, the chances of winning a reward decrease when the block propagates more slowly. That is, if the size τ^n is too large, the block is likely to be discarded due to long latency, which is called orphaning [3], [13]. After the winner of stage t is determined, a winning block is appended to the blockchain. At this moment, stage t of the mining process concludes and the next stage $t+1$ begins. A graphical illustration of block mining performed by miner U_n and the respective reward r_t^n is presented in Figure 2, where block mining process is shown in the form of a flowchart.

C. Winning Probabilities and Delay Factors for Miners

Let $\mathbf{b}_t^{-n} = \{b_t^i\}_{i\in\mathbf{N}\setminus\{n\}} \in \mathbf{M}^{-n}$ and $\mathbf{x}_t^{-n} = \{x_t^i\}_{i\in\mathbf{N}\setminus\{n\}} \in \mathbf{X}^{-n}$ denote the BSs and BPRs, respectively, selected by all miners except U_n. The sets $\mathbf{M}^{-n} = \times_{i\in\mathbf{N}\setminus\{n\}}\mathbf{M}^i$ and $\mathbf{X}^{-n} = \times_{i\in\mathbf{N}\setminus\{n\}}\mathbf{X}^i$ contain all possible BS and BPR decisions, respectively, of all miners except U_n. Then, the probability of a reward $r^n \in \mathbf{R}^n$ for miner U_n given its own BS/BPR decision $(b_t^n, x_t^n) \in \mathbf{M}^n \times \mathbf{X}^n$ and a decision profile $(\mathbf{b}_t^{-n}, \mathbf{x}_t^{-n}) \in \mathbf{M}^{-n} \times \mathbf{X}^{-n}$ of other miners is defined by

$$\Pr\left\{r^n \middle| b_t^n, x_t^n, \mathbf{b}_t^{-n}, \mathbf{x}_t^{-n}\right\} = \begin{cases} P_s^n\left(x_t^n, \mathbf{x}_t^{-n}\right)\left(1-P_o^n\left(b_t^n, \mathbf{b}_t^{-n}\right)\right), & r^n = R+r\tau^n, \\ 1-P_s^n\left(x_t^n, \mathbf{x}_t^{-n}\right)\left(1-P_o^n\left(b_t^n, \mathbf{b}_t^{-n}\right)\right), & r^n = 0, \end{cases} \tag{3a}$$

where P_s^n is the probability that miner U_n is successful in producing the output first, which is proportional to the miner's relative BPR, i.e.,

$$P_s^n\left(x_t^n, \mathbf{x}_t^{-n}\right) = x_t^n \Big/ \left(\sum_{i\in\mathbf{N}} x_t^i\right); \tag{3b}$$

P_o^n is the orphaning probability which depends on the latency in the wireless channel between miner U_n and its selected BS. Similar to [3], [9], [13], we can assume that the occurrence of executing any block task is a random variable following a Poisson distribution with the mean of $1/T_b$, where T_b is the expected block interval time (in the PoW protocol, $T_b = 600$ seconds). Then, the probability P_o^n can be approximated as [3], [9], [13]

$$P_o^n\left(b_t^n, \mathbf{b}_t^{-n}\right) = 1 - \exp\left(-\tau^n z^n\left(b_t^n, \mathbf{b}_t^{-n}\right)/T_b\right), \tag{3c}$$

where $z^n > 0$ is the delay factor, i.e., transmission time per transaction of miner U_n which reflects time/space-dependent features of a wireless channel between the miner and its BS.

As such, the delay factor is inversely proportional to the transmission rate of a wireless channel allocated to miner U_n by its selected BS. That is,

$$z^n\left(b_t^n, \mathbf{b}_t^{-n}\right) = z_t^n = \frac{1/\beta^n\left(b_t^n, \mathbf{b}_t^{-n}\right)}{\log_2\left(1+SINR^n\left(b_t^n\right)\right)}, \tag{4a}$$

where $SINR^n$ is signal-to-interference-plus-noise ratio (SINR) over the channel. Assuming that spectrum resources allocated by the BS are distributed equally among miners offloading their block tasks, the bandwidth β^n of a channel allocated to miner U_n is given by

$$\beta^n\left(b_t^n, \mathbf{b}_t^{-n}\right) = \sum_{m\in\mathbf{M}^n} \mathbf{1}_{b_t^n=m}\left(B^m \Big/ \sum_{i\in\mathbf{N}} \mathbf{1}_{b_t^i=m}\right). \tag{4b}$$

Note that since no miner U_n knows the BS decisions of other miners, it cannot estimate the values of β^n directly according to (4b). On the other hand, $SINR^n$ is given by

$$SINR^n(b_t^n) = \sum_{m\in\mathbf{M}^n} \frac{\mathbf{1}_{b_t^n=m} p^n G^{n,m}}{\sum_{i\in\mathbf{M}\setminus\{m\}} o^{i,m}\tilde{p}^i\tilde{G}^{i,m}+\sigma^2}, \tag{4c}$$

where $o^{i,m} \in \{0,\ 1\}$ is the binary bandwidth overlap indicator, such that $o^{i,m} = 1$, if the spectrum band of BS_m overlaps with the spectrum band of BS_i; p^n and $G^{n,m}$ are the transmit power of miner U_n and link gain of wireless channel between miner U_n and BS_m, respectively; $\tilde{p}^i$ and $\tilde{G}^{i,m}$ are the average transmit power of miners associated with BS_i and average link gain of wireless channels between the miners associated with BS_i and BS_m, respectively; σ^2 is the variance of a zero-mean additive white Gaussian noise (AWGN) power. Note that equation in (3a) presumes the interference among miners' transmissions, although the interference-free scenario is also possible. Any miner U_n knows the values of p^n and $G^{n,m}$ and can estimate the values of $\tilde{p}^i$, $\tilde{G}^{i,m}$, $o^{i,m}$ and σ^2, e.g., based on the channel statistics. Hence, the miner can always compute the value of $SINR^n$ according to (4c) when selecting its offloading BS.

Accordingly, at any stage t, every miner U_n selects one BS, $b_t^n \in \mathbf{M}^n$, to offload its task with the BPR $x_t^n \in \mathbf{X}^n$ over a wireless channel with the bandwidth β^n. As such, since all system parameters (including the numbers of miners and BSs, locations of miners, and channel and computing resource allocations) are constant during each stage, the connectivity of the blockchain network is always preserved within one mining stage, i.e., there are no handovers or access blockages for the miners.

D. Expected Stage and Long-Term Payoffs of Miners

The instantaneous stage payoff that miner U_n receives at the end of stage t from its BS/BPR decision $(b_t^n, x_t^n) \in \mathbf{M}^n \times \mathbf{X}^n$ resulting in the reward $r_t^n \in \mathbf{R}^n$ is expressed by

$$u^n\big(b_t^n, x_t^n \big| r_t^n\big) = r_t^n - \sum_{m \in \mathbf{M}^n} \mathbf{1}_{b_t^n = m} c^m x_t^n. \tag{5}$$

The payoff in (5) is a function of the miner's BS/BPR decision (b_t^n, x_t^n) and a random reward r_t^n. From (3a), the probability of winning a reward (in (3a)) depends not only on the miner's own BS/BPR decision (b_t^n, x_t^n) but also on the decision profile of other miners $(\mathbf{b}_t^{-n}, \mathbf{x}_t^{-n})$, which is unobservable by miner U_n. Since miner U_n is uncertain about the accurate values of $(\mathbf{b}_t^{-n}, \mathbf{x}_t^{-n})$, it cannot compute its payoff directly based on (5) when making its BS/BPR decision. Nonetheless, the miner can construct its own estimate of $(\mathbf{b}_t^{-n}, \mathbf{x}_t^{-n})$. This estimate, also called a "belief", represents the joint probability distribution $B_t^n = \{B_t^n(\mathbf{b}_t^{-n}, \mathbf{x}_t^{-n})\}_{(\mathbf{b}_t^{-n}, \mathbf{x}_t^{-n}) \in \mathbf{M}^{-n} \times \mathbf{X}^{-n}}$, where

$$\begin{aligned} B_t^n\big(\mathbf{b}_t^{-n}, \mathbf{x}_t^{-n}\big) &= \prod_{i \in \mathbf{N} \setminus \{n\}} B_t^n\Big(b^i, x^i\Big) \\ &= \Pr\Big\{\big(\mathbf{b}_t^{-n}, \mathbf{x}_t^{-n}\big) = \Big(\mathbf{b}^{-n}, \mathbf{x}^{-n}\Big)\Big\} \\ &= \prod_{i \in \mathbf{N} \setminus \{n\}} \Pr\Big\{\Big(b_t^i, x_t^i\Big) = \Big(b^i, x^i\Big)\Big\} \end{aligned}$$

is the probability that miner U_n assigns to other miners having a decision profile $(\mathbf{b}^{-n}, \mathbf{x}^{-n})$ at stage t; $B_t^n(x^i)$ is the marginal probability.

Given its current belief B_t^n, any miner U_n can estimate the expected (w.r.t. belief) stage payoff from its BS/BPR decision (b_t^n, x_t^n), as in

$$\begin{aligned} U^n(B_t^n, b_t^n, x_t^n) = &\sum_{\big(\mathbf{b}_t^{-n}, \mathbf{x}_t^{-n}\big) \in \mathbf{M}^{-n} \times \mathbf{X}^{-n}} B_t^n\big(\mathbf{b}_t^{-n}, \mathbf{x}_t^{-n}\big) \\ &\times\ U^n\big(b_t^n, x_t^n \ \big|\ \mathbf{b}_t^{-n}, \mathbf{x}_t^{-n}\big), \end{aligned} \tag{6a}$$

where $U^n(b_t^n, x_t^n \mid \mathbf{b}_t^{-n}, \mathbf{x}_t^{-n})$ is the expected stage payoff of miner U_n from its BS/BPR decision (b_t^n, x_t^n) given a decision profile $(\mathbf{b}_t^{-n}, \mathbf{x}_t^{-n})$ of other miners, defined as

$$\begin{aligned} &U^n\big(b_t^n, x_t^n \mid \mathbf{b}_t^{-n}, \mathbf{x}_t^{-n}\big) \\ &= \sum_{r^n \in \mathbf{R}^n} \Pr\big\{r^n \mid b_t^n, x_t^n, \mathbf{b}_t^{-n}, \mathbf{x}_t^{-n}\big\} u^n(b_t^n, x_t^n \mid r^n) \\ &= x_t^n \sum_{m \in \mathbf{M}^n} \mathbf{1}_{b_t^n = m} \left(\frac{R + r\tau^n}{\sum_{i \in \mathbf{N}} x_t^i} \exp\big(-\tau^n z^n\big(b_t^n, \mathbf{b}_t^{-n}\big)/T_b\big) - c^m \right). \end{aligned} \tag{6b}$$

Next, note that at any stage t, miner U_n aims to maximize the payoff from its BS/BPR decision (b_t^n, x_t^n). Thus, if the miner is myopic or short-sighted, it will select a decision $(b_t^n, x_t^n) = \operatorname{argmax}_{(b^n, x^n)} U^n(B_t^n, b^n, x^n)$ maximizing its expected stage payoff U^n in (6a). If the miner is long-visional, it will select a decision $(b_t^n, x_t^n) = \operatorname{argmax}_{(b^n, x^n)} V^n(B_t^n, b^n, x^n)$ maximizing its expected long-term payoff V^n that is defined by an infinite discounted sum of its stage payoffs (as the miner stays in the system for an indefinitely-long time) [46], i.e.,

$$\begin{aligned} &V^n(B_t^n, b^n, x^n) \\ &= \mathrm{E}\left\{ \sum_{\tau=t}^{\infty} \gamma^{\tau - t} u^n(b_\tau^n, x_\tau^n \mid r_\tau^n) \Big| B_t^n, b_t^n = b^n, x_t^n = x^n \right\} \\ &= \mathrm{E}\left\{ \sum_{\tau=t}^{\infty} \gamma^{\tau - t} U^n(B_\tau^n, b_\tau^n, x_\tau^n) \Big| B_t^n, b_t^n = b^n, x_t^n = x^n \right\}, \end{aligned} \tag{7}$$

where $\gamma \in (0, 1]$ is a discounting factor.

As such, the interactions of miners can be modeled as a stochastic game played repeatedly at every stage t by the set $\mathbf{N}$ of miners. At the beginning of stage t, each miner U_n takes an action, i.e., the BS/BPR decision (b_t^n, x_t^n) that maximizes its expected stage payoff U^n or long-term payoff V^n. The miners select their actions simultaneously and independently, and do not exchange information about their actions. As a result, no miner U_n can observe the actions $(\mathbf{b}_t^{-n}, \mathbf{x}_t^{-n})$ taken by other miners, i.e., information about miners' actions is incomplete. To deal with this issue, the miner forms its belief B_t^n about actions $(\mathbf{b}_t^{-n}, \mathbf{x}_t^{-n})$. The game is stochastic, since the miner's payoff is defined by the random reward r_t^n which depends on the unobservable actions $(\mathbf{b}_t^{-n}, \mathbf{x}_t^{-n})$. In the following, we formulate a game for the miners, develop a POMDP model of miners' decisions to find the game solution, and propose the novel BRL and BDL framework that enables all miners to update their beliefs about actions of other miners and find their optimal strategies.

IV. Interactions of Miners in the Blockchain

A. Stochastic Game for Miners With Incomplete Information

From the description of the mining process in Section III, the interactions of the miners can be modeled as a stochastic non-cooperative game with unobservable actions. The game is played repeatedly by the set $\mathbf{N}$ of players/miners at every stage t. Recall that within any stage t, the set $\mathbf{N}$ is fixed, but can change at the beginning of the next stage $t+1$ due to arrivals and departures of miners during stage t. Information about the set $\mathbf{N}$ is available publicly, i.e., all players know the current set $\mathbf{N}$. At the beginning of stage t, each player/miner U_n realizes its action, i.e., BS/BPR decision $a_t^n = (b_t^n, x_t^n) \in \mathbf{A}^n = \mathbf{M}^n \times \mathbf{X}^n$, where $\mathbf{A}^n = \mathbf{M}^n \times \mathbf{X}^n$ is the action space of the player that combines the set $\mathbf{M}^n$ of its possible BSs' decisions (i.e., set of BSs in the range of miner U_n) and set $\mathbf{X}^n$ of its possible BPR decisions. All players select their actions simultaneously and independently, without exchanging information about their action. As a result, no player U_n can observe the actions $\mathbf{a}_t^{-n} = \{a_t^i\}_{i \in \mathbf{N} \setminus \{n\}} \in \mathbf{A}^{-n} = \times_{i \in \mathbf{N} \setminus \{n\}} \mathbf{A}^i$ taken by

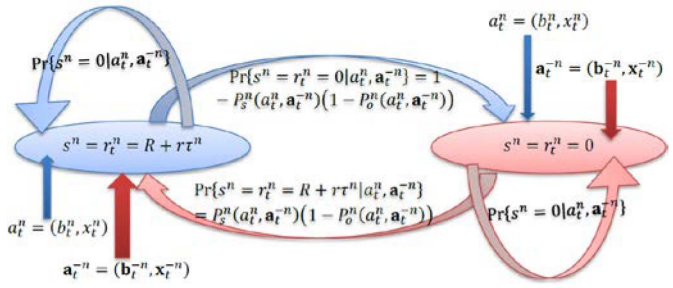

Fig. 3. Observable part of the Markov chain of player U_n, where the transition of state $s^n = r_t^n$ depends on the player's action $a_t^n = (b_t^n, x_t^n)$ and the unobservable actions $\mathbf{a}_t^{-n} = (\mathbf{b}_t^{-n}, \mathbf{x}_t^{-n})$ of other players.

other players, i.e., the information about $\mathbf{a}_t^{-n}$ is incomplete. Hence, the player forms its belief B_t^n about unobservable actions. After the actions $\mathbf{a}_t = \{a_t^n\}_{n\in\mathbf{N}} \in \mathbf{A} = \times_{n\in\mathbf{N}}\mathbf{A}^n$ are taken by all players, the game moves to some new random state $\mathbf{s} = \{s^n\}_{n\in\mathbf{N}} \in \mathbf{S} = \times_{n\in\mathbf{N}}\mathbf{S}^n$, where $s^n = r_t^n \in \mathbf{S}^n$ is the state of player U_n represented by the reward r_t^n received by the player at the end of stage t. Accordingly, the player's state space $\mathbf{S}^n = \mathbf{R}^n = \{0,\ R + r\tau^n\}$ comprises two possible states, $s^n = r_t^n = 0$ and $s^n = r_t^n = R + r\tau^n > 0$ with transitions defined according to (3a), as shown in Figure 3. The state $s^n = r_t^n$ is observable by player U_n at the end of stage t, i.e., after all players execute their actions $\mathbf{a}_t = (a_t^n, \mathbf{a}_t^{-n})$, but unknown at the beginning of stage t, i.e, before the actions $\mathbf{a}_t$ are taken. As such, the state space $\mathbf{S}^n$ is fully-observable but stochastic.

Based on the above, the proposed game is defined by the tuple $\mathbf{\Gamma} = (\mathbf{N}, \mathbf{A}^n, \mathbf{\Omega}^n, \mathbf{S}^n, \Pr, u^n)$ with the following elements:

- set of N players/miners $\mathbf{N}$ and, for each player/miner U_n,
- set of actions $\mathbf{A}^n = \mathbf{M}^n \times \mathbf{X}^n$ which is equivalent to the set of possible BS/BPR decisions of the player;
- partially-observable state space $\mathbf{\Omega}^n = \mathbf{S}^n \times \mathbf{A}^{-n}$ combining the observable and unobservable state spaces, $\mathbf{S}^n = \mathbf{R}^n$ and $\mathbf{A}^{-n}$, where the observable state space is represented by the set $\mathbf{R}^n$ of possible player's rewards and unobservable state space is represented by the set of possible action profiles of other players;
- state transition dynamics $\Pr\{\acute{s}^n, \acute{\mathbf{a}}^{-n} \mid s^n, a^n, \mathbf{a}^{-n}\}$, i.e., the probability of transiting to state $(\acute{s}^n, \acute{\mathbf{a}}^{-n}) = (\acute{r}^n, \acute{\mathbf{b}}^{-n}, \acute{\mathbf{x}}^{-n}) \in \mathbf{\Omega}^n$ given the action $a^n = (b^n, x^n) \in \mathbf{A}^n$ executed by the player in state $(s^n, \mathbf{a}^{-n}) = (r^n, \mathbf{b}^{-n}, \mathbf{x}^{-n}) \in \mathbf{\Omega}^n$ that can be further factored into two probabilities, $\Pr\{\acute{s}^n \mid s^n, a^n, \mathbf{a}^{-n}\}$ and $\Pr\{\acute{\mathbf{a}}^{-n} \mid \mathbf{a}^{-n}\}$;
- instantaneous stage payoff $u^n(s^n, a^n, \acute{s}^n)$ of the player from its action $a^n = (b^n, x^n) \in \mathbf{A}^n$ taken in state $s^n = r^n \in \mathbf{S}^n$ when transiting to state $\acute{s}^n = \acute{r}^n \in \mathbf{S}^n$, that is equivalent to the player's payoff $u^n(b^n, x^n \mid \acute{r}^n)$ in (5) from its BS/BPR decision $(b^n, x^n) \in \mathbf{M}^n \times \mathbf{X}^n$ resulting in reward $\acute{r}^n \in \mathbf{R}^n$.

Note that in conventional stochastic games with observable actions, the player's pure strategy $\alpha^n : \mathbf{\Omega}^n \to \mathbf{A}^n$ is a mapping from the state space $\mathbf{\Omega}^n$ to action space $\mathbf{A}^n$ [46]. In our game, however, the player's state space $\mathbf{\Omega}^n$ comprises two subsets, $\mathbf{S}^n$ and $\mathbf{A}^{-n}$, where the latter subset is unobservable by player U_n. Therefore, the player's pure strategy $\alpha^n(B^n, s^n) = a^n$ is a mapping from the 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: 1) at the beginning of stage t, the game is in state $s^n \in \mathbf{S}^n$; 2) after observing the state s^n, the player selects its pure strategy $\alpha^n(B_t^n, s^n) = a_t^n$ which maps from its current belief B_t^n and state $s^n \in \mathbf{S}^n$ to action $a_t^n \in \mathbf{A}^n$; 3) after the action $a_t^n \in \mathbf{A}^n$ is taken, the game moves to a new random state $\acute{s}^n \in \mathbf{S}^n$ and the player receives the payoff $u^n(s^n, a_t^n, \acute{s}^n)$. This procedure is repeated in the new state and the game continues for an infinite number of stages $T \to \infty$.

B. Solution Concepts for the Game

One possible solution concept applicable to game $\mathbf{\Gamma}$ is the Bayesian Nash equilibrium (BNE) [44]. In words, in the BNE, no player believes that it can increase its expected stage payoff by deviating from its pure strategy. Formally, BNE is the tuple $(\overline{\boldsymbol{\alpha}}, \overline{\mathbf{B}})$ that comprises a pure strategy profile $\overline{\boldsymbol{\alpha}} = \{\overline{\alpha}^n\}_{n\in\mathbf{N}} \in \mathbf{A} = \times_{n\in\mathbf{N}}\mathbf{A}^n$ and the players' beliefs $\overline{\mathbf{B}} = \{\overline{B}^n\}_{n\in\mathbf{N}}$, such that

$$U^n\left(\overline{B}^n, \overline{\alpha}^n \mid s^n\right) \geq U^n\left(\overline{B}^n, a^n \mid s^n\right), \quad \forall a^n \in \mathbf{A}^n, \forall s^n \in \mathbf{S}^n, \tag{8}$$

for each $n \in \mathbf{N}$, where $U^n(B^n, a^n | s^n)$ is the expected (w.r.t. belief B^n) stage payoff of player U_n from the action $a^n = (b^n, x^n)$ executed in state $s^n = r^n$, which is equivalent to the expected payoff $U^n(B^n, b^n, x^n)$ defined in (6a) and given by In (9a), the probability $\Pr\{\acute{s}^n | s^n, a^n, \mathbf{a}^{-n}\}$ is equivalent to the probability $\Pr\{\acute{r}^n | b^n, x^n, \mathbf{b}^{-n}, \mathbf{x}^{-n}\}$ defined in (3a), i.e., (9b), as shown at the bottom of the page, $U^n(a^n | s^n, \mathbf{a}^{-n})$ is the expected stage payoff of player U_n from the action

$$\begin{aligned} U^n\left(B^n, a^n \middle| s^n\right) &= \sum_{\acute{s}^n \in \mathbf{S}^n} \Pr\left\{\acute{s}^n \middle| s^n, a^n, B^n\right\} u^n(s^n, a^n, \acute{s}^n) \\ &= \sum_{\mathbf{a}^{-n} \in \mathbf{A}^{-n}} B^n\left(\mathbf{a}^{-n}\right) \sum_{\acute{s}^n \in \mathbf{S}^n} \Pr\left\{\acute{s}^n \middle| s^n, a^n, \mathbf{a}^{-n}\right\} u^n(s^n, a^n, \acute{s}^n) \\ &= \sum_{\mathbf{a}^{-n} \in \mathbf{A}^{-n}} B^n\left(\mathbf{a}^{-n}\right) U^n\left(a^n \middle| s^n, \mathbf{a}^{-n}\right) \end{aligned} \tag{9a}$$

$$\Pr\left\{\acute{s}^n \middle| s^n, a^n, \mathbf{a}^{-n}\right\} = \begin{cases} P_s^n\left(a^n, \mathbf{a}^{-n}\right)\left(1 - P_o^n\left(a^n, \mathbf{a}^{-n}\right)\right), & \acute{s}^n = \acute{r}^n = R + r\tau^n \\ 1 - P_s^n\left(a^n, \mathbf{a}^{-n}\right)\left(1 - P_o^n\left(a^n, \mathbf{a}^{-n}\right)\right), & \acute{s}^n = \acute{r}^n = 0 \end{cases} \tag{9b}$$

$a^n = (b^n, x^n)$ taken in state $s^n = r^n$ given actions $\mathbf{a}^{-n} = (\mathbf{b}^{-n}, \mathbf{x}^{-n})$ of other players, which is equivalent to the expected payoff $U^n(b^n, x^n|\mathbf{b}^{-n}, \mathbf{x}^{-n})$ in (6b) and given by

$$U^n\big(a^n\big|s^n, \mathbf{a}^{-n}\big) = \sum_{\acute{s}^n \in \mathbf{S}^n} \Pr\big\{\acute{s}^n\big|s^n, a^n, \mathbf{a}^{-n}\big\} \times u^n(s^n, a^n, \acute{s}^n). \tag{9c}$$

Note that since the number of players N and their state/action space $\mathbf{S} \times \mathbf{A}$ are finite, at any mining stage, game $\mathbf{\Gamma}$ admits a BNE [44].

Unfortunately, the concept of BNE is rather incomplete for analysis of stochastic games played repeatedly over multiple stages, as it puts no restrictions on the dynamics (i.e., updating rule) of players' beliefs [14], [44], [45]. In order to refine the solution generated by BNE, the concept of a myopic perfect Bayesian equilibrium (MPBE) [14], [45] can be utilized. In words, MPBE is the BNE where, at every stage, the players' beliefs are consistent, i.e., updated with Bayesian inference, and their pure strategies are sequentially rational, i.e., for any information set or history $(s_0^n, a_0^n, \ldots, s_{t-1}^n, a_{t-1}^n, s_t^n)$, the pure strategies maximize expected stage payoffs of the players [14], [45]. Formally, MPBE is the tuple $(\overline{\boldsymbol{\alpha}}, \overline{\mathbf{B}})$ that satisfies (8), for all $n \in \mathbf{N}$, where the belief B^n is updated based on the last-observed transition $(\acute{s}^n, a^n, s^n)$ according to the Bayes' rule:

$$\overline{B}^n\big(\mathbf{a}^{-n}\big) = \frac{\Pr\big\{\acute{s}^n \;\big|\; s^n, a^n, \mathbf{a}^{-n}\big\} B^n\big(\mathbf{a}^{-n}\big)}{\sum_{\acute{\mathbf{a}}^{-n} \in \mathbf{A}^{-n}} \Pr\big\{\acute{s}^n \;\big|\; s^n, a^n, \acute{\mathbf{a}}^{-n}\big\} B^n(\acute{\mathbf{a}}^{-n})}, \tag{10}$$

for all $\mathbf{a}^{-n} \in \mathbf{A}^{-n}$. Similar to BNE, game $\mathbf{\Gamma}$ admits a MPBE, as the number of players and their state/action space are finite [44].

Note that in the MPBE, the players are myopic, i.e., short-sighted, because they only select the strategies that maximize their one-stage (rather than the long-term) expected payoffs. In this regard, the notion of a perfect Bayesian equilibrium (PBE) is more suitable for game $\mathbf{\Gamma}$ [14], [44]. In words, PBE is the MPBE where the player selects a pure strategy that maximizes its value, i.e., expected long-term payoff [14], [44]. Formally, PBE is the tuple $(\overline{\boldsymbol{\alpha}}, \overline{\mathbf{B}})$, such that

$$V^n\big(\overline{B}^n, \overline{\alpha}^n \;\big|\; s^n\big) \geq V^n\big(\overline{B}^n, a^n \;\big|\; s^n\big), \quad \forall a^n \in \mathbf{A}^n, \forall s^n \in \mathbf{S}^n, \tag{11}$$

for all $n \in \mathbf{N}$. In (11), $V^n(B^n, a^n|s^n)$ is the value or expected (w.r.t. belief B^n) long-term payoff of player U_n from its action $a^n = (b^n, x^n)$ taken in state $s^n = r^n$, which can be defined, according to (7), as

$$\begin{aligned} &V^n(B^n, a^n \mid s^n) \\ &= \mathrm{E}\left\{ \sum_{t=0}^{\infty} \gamma^t u^n(s^n, a^n, \acute{s}^n) \;\middle|\; B^n, a^n, s^n \right\} \\ &= \sum_{\acute{s}^n \in \mathbf{S}^n} \Pr\{\acute{s}^n \mid s^n, a^n, B^n\} \\ &\quad \times \left(u^n(s^n, a^n, \acute{s}^n) + \gamma V^n\left(B_{a^n}^{s^n, \acute{s}^n}, a^n \;\middle|\; \acute{s}^n\right)\right) \\ &= \mathrm{E}\left\{ \sum_{t=0}^{\infty} \gamma^t U^n(s^n, a^n, \acute{s}^n) \;\middle|\; B^n, a^n, s^n \right\} \\ &= U^n(B^n, a^n \mid s^n) \\ &\quad + \gamma \sum_{\acute{s}^n \in \mathbf{S}^n} \Pr\{\acute{s}^n \mid s^n, a^n, B^n\} V^n\left(B_{a^n}^{s^n, \acute{s}^n}, a^n \;\middle|\; \acute{s}^n\right) \\ &= \sum_{\mathbf{a}^{-n} \in \mathbf{A}^{-n}} B^n\big(\mathbf{a}^{-n}\big) V^n\big(a^n \mid s^n, \mathbf{a}^{-n}\big), \end{aligned} \tag{12a}$$

where for every transition $(s^n, a^n, \acute{s}^n)$ and all $\mathbf{a}^{-n} \in \mathbf{A}^{-n}$, the belief $B_{a^n}^{s^n, \acute{s}^n}$ is updated according to the Bayes' rule in (10); $V^n(a^n|s^n, \mathbf{a}^{-n})$ is the expected long-term payoff of player U_n from the action $a^n = (b^n, x^n)$ executed in state $s^n = r^n$ given actions $\mathbf{a}^{-n} = (\mathbf{b}^{-n}, \mathbf{x}^{-n})$ of other players, defined by

given actions $\mathbf{a}^{-n} = (\mathbf{b}^{-n}, \mathbf{x}^{-n})$ of other players

$$\begin{aligned} &V^n\big(a^n \mid s^n, \mathbf{a}^{-n}\big) \\ &= \mathrm{E}\left\{ \sum_{t=0}^{\infty} \gamma^t u^n(s^n, a^n, \acute{s}^n) \;\middle|\; a^n, s^n, \mathbf{a}^{-n} \right\} \\ &= \sum_{\acute{s}^n \in \mathbf{S}^n} \Pr\big\{\acute{s}^n \mid s^n, a^n, \mathbf{a}^{-n}\big\} \\ &\quad \times \big(u^n(s^n, a^n, \acute{s}^n) + \gamma V^n\big(a^n \mid \acute{s}^n, \mathbf{a}^{-n}\big)\big) \\ &= \mathrm{E}\left\{ \sum_{t=0}^{\infty} \gamma^t U^n(s^n, a^n, \acute{s}^n) \;\middle|\; a^n, s^n, \mathbf{a}^{-n} \right\} \\ &= U^n\big(a^n \mid s^n, \mathbf{a}^{-n}\big) \\ &\quad + \gamma \sum_{\acute{s}^n \in \mathbf{S}^n} \Pr\big\{\acute{s}^n \mid s^n, a^n, \mathbf{a}^{-n}\big\} V^n\big(a^n \mid \acute{s}^n, \mathbf{a}^{-n}\big). \end{aligned} \tag{12b}$$

Similar to the BNE and MPBE, game $\mathbf{\Gamma}$ admits a PBE, as the number of players and their state/action space are finite [44]. However, the computation of the PBE or MBPE is non-deterministic polynomial time (NP) hard, because it is at least as complex as the computation of the BNE, which is an NP-hard problem [44]. Thus, instead of directly computing the PBE, in the next section, we formulate a POMDP model of the miner's decision, the solution of which represents a PBE. Based on the POMDP, we develop the BRL algorithm which allows each miner to reach this solution independently, i.e., without communicating with other miners.

C. Bayesian Reinforcement Learning Algorithm for Miners

In repeated game settings, any player/miner can estimate the unknown state transition dynamics and update its belief about unobservable actions of other players by adopting stochastic dynamic programming and RL. Accordingly, in the following, we develop the BRL framework that allows all players/miners to select their optimal strategies and infer the distributions of unobservable actions and observable states during repeated interactions with each other and a stochastic environment. We start by defining a POMDP that will be used to further develop a RL model. Formally, a POMDP is the tuple $(\mathbf{\Omega}, \mathbf{S}, \mathbf{A}, \Pr, u, \gamma)$ with the following elements [46]: i) partially-observable state space $\mathbf{\Omega} = \mathbf{S} \times \mathbf{\Theta}$ combining the observable and unobservable state spaces, $\mathbf{S}$ and $\mathbf{\Theta}$; ii) action

space $\mathbf{A}$; iii) state transition dynamics $\Pr\{\acute{s},\acute{\theta}|s,a,\theta\} = \Pr\{\acute{s}|s,a,\theta\}\Pr\{\acute{\theta} \mid \theta\}$ defining the probability of transiting to state $(\acute{s},\acute{\theta}) \in \mathbf{\Omega}$ given the action $a \in \mathbf{A}$ taken in state $(s,\theta) \in \mathbf{\Omega}$; iv) payoff $u(s,a,\acute{s})$ from the action $a \in \mathbf{A}$ taken in state $s \in \mathbf{S}$ when transiting to state $\acute{s} \in \mathbf{S}$; v) discounting factor $\gamma \in (0,1]$. As the transition $\Pr\{\acute{\theta}|\theta\}$ is not directly observable, a learner relies only on the observable transition $(s,a,\acute{s})$ to infer an underlying distribution or belief $B = \{B(\theta)\}_{\theta\in\mathbf{\Theta}}$, for $B(\theta) = \Pr\{\theta\}$. The RL problem it to find the optimal pure strategy $\overline{\alpha}(B) \in \mathbf{A}$ that maximizes the value $V^{\alpha}(B) = \sum_{t=0}^{\infty}\gamma^t u(\alpha(B_t), s_t)$ in all beliefs states, i.e., $V^{\overline{\alpha}}(B) \geq V^{\alpha}(B)$, for all α and B, and satisfies the Bellman optimality equation [46]:

$$V^{\overline{\alpha}}(B) = \max_{a\in\mathbf{A}} \sum_{\acute{s}\in\mathbf{S}} \Pr\{\acute{s} \mid s,a,B\} \times \Big(u(s,a,\acute{s}) + \gamma V^{\overline{\alpha}}\Big(B_a^{s,\acute{s}}\Big)\Big), \tag{13}$$

where the belief $B_a^{s,\acute{s}}$ is updated according to Bayes' rule.

Apparently, the problem of finding optimal strategies of the players in a stochastic game with incomplete information can be cast as a POMDP [46], [47]. In particular, when applied to game $\mathbf{\Gamma}$, the POMDP for each player U_n is defined by the tuple $(\mathbf{\Omega}^n, \mathbf{S}^n, \mathbf{A}^n, \Pr, u^n, \gamma)$, with the elements described in Section IV. The RL problem is to find the optimal strategy $\overline{\alpha}^n(B^n, s^n) \in \mathbf{A}^n$ which maximizes the player's value, i.e., expected long-term payoff $V^n(B^n, a^n|s^n)$ defined in (12a). This problem can be solved with stochastic dynamic programming [45], [46]. In this case, we obtain the BRL algorithm where at every stage *t*, we store the set of $|\mathbf{S}^n \times \mathbf{A}|$ updated values $V_{t+1}^n(a^n|s^n,\mathbf{a}^{-n})$ indexed by the state $s^n \in \mathbf{S}^n$ and actions $(a^n,\mathbf{a}^{-n}) \in \mathbf{A}$. The proposed algorithm is implemented "on-line" – it is repeated at each stage *t* of the mining/learning process and does not require a prior "off-line" training and/or pre-existing training datasets. At the beginning of stage *t*, for each player/miner U_n, we perform the following steps.

Step 1: Select the optimal pure strategy $\alpha^n(B_t^n, s^n)$ for the current belief B_t^n and state s^n by solving the Bellman equation

$$\alpha^n(B_t^n, s^n) = \underset{a^n\in\mathbf{A}^n}{\operatorname{argmax}}\ V_{t+1}^n(B_t^n, a^n \mid s^n), \tag{14a}$$

where $V_{t+1}^n(B_t^n, a^n|s^n)$ is given by (14b), as shown at the bottom of the page. $V_{t+1}^n(a^n|s^n,\mathbf{a}^{-n})$ is estimated based on the last-updated value $V_t^n(a^n|\acute{s}^n,\mathbf{a}^{-n})$ according to (14c), as shown at the bottom of the page, and $\Pr\{\acute{s}^n|s^n,a^n,\mathbf{a}^{-n}\}$ is defined in (9b);

Step 2: Take action $a_t^n = \alpha^n(B_t^n, s^n)$, observe a resulting state $\acute{s}^n$. For the observed transition $(s^n, a_t^n, \acute{s}^n)$, update the belief $B_{t+1}^n(\mathbf{a}^{-n}) = B_{a_t^n}^{s^n,\acute{s}^n}(\mathbf{a}^{-n})$ with the Bayes' rule in (10), for all $\mathbf{a}^{-n} \in \mathbf{A}^{-n}$.

Note that the above algorithm does not need any additional exploration, because it is implicit in the computation of value $V_{t+1}^n(B_t^n, a^n|s_t^n)$ in (14b). Nevertheless, when the exploration ability is limited, e.g., due to a finite recursion depth, it may be useful to utilize some explicit exploration technique [46]. For example, we can adopt a simple ε-greedy exploration where, for a certain arbitrary small $\varepsilon \in [0,1)$, the optimal strategy $a_t^n = \alpha^n(B_t^n, s^n)$ is selected with probability $1-\varepsilon$, and a non-optimal strategy $\acute{a}_t^n \in \mathbf{A}^n\backslash\{a_t^n\}$ is selected with probability ε. Proposition 1 below shows that the proposed BRL algorithm that represents the sequence of iterations $\{V_t^n\}_{t\in\mathbb{N}}$ converges to the optimal value $\overline{V}^n$, i.e., value of an optimal pure strategy $\overline{\alpha}$.

Proposition 1: The BRL algorithm defined by the sequence of iterations $\{V_t^n\}_{t\in\mathbb{N}}$, for all $n \in \mathbf{N}$, converges to the optimal value $\overline{V}^n$ with probability one as time tends to infinity.

The proof of Proposition 1 is given in Appendix C in the supplementary material. From Proposition 1, we obtain Corollary 1 that establishes that our BRL algorithm converges to a stable state where the players' pure strategies and beliefs are in the PBE of game $\mathbf{\Gamma}$.

Corollary 1: The BRL algorithm defined by the sequence of iterations $\{V_t^n\}_{t\in\mathbb{N}}$, for all $n \in \mathbf{N}$, converges to a stable state in which the players' pure strategies and beliefs $(\overline{\boldsymbol{\alpha}}, \overline{\mathbf{B}})$ are in the perfect Bayesian equilibrium of game $\mathbf{\Gamma}$.

The proof of Corollary 1 is given in Appendix D in the supplementary material. Finally, in Proposition 2 below, we analyze the worst-case computational complexity and the rate of convergence of the BRL algorithm.

Proposition 2: The BRL algorithm defined by the sequence of iterations $\{V_t^n\}_{t\in\mathbb{N}}$, for all $n \in \mathbf{N}$, yields the worst-case time complexity of $\mathcal{O}(c^N)$, for $c = |\max_{n\in\mathbf{N}} \mathbf{A}^n|$ and an asymptotic convergence rate equal to $\mathcal{O}(\sqrt{\log(\log t)/t})$, for $\gamma \leq 0.5$, and $\mathcal{O}(1/t^{1-\gamma})$, for $\gamma > 0.5$.

The proof of Proposition 2 is provided in Appendix E in the supplementary material.

V. Bayesian Deep Learning by the Miners

From Proposition 2, the convergence rate of the proposed BRL algorithm is sublinear with order $1-\gamma \in (0,\ 0.5)$ and its complexity is exponential in the number of miners *N*. Hence, although the BRL algorithm converges to an exact solution of the POMDP, it is intractable for the large number of miners *N* and, to be practically-realizable, it requires an approximation. Accordingly, in this section, we devise a novel BDL algorithm where uncertainties about unobservable states of the POMDP are approximated by the BNN. This enables us to reduce the

$$V_{t+1}^n\big(B_t^n, a^n\big|s^n\big) = \sum_{\acute{s}^n\in\mathbf{S}^n} \Pr\big\{\acute{s}^n\big|s^n,a^n,B_t^n\big\}\Big(u^n(s^n,a^n,\acute{s}^n) + \gamma V_t^n\Big(B_{a^n}^{s_t^n,\acute{s}^n}, a^n\big|\acute{s}^n\Big)\Big)$$
$$= \sum_{\mathbf{a}^{-n}\in\mathbf{A}^{-n}} B_t^n\big(\mathbf{a}^{-n}\big)\, V_{t+1}^n\big(a^n\big|s^n,\mathbf{a}^{-n}\big) \tag{14b}$$

$$V_{t+1}^n\big(a^n\big|s^n,\mathbf{a}^{-n}\big) = \sum_{\acute{s}^n\in\mathbf{S}^n} \Pr\big\{\acute{s}^n\big|s^n,a^n,\mathbf{a}^{-n}\big\}\big(u^n(s^n,a^n,\acute{s}^n) + \gamma V_t^n\big(a^n\big|\acute{s}^n,\mathbf{a}^{-n}\big)\big) \tag{14c}$$

size of the unobservable state space and, thus, the complexity of a learning process. In addition, the proposed BDL algorithm is unsupervised, i.e., it allows the type of self-organized "online" learning that does not need any prior "off-line" training and/or pre-existing training datasets. The rest of this section is organized as follows. In Section V-A, we describe the existing BDL methods for Bayesian inference modeling with BNNs. In Section VII-B, we construct the BNN model of the miners' decisions. In Section VII-C, we develop the BDL algorithm.

A. Inference Approximation With Bayesian Neural Networks

DL algorithms have a proven record in producing rather accurate solutions for fully-observable MDPs [3], [11], [14], [42], [48]. However, they are not effective when applied to POMDPs [14], [49], [50]. The reason is that the DL methods are based on conventional NNs that cannot capture epistemic uncertainties about model parameters which are inherent in POMDPs where such uncertainties are defined as probability distributions over unobservable states (i.e., the beliefs). As a result, although the DL algorithms converge to some fixed values, in general, these values are rather far from the optimal POMDP solutions (a detailed explanation of reasons behind the poor performance of DL algorithms in POMDPs is given in [49]). Fortunately, epistemic uncertainties can be efficiently tracked with BDL methods where they are modeled with BNNs – NNs with random weights following some predefined prior distribution. Then, instead of optimizing the network weights directly, as in DL algorithms, we average over all possible weights, which is known as a marginalization [51]. In the following, we describe the existing BDL methods for Bayesian inference approximation with BNNs.

Consider a dataset $(\mathbf{X},\ \mathbf{Y}) = \{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1,\dots,F}$ that consists of F data points $(\mathbf{x}_t, \mathbf{y}_t)$, where $\mathbf{x}_t \in \mathbb{R}^{1\times D_I}$ is the row vector of inputs with D_I elements and $\mathbf{y}_t \in \mathbb{R}^{1\times D_O}$ is the row vector of outputs, i.e., targets, with D_O elements. Let $\Pr\{\mathbf{y}_t \mid f^{\mathbf{W}}(\mathbf{x}_t)\}$ be the likelihood of a data point $(\mathbf{x}_t, \mathbf{y}_t)$, where $f^{\mathbf{W}}(\mathbf{x}_t) = \widehat{\mathbf{y}}_t \in \mathbb{R}^{1\times D_O}$ is a random row output vector of the BNN model with some prior distribution over its weights $\mathbf{W}$, e.g., a standard normal distribution $\Pr\{\mathbf{W}\} = \mathcal{N}(0, I)$. In regression tasks, the likelihood is usually represented by the Gaussian distribution $\Pr\{\mathbf{y}_t \mid f^{\mathbf{W}}(\mathbf{x}_t)\} = \mathcal{N}(f^{\mathbf{W}}(\mathbf{x}_t), \sigma^2)$, where the mean $f^{\mathbf{W}}(\mathbf{x}_t)$ is defined by the model and a given observation noise variance σ^2. In classification tasks, the observed target output $\mathbf{y}_t \in \mathbf{Y} = \{1, \dots, D_O\}$ represents the label in the set $\mathbf{Y} = \{1, \dots, D_O\}$. Thus, the random output $f^{\mathbf{W}}(\mathbf{x}_t) = \widehat{\mathbf{y}}_t = (\hat{y}_t^1, \dots, \hat{y}_t^{D_O}) \in \mathbb{R}^{1\times D_O}$ of the BNN model is predicted being classified with a label in the set $\mathbf{Y} = \{1, \dots, D_O\}$. To utilize the model for classification, we pass the output through an element-wise softmax function to obtain the likelihood of the output label $\mathbf{y}_t = i \in \mathbf{Y}$, given by

$$\Pr\left\{\mathbf{y}_t = i \middle| f^{\mathbf{W}}(\mathbf{x}_t)\right\} = \mathrm{Softmax}\Big(i, f^{\mathbf{W}}(\mathbf{x}_t)\Big) = \frac{e^{\hat{y}_t^i}}{\sum_{j=1}^{D_O} e^{\hat{y}_t^j}}, \tag{15}$$

for each $i \in \mathbf{Y}$ [49], [51].

Given the likelihood $\Pr\{\mathbf{y}_t | f^{\mathbf{W}}(\mathbf{x}_t)\}$, we are interested in estimating the posterior over model weights $\mathbf{W}$ for a data set $(\mathbf{X}, \mathbf{Y})$, i.e., the probability Pr $\{\mathbf{W}|\mathbf{X}, \mathbf{Y}\}$. Note that typically, a posterior is updated with the Bayesian inference Pr $\{\mathbf{W}|\mathbf{X}, \mathbf{Y}\}$ = Pr $\{\mathbf{W}\}$Pr $\{\mathbf{Y}|\mathbf{X},\mathbf{W}\}$/Pr$\{\mathbf{Y}|\mathbf{X}\}$. Unfortunately, in the BNNs, the computation of inference is intractable, because the marginal probability Pr $\{\mathbf{Y}|\mathbf{X}\}$ involved in the estimation of Pr $\{\mathbf{W}|\mathbf{X}, \mathbf{Y}\}$ cannot be evaluated analytically [49]. Hence, instead of direct inference, in Bayesian modeling, a variational inference [49], [51] is utilized where the posterior Pr $\{\mathbf{W}|\mathbf{X}, \mathbf{Y}\}$ is fitted with some simple distribution $q^\theta(\mathbf{W})$ parameterized by θ, such that $\Pr\{\mathbf{W}|\mathbf{X},\ \mathbf{Y}\} \sim\ \overline{q}^\theta(\mathbf{W})$, where $\overline{q}^\theta$ is the optimal approximating distribution. This replaces an intractable problem of averaging over model weights $\mathbf{W}$ with a simpler problem of optimizing the parameters θ. More specifically, the optimal parameters $\overline{\theta}$ should minimize the Kullback-Leibler (KL) divergence to a true posterior Pr $\{\mathbf{W}|\mathbf{X}, \mathbf{Y}\}$, i.e.,

$$\overline{q}^\theta(\mathbf{W}) = \min_\theta D_{\mathrm{KL}}\Big(q^\theta(\mathbf{W}) \ \Big|\ \Pr\{\mathbf{W} \mid \mathbf{X},\ \mathbf{Y}\})\Big), \tag{16a}$$

where

$$D_{\mathrm{KL}}\Big(q^\theta(\mathbf{W}) \ \Big|\ \Pr\{\mathbf{W} \mid \mathbf{X},\ \mathbf{Y}\})\Big) = \int q^\theta(\mathbf{W}) \log \frac{q^\theta(\mathbf{W})}{\Pr\{\mathbf{W} \mid \mathbf{X},\ \mathbf{Y}\}} \mathrm{d}\mathbf{W} \tag{16b}$$

is the KL divergence, i.e., the difference between $q^\theta(\mathbf{W})$ and Pr $\{\mathbf{W}|\mathbf{X},\mathbf{Y}\}$. Note that the minimization of a KL divergence is equivalent to the maximization of the evidence lower bound (ELBO):

$$\begin{aligned}\mathrm{ELBO}(\theta) &= \int q^\theta(\mathbf{W}) \log \Pr\{\mathbf{Y} \mid \mathbf{X},\ \mathbf{W}\} \mathrm{d}\mathbf{W} \\ &\quad - D\Big(q^\theta(\mathbf{W}) \ \Big|\ \Pr\{\mathbf{W}\})\Big) \\ &= \sum_{t=1}^{F} \int q^\theta(\mathbf{W}) \log \Pr\Big\{\mathbf{y}_t \ \Big|\ f^{\mathbf{W}}(\mathbf{x}_t)\Big\} \mathrm{d}\mathbf{W} \\ &\quad - D_{\mathrm{KL}}\Big(q^\theta(\mathbf{W}) \ \Big|\ \Pr\{\mathbf{W}\})\Big) \\ &\le \log \Pr\{\mathbf{Y} \mid \mathbf{X}\},\end{aligned} \tag{17}$$

where $\log \Pr\{\mathbf{Y}|\mathbf{X}\}$ is the log-evidence. Maximizing the first term in (17) is equivalent to maximizing the expected log-likelihood. Maximizing the second term in (17) is equivalent to minimizing a KL divergence between a distribution $q^\theta(\mathbf{W})$ and a prior $\Pr\{\mathbf{W}\}$.

A practical approach to approximate inference in large and complex BNNs is a dropout variational inference [49], where dropout is interpreted as a variational Bayesian inference and a distribution $q^\theta(\mathbf{W})$ is modeled as a mixture of two Gaussians with small variances and the mean of one of Gaussians fixed at zero. The objective is to minimize the loss, given by [49]

$$\begin{aligned}\mathcal{L}(\theta) &= -\mathrm{ELBO}(\theta) \\ &= -\sum_{t=1}^{F} \int q^\theta(\mathbf{W}) \log \Pr\Big\{\mathbf{y}_t \ \Big|\ f^{\mathbf{W}}(\mathbf{x}_t)\Big\} \mathrm{d}\mathbf{W} \\ &\quad + D_{\mathrm{KL}}\Big(q^\theta(\mathbf{W}) \ \Big|\ \Pr\{\mathbf{W}\})\Big)\end{aligned}$$

$$= -\sum_{t=1}^{F} \log \Pr\Big\{ \mathbf{y}_t \;\Big|\; f^{\widehat{\mathbf{W}}_t}(\mathbf{x}_t) \Big\} + \int q^{\theta}(\mathbf{W}) \log \frac{q^{\theta}(\mathbf{W})}{\Pr\{\mathbf{W}\}} \mathrm{d}\mathbf{W}, \tag{18a}$$

where the weights $\widehat{\mathbf{W}}_t$ are distributed according to $q^{\theta}(\mathbf{W})$, i.e., $\Pr\{\widehat{\mathbf{W}}_t\} = q^{\theta}(\mathbf{W})$. Note that in the classification tasks, a log-likelihood of an observed output label $\mathbf{y}_t = i \in \mathbf{Y}$ is given by

$$\log \Pr\Big\{ \mathbf{y}_t = i \Big| f^{\widehat{\mathbf{W}}_t}(\mathbf{x}_t) \Big\} = \log\Big(\mathrm{Softmax}\Big(i, f^{\widehat{\mathbf{W}}_t}(\mathbf{x}_t) \Big) \Big), \tag{18b}$$

where $f^{\widehat{\mathbf{W}}_t}(\mathbf{x}_t) = \widehat{\mathbf{y}}_t = (\hat{y}_t^1, \ldots, \hat{y}_t^{D_O})$. Thus, to find the optimal parameters $\overline{\theta}$ of $q^{\theta}(\mathbf{W})$, we should estimate the derivatives of loss $\mathcal{L}(\theta)$ w.r.t. θ, which can be done with Monte Carlo (MC) estimation [49], [50].

A rather efficient MC estimators is the path-wise derivative estimator (PDE) [49], also called the infinitesimal perturbation analysis, stochastic back-propagation, or a re-parameterization trick. In PDE, $q^{\theta}(\mathbf{W})$ is re-parameterized as a parameter-free distribution $q(\epsilon)$ subject to $\widehat{\mathbf{W}}_t = g(\theta, \epsilon)$, where $\epsilon > 0$ is an infinitesimal number; $g(\cdot, \cdot)$ is a deterministic differentiable bivariate transformation, e.g., for Gaussian distribution $q^{\theta}(\mathbf{W}) = \mathcal{N}(\mu, \sigma^2)$, with $\theta = (\mu, \sigma)$, we can set $\widehat{\mathbf{W}}_t = g(\theta, \epsilon) = \mu + \sigma\epsilon$ and $q(\epsilon) = \mathcal{N}(0, I)$. Then, it can be shown that for a normal prior $\Pr\{\mathbf{W}\} = \mathcal{N}(0, I)$, the loss takes a simpler form [49]:

$$\mathcal{L}(\theta, \lambda) = \frac{1}{2}(1-\lambda)\|\theta\|^2 - \sum_{t=1}^{F} \log \Pr\Big\{ \mathbf{y}_t \;\Big|\; f^{g(\theta,\epsilon)}(\mathbf{x}_t) \Big\}, \tag{19a}$$

where $\lambda \in [0,\ 1]$ is a dropout probability which also must be optimized. In the classification tasks, a log-likelihood of any observed output label $\mathbf{y}_t = i \in \mathbf{Y}$ takes the form:

$$\log \Pr\Big\{ \mathbf{y}_t = i \;\Big|\; f^{g(\theta,\epsilon)}(\mathbf{x}_t) \Big\} = \log\Big(\mathrm{Softmax}\Big(i, f^{g(\theta,\epsilon)}(\mathbf{x}_t) \Big) \Big). \tag{19b}$$

where $f^{g(\theta,\epsilon)} = \widehat{\mathbf{y}}_t = (\hat{y}_t^1, \ldots, \hat{y}_t^{D_O})$.

Based on the theoretical findings in [49], if the number of data points F in the data set $(\mathbf{X},\ \mathbf{Y})$ is sufficiently large, then i) the loss $\mathcal{L}(\theta, \lambda)$ in (19a) approaches the loss $\mathcal{L}(\theta)$ in (18a), i.e,

$$\mathcal{L}(\theta, \lambda) \underset{F\to\infty}{\longrightarrow} \mathcal{L}(\theta) = -\mathrm{ELBO}(\theta) = D_{\mathrm{KL}}\Big(q^{\theta}(\mathbf{W}) \;\Big|\; \Pr\{\mathbf{W} \mid \mathbf{X},\ \mathbf{Y}\}) \Big), \tag{20a}$$

ii) weights $\widehat{\mathbf{W}}_t$ generated according to optimal approximating distribution $\overline{q}^{\theta}(\mathbf{W}) = q^{\overline{\theta}}(\mathbf{W})$ approach weights $\mathbf{W}_t$ updated with Bayesian inference. Thus, if F is sufficiently large, we have $\widehat{\mathbf{W}}_t \longrightarrow \mathbf{W}_t$ and

$$\begin{cases} \Pr\{\mathbf{W} \mid \mathbf{X},\ \mathbf{Y}\} \underset{F\to\infty}{\longrightarrow} \overline{q}^{\theta}(\mathbf{W}) = \Pr\Big\{\widehat{\mathbf{W}}\Big\}, \\ \Pr\{\mathbf{y} \mid \mathbf{x}, \mathbf{X},\ \mathbf{Y}\} \underset{F\to\infty}{\longrightarrow} \int \Pr\Big\{ \mathbf{y} \;\Big|\; f^{\widehat{\mathbf{W}}}(\mathbf{x}) \Big\} \overline{q}^{\theta}(\mathbf{W}) \mathrm{d}\mathbf{W} = \overline{q}^{\theta}(\mathbf{y} \mid \mathbf{x}), \\ \Pr\Big\{ \mathbf{Y} \;\Big|\; \mathbf{X}, \widehat{\mathbf{W}} \Big\} \underset{F\to\infty}{\longrightarrow} \Pr\{\mathbf{Y} \mid \mathbf{X}\}. \end{cases} \tag{20b}$$

B. Bayesian Neural Network Model of the Miners' Decisions

To summarize the presentation in Section V-A, the existing BDL framework provides the means to model and accurately approximate Bayesian inference with the BNNs. In particular, given the observable dataset $(\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1,\ldots,F}$ and the likelihood $\Pr\{\mathbf{y}_t \| f^{\mathbf{W}}(\mathbf{x}_t)\}$ of every data point $(\mathbf{x}_t, \mathbf{y}_t)$, in the BDL framework, a posterior $\Pr\{\mathbf{W}|\mathbf{X},\ \mathbf{Y}\}$ over random BNN weights $\mathbf{W}$ is approximated by some simple distribution $\overline{q}^{\theta}$ parameterized by θ. This replaces the intractable computation of Bayesian inference with a simpler problem of optimizing parameters θ in a way which minimizes the approximation loss $\mathcal{L}$, so that $\Pr\{\mathbf{W}|\mathbf{X},\ \mathbf{Y}\} \underset{F\to\infty}{\longrightarrow} \overline{q}^{\theta}(\mathbf{W})$ [49]. However, none of the existing BDL methods shows how to adopt the BNN-based inference approximation in practical learning. That is, they do not address the application of this BNN approximation in the settings when the learner must not only accurately model the Bayesian inference, but also determine an optimal strategy that maximizes the learner's value or expected payoff. As such, in the practical BDL algorithm, a BNN model should be utilized not only to represent Bayesian inference, but also to estimate the expected payoffs of a learner from each of its actions. In the following, we formulate such a practical BDL algorithm and apply it for decision making of each player/miner U_n in game $\mathbf{\Gamma}$. More specifically, we: i) devise a BNN model $f^{\mathbf{W}}(\mathbf{x})$ which approximates unobservable state transitions of the POMDP of the player; ii) show how to approximate the player's belief B^n and its value or expected stage payoff U^n with the BNN model $f^{\mathbf{W}}(\mathbf{x})$; iii) develop a BDL algorithm where the player selects a pure strategy α^n that maximizes its approximated value U^n and, at the same time, the random weights $\mathbf{W}$ of the BNN are updated to minimize the approximation loss $\mathcal{L}$ of the model; iv) prove that our BDL algorithm converges to the stable state in which the value U^n of player U_n is optimal and the pure strategies of all players are in the myopic PBE of game $\mathbf{\Gamma}$.

The proposed BDL algorithm is unsupervised, i.e., does not require a prior "off-line" training and/or pre-existing datasets. The algorithm exploits a feed-forward BNN model $f^{\mathbf{W}}(\mathbf{x})$ to approximate unobservable state transitions of the POMDP of every player/miner U_n. At any stage t, the dataset $(\mathbf{X}_t,\ \mathbf{Y}_t) = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=t-T,\ldots,t-1}$ represents the history about $T = \min(t, F)$ past consecutive data points $(\mathbf{x}_i, \mathbf{y}_i)$, where F is the size of the observation window. A row input vector $\mathbf{x}_i = (x_i^0, \ldots, x_i^3) = (a_i^n, s_i^n) \in \mathbb{R}^{1\times D_I}$ consists of $D_I = 3$ elements that define the player's action $a_i^n = (b_i^n, x_i^n) \in \mathbb{R}^{1\times 2}$ and state $s_i^n = r_{i-1}^n \in \mathbb{R}$ observable at the beginning of stage i. The observed output $\mathbf{y}_i \in \mathbf{Y} = \{0,\ 1\}$ represents the reward $r_i^n \in \mathbf{R}^n = \{0, R + r\tau^n\}$ received by the player at the end of stage i, as in

$$\mathbf{y}_i = \begin{cases} 1, & r_\tau^n = R + r\tau^n, \\ 0, & r_\tau^n = 0. \end{cases} \tag{21}$$

Thus, we have a classification task where, given the dataset $(\mathbf{X}_t, \mathbf{Y}_t)$, we must predict the random output $f^{\mathbf{W}}(\mathbf{x}_t) = \widehat{\mathbf{y}}_t = (\hat{y}_t^0, \hat{y}_t^1) \in \mathbb{R}^{1\times D_O}$ of the BNN model with $D_O = 2$ elements being classified as 0 or 1. For a feed-forward BNN with $K \geq 1$ neurons and input vector $\mathbf{x}_t = (a_t^n, s_t^n) \in \mathbb{R}^{1\times D_I}$, the random

row output vector is given by [49]

$$f^{\mathbf{W}}(\mathbf{x}_t) = \varphi\Big(\mathbf{x}_t\mathbf{W}_{(h)}+\mathbf{b}\Big)\mathbf{W}^{\mathrm{T}}_{(v)} = \widehat{\mathbf{y}}_t = \Big(\hat{y}^0_t,\ \hat{y}^1_t\Big) \in \mathbb{R}^{1\times D_O}. \tag{22}$$

The weights in (22) represent the tuple $\mathbf{W} = (\mathbf{W}_{(h)}, \mathbf{W}_{(v)}, \mathbf{b}) \in \mathbb{R}^{(D_I+D_O+1)\times K}$, where $\mathbf{W}_{(h)} = (w^{i,j}_{(h)}) \in \mathbb{R}^{D_I\times K}$ and $\mathbf{W}_{(v)} = (w^{i,j}_{(v)}) \in \mathbb{R}^{D_O\times K}$ are, respectively, the weights in the hidden and visible layers; $\mathbf{b} = (b^j) \in \mathbb{R}^{1\times K}$ are the biases; $\varphi(\cdot)$ is the logistic sigmoid function. Similar to the prior research (e.g., [49]–[51]), we consider a standard normal prior distribution over the weights $\mathbf{W}$, i.e., $\Pr\{\mathbf{W}\} = \mathcal{N}(0, I)$).

As such, the prior weight distribution $\Pr\{\mathbf{W}\}$ approximates a prior belief B^n_t of player U_n about unobservable actions $\mathbf{a}^{-n}$ of other players, i.e., $B^n_t \sim \Pr\{\mathbf{W}\}$. Accordingly, the likelihood of each data point $(\mathbf{x}_t, \mathbf{y}_t)$, given by

$$\Pr\Big\{\mathbf{y}_t \ \Big|\ f^{\mathbf{W}}(\mathbf{x}_t)\Big\} = \begin{cases} \mathrm{Softmax}\big(0, f^{\mathbf{W}}(\mathbf{x}_t)\big) = \frac{e^{\hat{y}^0_t}}{e^{\hat{y}^0_t}+e^{\hat{y}^1_t}}, \mathbf{y}_t = 0, \\ \mathrm{Softmax}\big(1, f^{\mathbf{W}}(\mathbf{x}_t)\big) = \frac{e^{\hat{y}^1_t}}{e^{\hat{y}^0_t}+e^{\hat{y}^1_t}}, \mathbf{y}_t = 1 \end{cases} \tag{23}$$

is equivalent to the probability $\Pr\{r^n_t | s^n_t, a^n_t, B^n_t\}$ of reward r^n_t given the action a^n_t executed by player U_n in state $s^n_t = r^n_{t-1}$ with belief B^n_t, i.e., $\Pr\{r^n_t | s^n_t, a^n_t, B^n_t\} \sim \Pr\{\mathbf{y}_t | f^{\mathbf{W}}(\mathbf{x}_t)\}$. Then, the expected stage payoff $U^n(B^n_t, a^n | s^n_t)$ of the player can be approximated by the function $U^n(\mathbf{x}_t|\mathbf{W})$, as in (24), shown at the bottom of the page, where the player's payoff $u^n(s^n_t, a^n_t, s^n_{t+1}) = u^n(s^n_t, a^n_t, r^n_t)$ is presented explicitly as a function of $(\mathbf{x}_t, \mathbf{y}_t) = (s^n_t, a^n_t, r^n_t)$.

Furthermore, note that in the BRL algorithm, the beliefs are updated with Bayesian inference. That is, at any stage t, given the prior belief $B^n_t \sim \Pr\{\mathbf{W}\}$, we estimate a posterior $B^{s^n_t, s^n_{t+1}}_{a^n_t} = B^{s^n_t, r^n_t}_{a^n_t} \sim \Pr\{\mathbf{W}|\mathbf{x}_t, \mathbf{y}_t\}$ for an observed transition $(s^n_t, a^n_t, s^n_{t+1}) = (s^n_t, a^n_t, r^n_t) = (\mathbf{x}_t, \mathbf{y}_t)$ according to Bayes' rule in (10). As such, this Bayesian inference is equivalent to the estimation of a posterior $\Pr\{\mathbf{W}|\mathbf{x}_t, \mathbf{y}_t\}$. That is,

$$B^{s^n_t, s^n_{t+1}}_{a^n_t} = \frac{\Pr\Big\{s^n_t \ \Big|\ s^n_{t-1}, a^n_{t-1}, \mathbf{a}^{-n}\Big\} B^n\big(\mathbf{a}^{-n}\big)}{\sum_{\acute{\mathbf{a}}^{-n}\in\mathbf{A}^{-n}} \Pr\Big\{s^n_t \ \Big|\ s^n_{t-1}, a^n_{t-1}, \acute{\mathbf{a}}^{-n}\Big\} B^n(\acute{\mathbf{a}}^{-n})} = \frac{\Pr\big\{r^n_t \mid s^n_{t-1}, a^n_{t-1}, \mathbf{a}^{-n}\big\} B^n\big(\mathbf{a}^{-n}\big)}{\sum_{\acute{\mathbf{a}}^{-n}\in\mathbf{A}^{-n}} \Pr\big\{r^n_t \mid s^n_{t-1}, a^n_{t-1}, \acute{\mathbf{a}}^{-n}\big\} B^n(\acute{\mathbf{a}}^{-n})}$$

$$\sim \Pr\{\mathbf{W} \mid \mathbf{x}_t, \mathbf{y}_t\} = \frac{\Pr\big\{\mathbf{y}_t \mid f^{\mathbf{W}}(\mathbf{x}_t)\big\}\Pr\{\mathbf{W}\}}{\Pr\{\mathbf{y}_t \mid \mathbf{x}_t\}} = \frac{\Pr\big\{\mathbf{y}_t \mid f^{\mathbf{W}}(\mathbf{x}_t)\big\}\Pr\{\mathbf{W}\}}{\int \Pr\big\{\mathbf{y}_t \mid f^{\mathbf{W}}(\mathbf{x}_t)\big\}\Pr\{\mathbf{W}\}\mathrm{d}\mathbf{W}}, \tag{25}$$

which is intractable, since the marginal probability distribution $\Pr\{\mathbf{y}_t|\mathbf{x}_t\} = \int \Pr\{\mathbf{y}_t|f^{\mathbf{W}}(\mathbf{x}_t)\}\Pr\{\mathbf{W}\}\mathrm{d}\mathbf{W}$ cannot be evaluated analytically. Therefore, instead of direct inference, we utilize variational inference and, instead of integrating over weights $\mathbf{W}$, we find the parameters $\overline{\theta}$ of the approximating distribution q^θ which minimize a KL divergence in (16b) or, equivalently, maximize the ELBO in (17), so that $\Pr\{\mathbf{W}|\mathbf{x}_t, \mathbf{y}_t\} \underset{t\to\infty}{\longrightarrow} \overline{q}^{\overline{\theta}}(\mathbf{W})$.

To estimate variational inference, we can employ PDE, in which case we re-parameterize an approximating distribution q^θ by the parameter-free distribution $q(\epsilon)$ subject to $\widehat{\mathbf{W}}_t = g(\theta, \epsilon)$. Then, assuming a Gaussian distribution $q^\theta(\mathbf{W}) = \mathcal{N}(\mu, \sigma^2)$, for $\theta = (\mu, \sigma)$, $\widehat{\mathbf{W}}_t = g(\theta, \epsilon) = \mu + \sigma\epsilon$ and $q(\epsilon) = \mathcal{N}(0, I)$, the BDL problem is to find the parameters $(\overline{\theta}, \overline{\lambda})$ that minimize the classification loss $\mathcal{L}(\theta, \lambda)$, given by

$$\mathcal{L}(\theta, \lambda) = -\sum_{i=t-T}^{t-1} \Big(\mathbf{1}_{\mathbf{y}_i=0}\log\Big(\mathrm{Softmax}\Big(0, f^{g(\theta,\epsilon)}(\mathbf{x}_i)\Big)\Big) + \mathbf{1}_{\mathbf{y}_i=1}\log\Big(\mathrm{Softmax}\Big(1, f^{g(\theta,\epsilon)}(\mathbf{x}_i)\Big)\Big)\Big) + \frac{1}{2}(1-\lambda)\|\theta\|^2. \tag{26}$$

This problem can be solved with a stochastic gradient descent [49], [50]. That is, at any stage t, given the dataset $(\mathbf{X}_t,\ \mathbf{Y}_t)$, we minimize the current loss $\mathcal{L}_t = \mathcal{L}(\theta_t, \lambda_t)$ by updating the parameters (θ_t, λ_t) in the direction of a negative gradient, as

$$\theta_{t+1} = \theta_t - \eta_t\nabla_{\theta_t}\mathcal{L}_t \text{ and } \lambda_{t+1} = \lambda_t - \eta_t\nabla_{\lambda_t}\mathcal{L}_t, \tag{27}$$

until (θ_t, λ_t) converges, where η_t is a learning rate of gradient descent, such that $\sum_{t=1}^{\infty}\eta_t = \infty$ and $\sum_{t=1}^{\infty}\eta_t^2 < \infty$, typically set as $\eta_t = 1/(t+\eta_1)^{\eta_2}, \forall\eta_1 \in (0,1), \eta_2 \in (0,1)$.

C. Practical Bayesian Deep Learning Algorithm for Miners

Based on the above, we design a practical BDL algorithm for each player/miner U_n in game $\mathbf{\Gamma}$ where, at any stage t, we deploy the parameter $\theta_t = (\theta_{t(h)}, \theta_{t(v)}, \theta_{t(\mathbf{b})}) \in \mathbb{R}^{(D_I+D_O+1)\times K}$ representing the tuple formed by elements $\theta_{t(h)} = (\theta^{i,j}_{t(h)}) \in \mathbb{R}^{D_I\times K}$, $\theta_{t(v)} = (\theta^{i,j}_{t(v)}) \in \mathbb{R}^{D_O\times K}$ and $\theta_{t(\mathbf{b})} = (\theta^{j}_{t(\mathbf{b})}) \in \mathbb{R}^{1\times K}$. The elements $\theta_{t(h)}$ and

$$U^n(B^n_t, a^n \mid s^n_t) = \sum_{s^n_{t+1}\in\mathbf{S}^n} \Pr\big\{s^n_{t+1} \ \big|\ s^n_t, a^n_t, B^n_t\big\} u^n\big(s^n_t, a^n_t, s^n_{t+1}\big) = \sum_{r^n_t\in\mathbf{R}^n} \Pr\{r^n_t \mid s^n_t, a^n_t, B^n_t\} u^n(s^n_t, a^n_t, r^n_t) \sim U^n(\mathbf{x}_t \mid \mathbf{W}) = \sum_{\mathbf{y}_t\in\mathbf{Y}} \Pr\Big\{\mathbf{y}_t \ \Big|\ f^{\mathbf{W}}(\mathbf{x}_t)\Big\} u^n(\mathbf{x}_t, \mathbf{y}_t) \tag{24}$$

$\theta_{t(v)}$ define the approximated weights $\widehat{\mathbf{W}}_{(h)} \in \mathbb{R}^{D_I \times K}$ and $\widehat{\mathbf{W}}_{(v)} \in \mathbb{R}^{D_O \times K}$; the element $\theta_{t(\mathbf{b})}$ defines the approximated biases $\widehat{\mathbf{b}} \in \mathbb{R}^{1\times K}$ of the BNN. Accordingly, we re-parameterize the Gaussian approximating distribution $q^{\theta}(\mathbf{W}_t) = \prod_{i,j} q^{\theta^{i,j}_{(h)}}(w^{i,j}_{t(h)}) \prod_{i,j} q^{\theta^{i,j}_{(v)}}(w^{i,j}_{t(v)}) \prod_j q^{\theta^{i}_{(\mathbf{b})}}(b^j_t)$ by the standard normal parameter-free distribution

$$q(\epsilon_t) = \prod_{i,j} q\Big(\epsilon^{i,j}_{t(h)}\Big) \prod_{i,j} q\Big(\epsilon^{i,j}_{t(v)}\Big) \prod_{j} q\Big(\epsilon^{j}_{t}\Big) \tag{28a}$$

$$\text{subject to:} \begin{cases} \hat{w}^{i,j}_{(h)} = g\Big(\theta^{i,j}_{t(h)}, \epsilon^{i,j}_{t(h)}\Big), \forall i \in \{0,\ldots,\ D_I-1\}, \\ \hat{w}^{i,j}_{(v)} = g\Big(\theta^{i,j}_{t(v)}, \epsilon^{i,j}_{t(v)}\Big), \forall i \in \{0, D_O-1\}, \\ \hat{b}^{j} = g\Big(\theta^{j}_{t(\mathbf{b})}, \epsilon^{i}_{t(\mathbf{b})}\Big), \end{cases} \tag{28b}$$

for $\epsilon_t = (\epsilon_{t(h)}, \epsilon_{t(v)}, \epsilon_{t(\mathbf{b})}) \in \mathbb{R}^{(D_I+D_O+1)\times K}$, $\epsilon_{t(h)} = (\epsilon^{i,j}_{t(h)}) \in \mathbb{R}^{D_I\times K}$, $\epsilon_{t(v)} = (\epsilon^{i,j}_{t(v)}) \in \mathbb{R}^{D_O\times K}$, $\epsilon_{t(\mathbf{b})} = (\epsilon^{j}_{t(\mathbf{b})}) \in \mathbb{R}^{1\times K}$, and $j \in \{0,\ldots,K-1\}$.

As a result, we obtain the algorithm where at every stage *t*, we store the last-updated parameters $(\theta_{t+1}, \lambda_{t+1})$ and a dataset $(\mathbf{X}_{t+1},\ \mathbf{Y}_{t+1})$. At stage $t = 0$, we initialize the parameter θ_0 according to $\Pr\{\theta_0\} = \Pr\{\widehat{\mathbf{W}}\} = \mathcal{N}(0, I)$. That is, we draw the random samples $\theta^{i,j}_{0(h)}$, $\theta^{0,j}_{0(v)}$, $\theta^{1,j}_{0(v)}$, $\theta^{j}_{0(\mathbf{b})}$, $i \in \{0,\ldots,\ D_I-1\}$, $j \in \{0,\ldots,\ K-1\}$, from the standard normal distribution. At stage $t = 0,\ldots,T$, we repeat the following steps.

Step 1: Generate parameter ϵ_t according to $(\epsilon) = \mathcal{N}(0, I)$, i.e., draw random samples $\epsilon^{i,j}_{0(h)}$, $\epsilon^{0,j}_{0(v)}$, $\epsilon^{1,j}_{0(v)}$, $\epsilon^{j}_{0(\mathbf{b})}$, for $i \in \{0,\ldots,\ D_I-1\}, j \in \{0,\ldots,K-1\}$, from the standard normal distribution. Given parameters (θ_t, ϵ_t), estimate the weights $\widehat{\mathbf{W}} = g(\theta_t, \epsilon_t)$ based on (28b).

Step 2: Observe the current state $s^n_t = r^n_{t-1}$ and select a pure strategy $\alpha^n(\theta_t, s^n_t) = \text{argmax}_{a^n\in\mathbf{A}^n} U^n(a^n|s^n_t, g(\theta_t,\epsilon_t))$ that maximizes the approximated expected stage payoff:

$$\begin{aligned} U^n(a^n \mid s^n_t, g(\theta_t,\epsilon_t)) &= U^n\Big(a^n \;\Big|\; s^n_t, \widehat{\mathbf{W}}\Big) \\ &= \sum_{\mathbf{y}_t\in\mathbf{Y}} \Pr\Big\{\mathbf{y}_t \;\Big|\; f^{\widehat{\mathbf{W}}}(s^n_t, a^n)\Big\} u^n(s^n_t, a^n, \mathbf{y}_t) \end{aligned} \tag{29}$$

with $f^{\widehat{\mathbf{W}}}(s^n_t, a^n) = f^{g(\theta_t,\epsilon_t)}$ and $\Pr\{\mathbf{y}_t|f^{\widehat{\mathbf{W}}}(s^n_t, a^n)\}$ defined in (22) and (23), respectively.

Step 3: Execute action $a^n_t = \alpha^n(\theta_t, s^n_t)$, observe the reward r^n_t, and update a dataset as $(\mathbf{X}_{t+1},\ \mathbf{Y}_{t+1}) = (\mathbf{X}_t,\ \mathbf{Y}_t) \cup (\mathbf{x}_t,\ \mathbf{y}_t) \setminus (\mathbf{x}_{t-T},\ \mathbf{y}_{t-T})$, where $(\mathbf{x}_t,\ \mathbf{y}_t) = (s^n_t, a^n_t, r^n_t)$ and the parameters (θ_t, λ_t) in the negative gradient direction as in (27a).

Note that the above algorithm is unsupervised, i.e., it allows "on-line" learning that does not need a prior "off-line" training and/or pre-existing training datasets. This algorithm does not require any additional exploration, because it is implicit in the computation of the random BNN output $f^{\widehat{\mathbf{W}}}(s^n_t, a^n)$ in (22). Proposition 3 below shows that the proposed BDL algorithm that represents a sequence of iterations $\{U^n_t\}_{t\in\mathbb{N}}$, with $U^n_t = U^n(a^n|s^n_t, g(\theta_t,\epsilon_t))$, converges to an optimal value $\overline{U}^n$, i.e., the expected stage payoff of the player from its optimal pure strategy $\overline{\alpha}^n$.

Proposition 3: The BDL algorithm defined by sequence of iterations $\{U^n_t\}_{t\in\mathbb{N}}$, for all $n \in \mathbf{N}$, converges to the optimal value $\overline{U}^n$ with probability one as time tends to infinity.

The proof of Proposition 3 is given in Appendix F in the supplementary material. From Proposition 3, we obtain Corollary 2 that establishes that our BDL algorithm converges to a stable state where the players' pure strategies are in the MPBE of game Γ.

Corollary 3: The BDL algorithm defined by the sequence of iterations $\{U^n_t\}_{t\in\mathbb{N}}$, for all $n \in \mathbf{N}$, converges to a stable state in which the players' pure strategies and beliefs $(\overline{\boldsymbol{\alpha}}, \overline{\mathbf{B}})$ are in the myopic perfect Bayesian equilibrium of game $\boldsymbol{\Gamma}$.

The proof of Corollary 2 is given in Appendix G in the supplementary material. Finally, in Proposition 4 below, we analyze the worst-case computational complexity and the rate of convergence of the BDL algorithm.

Proposition 4: The BDL algorithm defined by the sequence of iterations $\{U^n_t\}_{t\in\mathbb{N}}$, for all $n \in \mathbf{N}$, yields the worst-case time complexity of $\mathcal{O}(K(c+T))$, where $c = |\max_{n\in\mathbf{N}} \mathbf{A}^n|$, *K* is the number of neurons and *T* is the number of data points in the BNN model, and the convergence rate of $\mathcal{O}(1/t^{1-\frac{1}{2-\lambda}})$, where $\lambda \in [0,\ 1)$ is the average dropout rate.

The proof of Proposition 4 is given in Appendix H in the supplementary material. From Proposition 4, the convergence rate of the BDL algorithm is sublinear with order $1-\frac{1}{2-\lambda} \in (0,\ 0.5]$ and its complexity is polynomial. In particular, the algorithm complexity depends on the size of the BNN model defined by the product *KT* of the number of neurons *K* and number of data point *T*.

VI. Performance Evaluation

We simulate the blockchain application realized in the long-term evolution advanced (LTE-A) time division duplex (TDD) [52] based MEC network. A simulation model of the network is implemented with the OPNET development package [53]. The model comprises $M = 3$ BSs represented by the macro-cell LTE-A evolved NodeB (eNB) labeled as BS_1, a micro-cell eNB labeled as BS_2, and a femto-cell eNB labeled as BS_3, that are placed as depicted in Figure 4. The default number of miners is $N = 10$. Other parameters of the simulation model are specified in Appendix I in the supplementary material.

Note that the most efficient learning algorithm returns the highest miners' payoffs in a shortest time with the smallest number of computations. Hence, in the following, we evaluate the efficiency of the proposed BRL and BDL algorithms w.r.t. payoffs of the miners, convergence time and complexity under different simulation settings, and compare their performance with that of two state-of-the-art on-line DL models:

1) *UDL* - unsupervised DL algorithm (described, e.g., in [41]) where at any stage *t*, miner U_n selects a strategy $\alpha^n(\omega_t, s^n_t) = \text{argmax}_{a^n\in\mathbf{A}^n} U^n(a^n|s^n_t, \omega_t)$ maximizing its expected stage payoff $U^n(a^n|s^n_t, \omega_t)$ which is approximated by the NN with deterministic weights ω_t. The goal is to minimize the loss, i.e., distance between $U^n(a^n|s^n_t, \omega_t)$ and

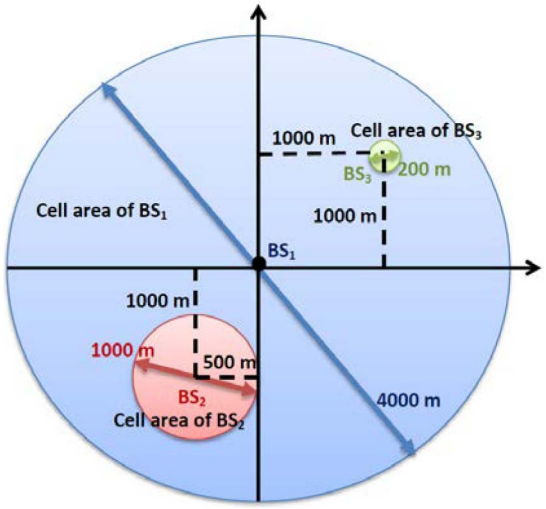

Fig. 4. Placement of BSs in the model: service area of the MEC network coincides with the macro-cell area, i.e., BS_1 is placed in the center; the diameters of BS_1, BS_2, and BS_3 are 4000, 1000, and 200 m, respectively.

approximation target represented by the optimal expected stage payoffs $\overline{U}^n$.

2) *DQL* - deep Q-learning algorithm (described, e.g., in [14], [41], [48]), where miner U_n selects a strategy $\alpha^n(\omega_t, s_t^n) = \text{argmax}_{a^n \in \mathbf{A}^n} V^n(a^n|s_t^n, \omega_t)$ that maximizes its approximated expected long-term payoff $V^n(a^n|s_t^n, \omega_t)$. Thus, unlike UDL where miners are myopic, here the miners are long-visional.

We also benchmark the performance of the algorithms with the performance of following game models:

1) *CI* - complete information game, where miner U_n knows current actions $\mathbf{a}_t^{-n}$ of other miners. Thus, no miner U_n needs to learn the actions of its opponents. Instead, at any stage *t*, it selects an action $\overline{a}_t^n$ that maximizes its expected one-stage payoff $U^n(a^n|s_t^n, \mathbf{a}_t^{-n})$. As such, the miners' actions $\overline{\mathbf{a}}_t$ form a Nash equilibrium (NE). Note that the CI scenario is infeasible in mobile blockchains. Thus, we use it only for benchmarking purposes.

2) *NL* - game with no learning under incomplete information, where assumptions are the same as in BRL and BDL, i.e., no miner U_n knows the actions $\mathbf{a}_t^{-n}, \ldots, \mathbf{a}_0^{-n}$ of other miners, but no learning is adopted. As such, each miner U_n operates under some fixed initial beliefs B_0^n about unobservable actions. At any stage *t*, the miner selects an action $\overline{a}_t^n$ that maximizes its expected one-stage payoff $U^n(B_0^n, a^n|s_t^n)$, i.e., miners' actions $\overline{\mathbf{a}}_t$ form a BNE.

Figure 5 shows dynamics of the average miner's payoff in all simulated learning/game models under default parameters during the period $t \in \{30, \ldots, 70\}$ stages. Note that the average payoff is stable in CI and NL where the miners' actions form a NE and a BNE, respectively. Results also show that payoffs achieved in UDL, DQL, BRL and BDL grow consistently with time converging to some stable levels after about 70, 65, 55 and 60 stages, respectively. Yet, the stable-state values of the payoffs in BRL and BDL are higher than those in UDL and DQL. In particular, in BRL and BDL, these values approach the optimum (i.e., a NE payoff achieved in CI where

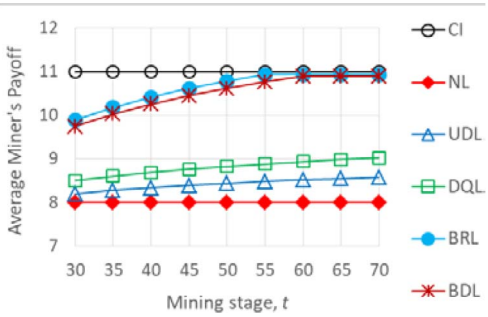

Fig. 5. Average miner's payoff vs. mining stage, *t*.

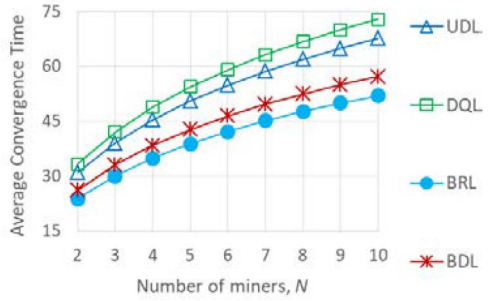

Fig. 6. Convergence time vs. number of miners, *N*.

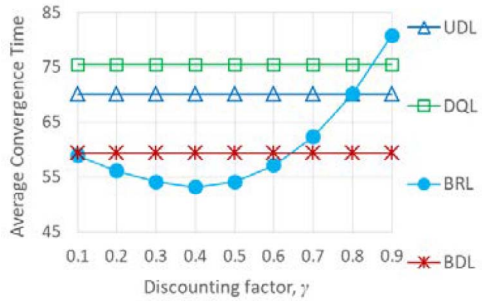

Fig. 7. Convergence time vs. discounting factor, γ.

miners are fully-informed about actions of other miners). In UDL and DQL, these values are closer to that in NL (i.e., a BNE payoff received under fixed miners' beliefs with no learning adopted) and comprise less than 83% of the optimal value. The reason for such poor performance of UDL and DQL is that they are based on the conventional NN models. Hence, although they can produce the accurate approximations of fully-observable MDPs, they are not efficient for POMDPs, as they do not have any means to track epistemic uncertainties about unobservable states of POMDPs where these uncertainties are represented by the beliefs [49]–[51]. Accordingly, although both UDL and DQL converge to the stable states, the values obtained in these states are far from the optimal one and closer to that received under fixed miner's beliefs.

Figure 6 shows the convergence times of different learning algorithms plotted as functions of the number of miners *N*. From Figure 6, BRL converges faster than BDL, UDL and DQL. The reason is that in BDL, UDL and DQL, the loss $\mathcal{L}$ is non-convex w.r.t. weights of the BNN/NN models. Hence, in general, a stochastic gradient descent applied to non-convex function goes through several local minima until it reaches

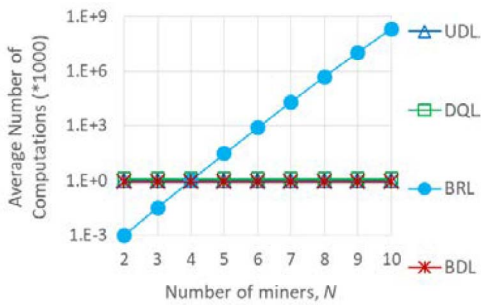

Fig. 8. Average complexity in log-scale vs. number of miners, *N*.

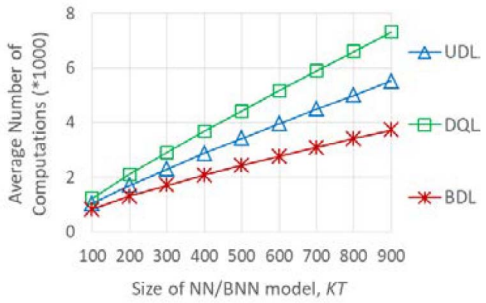

Fig. 9. Average complexity vs. size of NN/BNN model, *KT*.

a global optimum, which increases the total convergence time of BDL, UDL and UDL. Moreover, from [14], the convergence rate of UDL and DQL depends on the weights of the NN model which are optimized during training and, hence, cannot be predicted before we train the model. Similarly, from Proposition 4, the convergence rate of BDL is $\mathcal{O}(1/t^{1-\frac{1}{2-\lambda}})$, i.e., it depends on the average dropout rate λ that is optimized together with the weights of the BNN. That is, the value of λ cannot be specified in advance. As such, exact convergence times of BDL, UDL and DQL cannot be determined prior to training the model. They can only be estimated (e.g., based on the training records) after the algorithms converge. On the contrary, from Proposition 2, the convergence rate of BRL is $\mathcal{O}(\sqrt{\log(\log t)/t})$, for $\gamma \leq 0.5$ and $\mathcal{O}(1/t^{1-\gamma})$, for $\gamma > 0.5$, i.e., it depends only on the predefined value of a discounting factor γ. This dependency is demonstrated in Figure 7 that shows convergence times of different algorithms as functions of γ. From Figure 7, for $\gamma > 0.85$, the convergence time of BRL is larger than those of BDL, UDL and DQL for $\gamma > 0.85$; for $\gamma \in [0.1,\ 0.65]$, the convergence time of BRL is less than that of BDL, UDL and DQL; the minimal convergence time is achieved with $\gamma \approx 0.4$. As such, in BRL, it is possible to preselect the value of γ that yields the minimal convergence time, which is not possible to do in BDL, UDL and DQL.

Figures 8 and 9 show computational complexity of different learning algorithms measured in terms of average number of computations as functions of the number of miners *N* and the size *KT* of NN/BNN models, respectively. From Figure 8, the number of computations in BRL is exponential in the number of miners *N*. This conforms to the finding of Proposition 2 that the complexity of BRL is $\mathcal{O}(c^N)$. As such, although the BRL algorithm shows a superior performance, it is intractable for a large number of miners. On the other hand, from Figure 9,

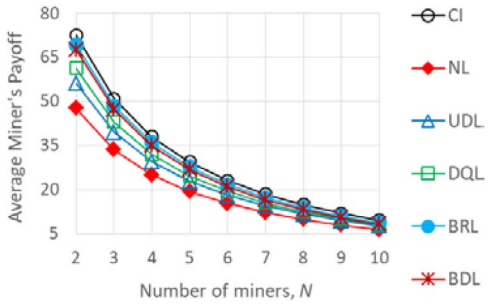

Fig. 10. Average miner's payoff vs. number of miners, *N*.

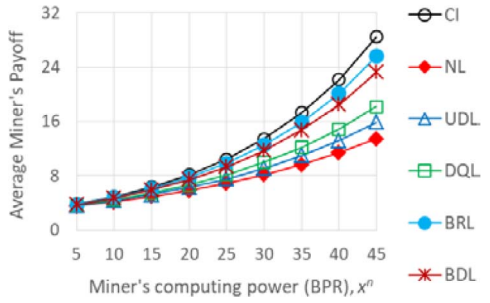

Fig. 11. Average miner's payoff vs. average computing power (BPR), x^n.

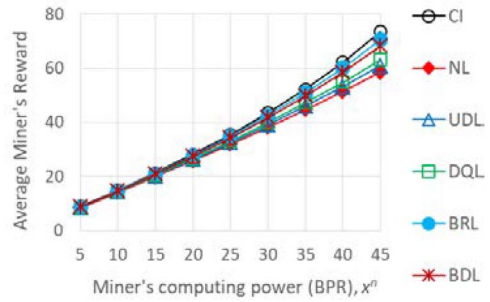

Fig. 12. Average miner's reward vs. average computing power (BPR) x^n.

the numbers of computations in UDL, DQL and BDL are linear in the product of the number of neurons *K* and number of data points *T* in the NN/BNN model. This conforms to the findings in [14] and Proposition 4 that the complexities of UDL and DQL are linear in *KT*, and complexity of BDL is $\mathcal{O}(K(c+T))$, respectively. Thus, due to its performance and scalability (w.r.t. number of miners), a BDL algorithm is preferable when the number of miners is large.

Figure 10 shows the impact of the number of miners *N* on their average payoff in the stable states in different learning/game models. From Figure 10, the average payoff decreases with the number of miners *N*. Indeed, when *N* is increasing, the ratio of miners that do not receive a reward is growing and, hence, the average miner's payoff is reducing. The impact of the computing power or BPR (in ECU) x^n selected by each miner U_n on its average stable-state payoff is shown in Figure 11. From Figure 11, the miner's payoff increases with x^n. The reason is that when x^n increases, the probability P_s^n that miner U_n is the first to generate the output (which is proportional to its relative BPR in (3b)) grows. This increases the chances that miner U_n wins a mining reward, as it follows from Figure 12 which shows the average stable-state reward

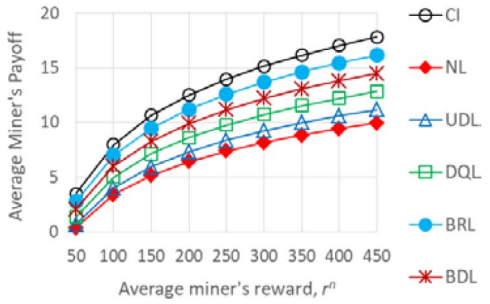

Fig. 13. Average miner's payoff vs. average reward, r^n.

of miner U_n as a function of its computing power or BPR x^n. Finally, Figure 13 shows the relationship between the reward r^n of miner U_n and its average payoff achieved in the stable states in different learning/game models. From Figure 13, the average miner's payoff increases with its reward r^n, which follows directly from the payoff expression in (5), i.e., the miner's payoff is the difference between its reward and the cost of its selected computing power.

VII. Conclusion

We have presented a novel Bayesian RL and DL framework for a stochastic game with incomplete information to model interactions among miners in a blockchain with MEC. Within the framework, we have formulated a BRL algorithm based on the POMDP model of the miners' decisions and developed a novel unsupervised BDL algorithm where uncertainties about unobservable states of POMDP are modeled with the BNN. We have proven that the proposed BRL and BDL algorithms converge to the stable states where the miners' actions form the PBE and MPBE, respectively.

References

[1] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: http://bitcoin.org/bitcoin.pdf

[2] D. C. Nguyen, P. N. Pathirana, M. Ding, A. Seneviratne, "Blockchain for 5G and beyond networks: A state of the art survey," Dec. 2019. [Online]. Available: arXiv:1912.05062.

[3] A. Asheralieva and D. Niyato, "Learning-based mobile edge computing resource management to support public blockchain networks," *IEEE Trans. Mobile Comput.*, early access, Dec. 16, 2019, doi: 10.1109/TMC.2019.2959772.

[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, 2017.

[5] W. Wang *et al.*, "A survey on consensus mechanisms and mining strategy management in blockchain networks," *IEEE Access*, vol. 7, pp. 22328–22370, 2019.

[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, Z. Xiong, D. Niyato, D. Ye, D. 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.

[8] Z. Xiong, Y. Zhang, D. Niyato, P. Wang, and Z. Han, "When Mobile Blockchain Meets Edge Computing," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 33–39, Aug. 2018.

[9] Z. Xiong, S. Feng, W. Wang, D. Niyato, P. Wang, and Z. Han, "Cloud/fog computing resource management and pricing for blockchain networks," *IEEE Internet 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. Emerg. Topics Comput. Intell.*, early access, Feb. 6, 2019, doi: 10.1109/TETCI.2019.2892733.

[11] 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 Interet Things J.*, vol. 6, no. 5, pp. 8753–8769, Oct. 2019.

[12] R. Yang, 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, 2nd Quart., 2019.

[13] N. Houy, "The bitcoin mining game," *Ledger*, vol. 1, pp. 53–68, Dec. 2016. [Online]. Available: https://ledgerjournal.org/ojs/index.php/ledger/ article/view/13

[14] A. Asheralieva and D. Niyato, "Distributed dynamic resource management and pricing in the IoT systems with blockchain-as-a-service and UAV-enabled mobile edge computing," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 1974–1993, Mar. 2020.

[15] K. Kotobi and S. G. Bilen, "Secure blockchains for dynamic spectrum access: A decentralized database in moving cognitive radio networks enhances security and user access," *IEEE Veh. Technol. Mag.*, vol. 13, no. 1, pp. 32–39, Mar. 2018.

[16] M. Grissa, A. A. Yavuz, and B. Hamdaoui, "TrustSAS: A trustworthy spectrum access system for the 3.5 GHz CBRS band," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Paris, France, 2019, pp. 1495–1503.

[17] S. Bayhan, A. Zubow, P. Gaw, and A. Wolisz, "Smart contracts for spectrum sensing as a service," *IEEE Trans. Cogn. Commun. Netw.*, vol. 5, no. 3, pp. 648–660, Sep. 2019.

[18] S. Bayhan, A. Zubow, and A. Wolisz, "Spass: Spectrum sensing as a service via smart contracts," in *Proc. IEEE Int. Symp. Dyn. Spectr. Access Netw. (DySPAN)*, Seoul, South Korea, 2018, pp. 1–10.

[19] F. den Hartog, F. Bouhafs, and Q. Shi, "Toward secure trading of unlicensed spectrum in cyber-physical systems," in *Proc. 16th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Las Vegas, NV, USA, 2019, pp. 1–4.

[20] M. Condoluci and T. Mahmoodi, "Softwarization and virtualization in 5G mobile networks: Benefits, trends and challenges," *Comput. Netw.*, vol. 146, pp. 65–84, Dec. 2018.

[21] G. A. F. Rebello *et al.*, "Providing a sliced, secure, and isolated software infrastructure of virtual functions through blockchain technology," in *Proc. IEEE 20th Int. Conf. High Perform. Switching Routing (HPSR)*, Xi'An, China, 2019, pp. 1–6.

[22] M. F. Franco, E. J. Scheid, L. Z. Granville, and B. Stiller, "BRAIN: Blockchain-based reverse auction for infrastructure supply in virtual network functions-as-a -service," in *Proc. IFIP Netw. Conf.*, Warsaw, Poland, 2019, pp. 1–9.

[23] F. D. Calabrese, L. Wang, E. Ghadimi, G. Peters, L. Hanzo, and P. Soldati, "Learning radio resource management in RANs: Framework, opportunities, and challenges," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 138–145, Sep. 2018.

[24] Y. Le, X. Ling, J. Wang, and Z. Ding, "Prototype design and test of blockchain radio access network," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Shanghai, China, 2019, pp. 1–6.

[25] D. Li, R. Du, Y. Fu, and M. H. Au, "Meta-key: A secure data-sharing protocol under blockchain-based decentralized storage architecture," *IEEE Netw. Lett.*, vol. 1, no. 1, pp. 30–33, Mar. 2019.

[26] S. Wang, Y. Zhang, and Y. Zhang, "A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems," *IEEE Access*, vol. 6, pp. 38437–38450, 2018.

[27] W. Liang, M. Tang, J. Long, X. Peng, J. Xu and K. Li, "A secure FaBric blockchain-based data transmission technique for industrial Internet-of-Things," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3582–3592, Jun. 2019.

[28] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Serv.*, vol. 14, no. 4, pp. 352–375, 2018.

[29] A. P. Joshi, M. Han, Y. Wang, "A survey on security and privacy issues of blockchain technology," *Math. Found. Comput.*, vol. 1, no. 2, pp. 121–147, May 2018.

[30] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A survey of distributed consensus protocols for blockchain networks," 2019. [Online]. Available: arXiv:1904.04098.

[31] E. Kokoris-Kogias, P. Jovanovic, L. Gasser,N. Gailly, E. Syta, and B. Ford, "OmniLedger: A secure, scale-out, decentralized ledger via

sharding," in *Proc. IEEE Symp. Security and Privacy (SP)*, San Francisco, CA, USA, 2018, pp. 583–598.
[32] H. Yoo, J. Yim, and S. Kim, "The blockchain for domain based static sharding," in *Proc. 17th IEEE Int. Conf. Trust Security Privacy Comput. Commun./ 12th IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE)*, New York, NY, USA, 2018, pp. 1689–1692.
[33] S. Forestier, D. Vodenicarevic, and A. Laversanne-Finot "Blockclique: scaling blockchains through transaction sharding in a multithreaded block graph," 2018. [Online]. Available: arXiv:1803.09029.
[34] M. H. Manshaei, M. Jadliwala, A. Maiti, and M. Fooladgar, "A game-theoretic analysis of shard-based permissionless blockchains," *IEEE Access*, vol. 6, pp. 78100–78112, 2018.
[35] S. Li, M. Yu, C.-S. Yang, A. S. Avestimehr, S. Kannan, and P. Viswanath, "PolyShard: Coded sharding achieves linearly scaling efficiency and security simultaneously," Sep. 2018. [Online]. Available: arXiv:1809.10361.
[36] Y. Jiao, P. Wang, D. Niyato, and Z. Xiong, "Social welfare maximization auction in edge computing resource allocation for mobile blockchain," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kansas City, MO, USA, May 2018, pp. 1–6.
[37] N. C. Luong, Z. Xiong, P. Wang, and D. Niyato, "Optimal auction for edge computing resource management in mobile blockchain networks: A deep learning approach," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kansas City, MO, USA, May 2018, pp. 1–6.
[38] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Distributed resource allocation in blockchain-based video streaming systems with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 695–708, Jan. 2019.
[39] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Computation offloading and content caching in wireless blockchain networks with mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11008–11021, Nov. 2018.
[40] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge MA, USA: MIT Press, 1998, pp. 1–356.
[41] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Artificial neural networks-based machine learning for wireless networks: A tutorial," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3039–3071, 4th Quart., 2019.
[42] A. Asheralieva and Y. Miyanaga, "An autonomous learning-based algorithm for joint channel and power level selection by D2D pairs in heterogeneous cellular networks," *IEEE Trans. Commun.*, vol. 64, no. 9, pp. 3996–4012, Sep. 2016.
[43] A. Asheralieva and D. Niyato, "Game theory and Lyapunov optimization for cloud-based content delivery networks with device-to-device and UAV-enabled caching," *IEEE Trans. Veh. Technol.*, vol. 68, no. 10, pp. 10094–10110, Oct. 2019.
[44] M. J. Osborne and A. Rubenstein, *A Course in Game Theory*. Cambridge MA, USA: MIT Press, 1994, pp. 1–374.
[45] 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.
[46] M. Ghavamzadeh, S. Mannor, J. Pineau, and A. Tamar, "Bayesian reinforcement learning: A survey," *Found. Trends Mach. Learn.*, vol. 8, nos. 5–6, pp. 359–483, 2016.
[47] P. Poupart, N. Vlassis, J. Hoey, and K. Regan, "An analytic solution to discrete Bayesian reinforcement learning," in *Proc. ACM 23rd Int. Conf. Mach. Learn. (ICML)*, Jun. 2006, pp. 697–704.
[48] Y. Sun, M. Peng, and S. Mao, "Deep reinforcement learning-based mode selection and resource management for green fog radio access networks," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1960–1971, Apr. 2019.
[49] Y. Gal, "Uncertainty in deep learning," Ph.D. dissertation, Dept. Eng., Univ. Cambridge, Cambridge, U.K, 2016.
[50] M. Segù, A. Loquercio, D. Scaramuzza, "A general framework for uncertainty estimation in deep learning," Jul. 2019. [Online]. Available: arXiv:1907.06890.
[51] A. Y. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" in *Proc. 31st Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 5580–5590.
[52] *Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2, Release 13*, 3GPP Standard TS 36.300, 2016.
[53] *OPNET Simulation and Development Tool*. Accessed: May 2020. [Online]. Available: http://www.opnet.com
[54] C. Szepesvári, "The asymptotic convergence-rate of Q-learning," in *Proc. 10th Int. Conf. Adv. Neural Inf. Process. Syst.*, 1998, pp. 1064–1070.
[55] M. Hardt, B. Recht, Y. Singer, "Train faster, generalize better: Stability of stochastic gradient descent," Sep. 2015. [Online]. Available: arXiv:1509.01240.
[56] 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. 483–502, 2002.

Alia Asheralieva received the B.S. degree from Kyrgyz Technical University in 2004, the M.E. degree from the Asian Institute of Technology, Thailand, in 2007, and the Ph.D. degree from the University of Newcastle, Australia, in 2015. From 2015 to 2016, she was a Research Assistant Professor with the Graduate School of Information Science and Technology, Hokkaido University, Japan. In 2017, she was a Postdoctoral Research Fellow with 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, Southern University of Science and Technology, China. Her main research interests span many areas of communications and networking, including cognitive radio networks, heterogeneous networks, D2D and IoT communications, cloud/edge/fog computing, mobile blockchains, cross-layer resource allocation and optimization, congestion control and routing, game theory, computational and artificial intelligence for wireless networks, as well as queuing theory, simulation and network modeling, QoS, and performance evaluation.

Dusit Niyato (Fellow, IEEE) received the B.Eng. degree from the King Mongkut's Institute of Technology Ladkrabang, Thailand, in 1999, and the Ph.D. 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.