Distributed Dynamic Resource Management and Pricing in the IoT Systems With Blockchain-as-a-Service and UAV-Enabled Mobile Edge Computing

Alia Asheralieva and Dusit Niyato, *Fellow, IEEE*

***Abstract*—In this article, we study the pricing and resource management in the Internet of Things (IoT) system with blockchain-as-a-service (BaaS) and mobile-edge computing (MEC). The BaaS model includes the cloud-based server to perform blockchain tasks and the set of peers to collect data from local IoT devices. The MEC model consists of the set of terrestrial and aerial base stations (BSs), i.e., unmanned aerial vehicles (UAVs), to forward the tasks of peers to the BaaS server. Each BS is also equipped with an MEC server to run some blockchain tasks. As the BSs can be privately owned or controlled by different operators, there is no information exchange among them. We show that the resource management and pricing in the BaaS-MEC system are modeled as a stochastic Stackelberg game with multiple leaders and incomplete information about actions of leaders/BSs and followers/peers. We formulate a novel hierarchical reinforcement learning (RL) algorithm for the decision makings of BSs and peers. We also develop an unsupervised hierarchical deep learning (HDL) algorithm that combines deep *Q*-learning (DQL) for BSs with the Bayesian deep learning (BDL) for peers. We prove that the proposed algorithms converge to stable states in which the peers' actions are the best responses to optimal actions of BSs.**

***Index Terms*—Bayesian methods, blockchain, deep learning, game theory, incomplete information, Internet of Things (IoT), Markov processes, mobile-edge computing (MEC), reinforcement learning (RL), resource allocation, unmanned aerial vehicles (UAVs).**

I. Introduction

Rapid proliferation of the Internet of Things (IoT) systems poses new challenges to data management and analytics, as vast volumes of sensing and control data generated by the IoT devices must be gathered, sorted, validated, and stored in a secure and decentralized way [1], [2]. Due to limited scalability, single points of failure, and absence of transparency, the centralized ledger approaches cannot meet these challenges [3]. Instead, a distributed ledger (DL) technology called blockchain has been recently proposed for scalable, secure, and transparent IoT data management [2], [3]. In blockchains, data are organized in the form of blocks, e.g., records of IoT transactions, as a linked list data structure to preserve logical relations in the appended blocks. Blocks are verified, copied, and distributed across the blockchain network—a connected system formed by the geo-scattered blockchain nodes or peers. This allows to store and process IoT data with built-in robustness, integrity, and reliability [4], [5].

Although blockchains have already been adopted in several distributed scenarios, e.g., connected vehicles and smart grids [6], [7] and content delivery networks [8], they are still not widely deployed in the IoT systems. The reason is that the typical IoT devices, e.g., small sensors, have scarce battery, computing, and caching resources. Thus, they cannot independently run many compute-intensive blockchain tasks, e.g., block processing and confirmation, and store transaction records [9]. To address this problem, along with leading tech giants, such as Microsoft, IBM, and Amazon, many startups are now offering a viable solution through the blockchain-as-a-service (BaaS) model [10], [11]. BaaS is a fusion of blockchain and cloud computing which allows its customers to leverage cloud-based solutions for blockchain applications. As such, a BaaS provider maintains the required infrastructure and resource/security management for a fee paid by its customers. Unfortunately, despite many benefits, there are several issues that restrict the adoption of BaaS in the IoT systems [10], [11]. First, in the current service model, the BaaS customer must be connected to a cloud where its data are managed by the core servers. This yields high propagation delay, backhaul load, and power consumption for the IoT devices which are typically located at the network "edges," i.e., far from the cloud. In addition, it may be unprofitable to perform spectrum/resource allocation for the large number of spatially distributed IoT

Manuscript received August 26, 2019; revised November 6, 2019 and November 29, 2019; accepted December 19, 2019. Date of publication December 24, 2019; date of current version March 12, 2020. This work was supported in part by the National Natural Science Foundation of China under Project 61950410603; in part by the Singapore NRF National Satellite of Excellence, Design Science and Technology for Secure Critical Infra-Structure NSoE under Grant DeST-SCI2019-0007; in part by the A*STAR-NTU-SUTD Joint Research Grant Call on Artificial Intelligence for the Future of Manufacturing RGANS 1906, WASP/NTU M4082187 (4080); in part by the Singapore MOE Tier 1 under Grant 2017-T1-002-007 RG122/17 and MOE Tier 2 under Grant MOE2014-T2-2-015 ARC4/15; in part by the Singapore NRF under Grant 2015-NRF-ISF001-2277; and in part by the Singapore EMA Energy Resilience under Grant NRF2017 EWT-EP003-041. *(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 article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author.

Digital Object Identifier 10.1109/JIOT.2019.2961958

devices, as the capital and operating expenditure (CAPEX and OPEX) on the necessary infrastructure can be overwhelming.

To tackle the above issues, the BaaS model can be integrated with mobile-edge computing (MEC) [12]–[15]. In this case, the spectrum/resource allocation can be implemented at the base stations (BSs) operated by some external, i.e., different from the BaaS provider, MEC operator(s). This will significantly reduce the expenses of the BaaS provider. Next, note that due to their mobility and ease of deployment, current MEC operators increasingly utilize unmanned aerial vehicles (UAVs) as aerial BSs [15]–[17]. Hence, it is reasonable to assume that some BSs can be represented by UAVs to extend connectivity and facilitate cost-effective solutions. Furthermore, to minimize backhaul load and propagation delays, the BaaS provider can deploy the set of blockchain peers represented, e.g., by mobile terminals and desktop computers, in proximity to the IoT devices. The peers can manage the IoT-blockchain data, e.g., gather, process, verify, and store, confirmed the IoT transaction records. Some compute-intensive tasks can also be offloaded to the core servers or to MEC servers installed at BSs.

In this article, we study the integration of BaaS with MEC, where MEC services are provided through terrestrial BSs and UAVs acting as aerial BSs. The main challenge here is how to provide the efficient BaaS support for IoT applications and, at the same time, maintain the profitability of the BaaS-MEC model. More specifically, each BS charges a certain fee for offloading/communication services from blockchain peers. To maximize its payoff, the BS must determine its optimal service fee and, in the case of aerial BS, its optimal flight parameters. On the other hand, the objective of a blockchain peer is to finish its block task before a given deadline at a smallest possible fee. In particular, the peer has the following options: 1) perform the task locally; 2) offload the task to the MEC server of some BS, in which case it must pay the processing fee to the respective BS; and 3) offload the task to the core server, in which case it must pay the forwarding fee to the BS that forwards the task. As such, we must analyze the interactions between the BSs and peers while also considering the following critical issues.

1) In mobile blockchains, communication among peers results in additional delay and energy overheads that yield growing latencies and power costs. To minimize data exchange, the peers should select their offloading options simultaneously and independently, without informing each other about their decisions. As such, peers have incomplete information about offloading decisions of other peers.
2) Due to the absence of communications among BSs, each BS selects its optimal service cost and (in the case of aerial BSs) its flight parameters simultaneously and independently. As such, similar to the peers, BSs have incomplete information about cost assignments and flight parameters of other BSs.
3) The state of a wireless environment (e.g., locations of peers and aerial BS, and channel quality) is highly stochastic. Thus, to maximize their payoffs in the BaaS-MEC model, the peers and BSs should be able to dynamically adjust their decisions in response to the random environmental changes.

Consequently, in this article, we introduce a novel game-theoretic, reinforcement learning (RL) and deep learning framework aimed to model the interactions between the BSs and peers, and to address the aforementioned issues. The main contributions of this article are listed as follows.

1) We formulate a detailed analytical model of the IoT system with BaaS-MEC support that considers dynamic parameters of the blockchain tasks and wireless channels. Based on this model, we derive exact expressions of the delay and energy for completing each task and the corresponding costs and payoffs of the BSs and peers.
2) We represent the interactions between BSs and peers as a stochastic Stackelberg game with multiple leaders, where both the leaders/BSs and followers/peers have incomplete information about the actions of other leaders and followers, respectively. The game is stochastic because the payoffs of leaders and followers are affected by random environmental states with unknown transitions.
3) As no follower observes the actions of other followers, we model its decision making by a partially observable Markov decision process (POMDP). We prove that a solution of the POMDP determines the best response strategies of followers which form a perfect Bayesian equilibrium (PBE). We also develop a Bayesian RL (BRL) algorithm that allows each follower to update its belief about unobservable states of the POMDP and predict its strategy through the interactions with the other followers and with the environment.
4) To reduce the complexity of the POMDP, we devise a novel polynomial-time Bayesian deep learning (BDL) algorithm in which uncertainties about unobservable states are modeled by a Bayesian neural network (BNN)—neural network (NN) with random weights distributed according to a predefined prior. The algorithm is unsupervised, as it enables a self-organized "online" learning that does not need a prior "offline" training and/or pre-existing training data sets. We prove that the BDL algorithm converges to a stable state in which the followers' strategies form a myopic PBE (MPBE)—the PBE with short-sighted players that aim to maximize their one-stage (rather than long-term) payoffs.
5) We derive a stochastic optimization problem for each leader of a Stackelberg game. The problem incorporates a random state defined by the followers' best responses. To solve this problem, we model it as a Markov decision process (MDP), the solution of which is equivalent to the solution of the leader's problem. Based on the MDP, we develop the RL algorithm that allows the leader to maximize its individual expected long-term payoff by dynamically adjusting its MEC service cost and flight parameters (in the case of aerial BSs).
6) We formulate a polynomial-time deep Q-learning (DQL) algorithm in which the complexity of the MDP is reduced by approximating the value function with the output of the NN in an unsupervised manner, i.e.,

TABLE I
LIST OF MAIN NOTATIONS USED IN THIS ARTICLE

Notation	Description
$\mathbf{N}$, $\mathbf{M}_\mathrm{T}$, $\mathbf{M}_\mathrm{A}$, $\mathbf{M}$	Set of N blockchain peers, set of M_T terrestrial BSs, set of M_A aerial BSs, and total set of M BSs, respectively
$\rho^0, \rho^n, \rho^m, \forall n \in \mathbf{N}, m \in \mathbf{M}$	Computing power of the blockchain server processor, processor of peer P$_n$, and MEC server of BS$_m$, respectively
$\varphi^0, \varphi^n, \varphi^m, \forall n \in \mathbf{N}, m \in \mathbf{M}$	Energies spent per CPU cycle by the blockchain server, peer P$_n$, and MEC server of BS$_m$, respectively
$t \in \{0, \dots, T\}$, Δt	Mining stage and stage duration (in seconds), respectively
$\mathbf{c}_t = \{c_t^m = (c_t^{m(I)}, c_t^{m(P)}) \in \mathbf{C}^m \mid m \in \mathbf{M}\}$	Stage costs for MEC services assigned by BSs, where c_t^m is the stage cost of BS$_m$, which includes forwarding cost $c_t^{m(I)}$ and processing cost $c_t^{m(P)}$, $\mathbf{C}^m = \{(0,0), \dots, (C_{max}^{m(I)}, C_{max}^{m(P)})\}$ is the set of possible costs, $C_{max}^{m(I)}$ and $C_{max}^{m(P)}$ are the maximal processing and forwarding costs, respectively
$\mathbf{l}_t^\mathrm{p} = \{l_t^n \in \mathbf{L}^n \subset \mathbf{R} \mid n \in \mathbf{N}\} \in \mathbf{L}^\mathrm{p}$	2D locations of peers at stage t, where l_t^n is the 2D location of peer P$_n$, $\mathbf{L}^n$ is the set of possible 2D locations of peer P$_n$, $\mathbf{R}$ is the network service area
$\mathbf{l}_t = \{l_t^m = (h_t^m, \ell_t^m) \in \mathbf{L}^m \mid m \in \mathbf{M}\} \in \mathbf{L}$	3D location of BSs at stage t, where l_t^m is the 3D location of BS$_m$, h_t^m is the altitude, ℓ_t^m is the projected (to the ground) 2D location, $\mathbf{L}^m$ is the set of possible 3D locations of BS$_m$
$B_m, R_m, \mathbf{R}_m(l_t^m) \subset \mathbf{R}, \forall m \in \mathbf{M}$	Bandwidth, service radius and service range of BS$_m$, respectively
$v_{\max}^m, T_{\max}^m, E_a^m, \forall m \in \mathbf{M}$	Maximal speed, maximal endurance and energy consumption per stage of active aerial BS$_m$, respectively
$\mathbf{z}_t = \{z_t^m \in \mathbf{Z}^m \mid m \in \mathbf{M}\} \in \mathbf{Z}$	BSs' activity vector at stage t, where z_t^m is the activity of BS$_m$, where $\mathbf{Z}^m$ is the set of possible values of z_t^m
$\boldsymbol{\theta}_t = \{\theta_t^n = (\theta_t^{n(I)}, \theta_t^{n(P)}, \theta_t^{n(D)}, \theta_t^{n(O)}) \in \boldsymbol{\Theta}^n \mid n \in \mathbf{N}\} \in \boldsymbol{\Theta}$	Mining tasks recorded by peers at stage t, where θ_t^n is the task of peer P$_n$, $\theta_t^{n(I)}$ is the input task size; $\theta_t^{n(P)}$ is the processing task size; $\theta_t^{n(D)}$ is the task completion deadline; $\theta_t^{n(O)}$ is the task output size in bits, $\boldsymbol{\Theta}^n = \{(0,0,0,0), \dots, (\theta_{max}^{n(I)}, \theta_{max}^{n(P)}, \theta_{max}^{n(D)}, \theta_{max}^{n(O)})\}$ is the set of possible task parameters; $\theta_{max}^{n(I)}$, $\theta_{max}^{n(P)}$, $\theta_{max}^{n(D)}$ and $\theta_{max}^{n(O)}$ are maximal input size, processing size, completion deadline and output size, respectively
$\tilde{\theta}^{(I)}, \tilde{\theta}^{(P)}$	Average input and processing sizes, respectively, of the mining tasks of the peers
$\mathbf{a}_t = (\mathbf{x}_t, \mathbf{y}_t) = \{a_t^n = (y_t^n, x_t^n) \in \mathbf{A}^n \mid n \in \mathbf{N}\} \in \mathbf{A}$	Actions or offloading decisions of all peers at stage t, where a_t^n is the action or offloading decision of peer P$_n$ that includes the processing decision y_t^n and forwarding decision x_t^n, where $\mathbf{A}^n$ is the set of possible actions of the peer
$p^n, R_U^{n,m}, \beta_U^{n,m}, G^{n,m}, \forall n \in \mathbf{N}, m \in \mathbf{M}$	Transmit power of peer P$_n$, data rate of UL cellular channel between peer P$_n$ and BS, UL bandwidth of a cellular link between peer P$_n$ and BS$_m$, and gain of a cellular link between peer P$_n$ and BS$_m$, respectively
$\tilde{p}^i, b_U^{m,i}, \tilde{G}^{i,m}, \sigma^2, \forall m, i \in \mathbf{M}$	Average transmit power of peers transmitting to BS$_i$, UL spectrum overlap indicator between BS$_m$ and BS$_i$, average channel gain of cellular links between BS$_i$ and the peers transmitting to BS$_i$, and variance of zero-mean AWGN power, respectively.
$\mathbf{Q}_t^n = \{Q_t^n\} \cup \{Q_t^m\}_{m \in \mathbf{M}} \cup \{Q_t^{m,0}\}_{m \in \mathbf{M}_\mathbf{T}} \in \boldsymbol{\Omega}^n, \forall n \in \mathbf{N}$	Queue backlogs observable by peer P$_n$, where Q_t^n is the backlog of the processor queue of peer P$_n$, Q_t^m is the backlog of the MEC server queue at BS$_m$, $Q_t^{m,0}$ is the backlog of the blockchain server queue connected to BS$_m$, $\boldsymbol{\Omega}^n$ is the set of possible values of $\mathbf{Q}_t^n$
$D^n, D^{n(I)}, D^{n(P)}, \forall n \in \mathbf{N}$	Transaction delay for the task recorded by peer P$_n$ at stage t which includes the transmission delay $D^{n(I)}$ and processing delay $D^{n(P)}$
$E^n, E^{n(I)}, E^{n(P)}, \forall n \in \mathbf{N}$	Total energy spent on transacting the task recorded by peer P$_n$ at stage t which includes the total energy $E^{n(I)}$ on transmitting the task and total energy $E^{n(P)}$ on processing the task
$E^{n,0}, E^{n,0(I)}, E^{n,0(P)}, \forall n \in \mathbf{N}$	Energy spent by peer P$_n$ or the blockchain server on transacting its task recorded at stage t which includes the energy $E^{n,o(I)}$ spent by the peer or the blockchain server on transmitting the task and energy $E^{n(P)}$ spent by the peer or the blockchain server on processing the task
$E^{n,m}, E^{n,m(I)}, E^{n,m(P)}, \forall n \in \mathbf{N}, m \in \mathbf{M}$	Total energy spent by BS$_m$ on transacting the task comprising the energy $E^{n,m(I)}$ on forwarding the task to the blockchain server and energy $E^{n,m(P)}$ on processing the task at the BS's MEC server.
$B_t^n(\mathbf{a}^{-n}) = \mathrm{Pr}_n\{\mathbf{a}_t^{-n} = \mathbf{a}^{-n}\} = \prod_{i \in \mathbf{N} \setminus \{n\}} B_t^n(a^i) = \prod_{i \in \mathbf{N} \setminus \{n\}} \mathrm{Pr}_n\{a_t^i = a^i\}, \forall n \in \mathbf{N}$	Belief of peer P$_n$ about actions $\mathbf{a}_t^{-n} = \{a_t^i\}_{i \in \mathbf{N} \setminus \{n\}} = \mathbf{A}^{-n}$ of other peers at stage t, i.e., probability $\mathrm{Pr}_n\{\mathbf{a}_t^{-n} = \mathbf{a}^{-n}\}$ that peer P$_n$ assigns to other peers having the action profile $\mathbf{a}_t^{-n} = \mathbf{a}^{-n}$
$s_t^n = (\theta_t^n, l_t^n, \mathbf{c}_t, \mathbf{l}_t, \mathbf{z}_t, \mathbf{Q}_t^n) \in \mathbf{S}^n, \forall n \in \mathbf{N}$	Stochastic state of peer P$_n$ that combines the parameters observable by the peer at the beginning of stage t, where $\mathbf{S}^n = \boldsymbol{\Theta}^n \times \mathbf{L}^n \times_{m \in \mathbf{M}} (\mathbf{C}^m, \mathbf{L}^m, \mathbf{Z}^m) \times \boldsymbol{\Omega}^n$ is the state space
$\xi^n, C^n, \zeta^n, \forall n \in \mathbf{N}$	Transaction fee for the task of peer P$_n$, total stage costs and cost per energy unit paid by peer P$_n$, respectively
$u^n, U^n, V^n, \forall n \in \mathbf{N}$	Instantaneous stage payoff, expected stage payoff and long-term payoff of peer P$_n$, respectively
$\boldsymbol{a}_t = \{a_t^m = (c_t^m, l_t^m, z_t^m) \in \boldsymbol{\mathcal{A}}^m \mid m \in \mathbf{M}\} \in \boldsymbol{\mathcal{A}}$, $\boldsymbol{a}_t^{-m} = \{a_t^i\}_{i \in \mathbf{M} \setminus \{m\}} = \boldsymbol{\mathcal{A}}^{-m}, \forall m \in \mathbf{M}$	Action profile of BSs and action profile of BSs different from BS$_m$, respectively, where $a_t^m \in \boldsymbol{\mathcal{A}}^m$ is the action of BS$_m$ represented by its selected stage parameters, $\boldsymbol{\mathcal{A}}^m = \mathbf{C}^m \times \mathbf{L}^m \times \mathbf{Z}^m$ is the action space of BS$_m$
$\zeta^m, u^m, V^m, \forall m \in \mathbf{M}$	Cost per energy unit to BS$_m$, instantaneous stage payoff of BS$_m$ and long-term payoff of BS$_m$, respectively
$\mathcal{s}_t = (\mathbf{a}_t, \boldsymbol{\theta}_t, \mathbf{l}_t^\mathrm{p}) \in \boldsymbol{\mathcal{S}}$	Stochastic state of BSs that combines the parameters observable by BSs at the end of stage t, where $\boldsymbol{\mathcal{S}} = \mathbf{A} \times \boldsymbol{\Theta} \times \mathbf{L}^\mathrm{p}$ is the state space
$\alpha^n, \alpha^m, \forall n \in \mathbf{N}, m \in \mathbf{M}$	Pure strategies of peer P$_n$ and BS$_m$, respectively
γ, λ, η	Discount rate, dropout rate, and learning rate of stochastic gradient descent, respectively
$(\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1,\dots,F}$	Dataset comprising 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 $\mathbf{y}_t \in \mathbf{Y} = \{1, \dots, D_O\}$ is the observed output from the set $\mathbf{Y}$
$\mathrm{Pr}\{\mathbf{y}_t \mid f^\mathbf{W}(\mathbf{x}_t)\}$	Likelihood of data point $(\mathbf{x}_t, \mathbf{y}_t)$, where $f^\mathbf{W}(\mathbf{x}_t) = \hat{\mathbf{y}}_t = (\hat{y}_t^1, \dots, \hat{y}_t^{D_O}) \in \mathbb{R}^{1 \times D_O}$ is the random row output vector of the BNN with random weights $\mathbf{W}$ given the input $\mathbf{x}_t$
$\mathrm{Softmax}(i, f^\mathbf{W}(\mathbf{x}_t)), \forall i \in \mathbf{Y}$	Element-wise softmax function: $\mathrm{Softmax}(i, f^\mathbf{W}(\mathbf{x}_t)) = e^{\hat{y}_t^i} / (\sum_{j=1}^{D_O} e^{\hat{y}_t^j})$
q^θ, q	Approximating distribution with parameter θ and parameter-free approximating distribution, respectively
D_KL, ELBO, $\mathcal{L}$	KL divergence, ELBO and approximation loss, respectively
$\widehat{\mathbf{W}} = g(\theta, \epsilon)$	BNN weights generated according to approximating distribution q^θ with parameter θ, where $\epsilon > 0$ is an infinitesimal number; $g(\cdot\,,\cdot)$ is a deterministic differentiable bivariate transformation.
$\hat{\mathbf{y}}_t = \varphi(\mathbf{x}_t \mathbf{W}_{(h)} + \mathbf{b}) \mathbf{W}_{(v)}^\mathrm{T} = f^\mathbf{W}(\mathbf{x}_t) \in \mathbb{R}^{1 \times D_O}$	Output of the feed-forward BNN model of peer P$_n$ with weights $\mathbf{W} = (\mathbf{W}_{(h)}, \mathbf{W}_{(v)}, \mathbf{b}) \in \mathbb{R}^{(D_I + D_O + 1) \times K}$ given input $\mathbf{x}_t \in \mathbb{R}^{1 \times D_I}$, where $K \geq 1$ is the number of neurons in the BNN model; $\mathbf{W}_{(h)} = (w_{(h)}^{i,j}) \in \mathbb{R}^{D_I \times K}$ and $\mathbf{W}_{(v)} = (w_{(v)}^{i,j}) \in \mathbb{R}^{D_O \times K}$ are the weights of neurons in the hidden and visible layers, respectively; $\mathbf{b} = (b^j) \in \mathbb{R}^{1 \times K}$ are the biases; $\varphi(\cdot)$ is a logistic sigmoid function
$\hat{V}_{t+1}^m(a^m \mid \mathcal{s}, \boldsymbol{\omega}) = \varphi\left((a^m, \mathcal{s}) \boldsymbol{\omega}_{(h)} + \mathbf{b}\right) \boldsymbol{\omega}_{(v)}^\mathrm{T} = f^{\boldsymbol{\omega}}(a^m, \mathcal{s}) \in \mathbb{R}, \forall m \in \mathbf{M}$	Output of the NN model of BS$_m$ with the weights $\boldsymbol{\omega} = (\boldsymbol{\omega}_{(h)}, \boldsymbol{\omega}_{(v)}, \mathbf{b}) \in \mathbb{R}^{(D_I + 2) \times K}$ given input $(a^m, \mathcal{s}) \in \mathbb{R}^{1 \times D_I}$, where $K \geq 1$ is the number of neurons in the NN model; $\boldsymbol{\omega}_{(h)} = (\omega_{(h)}^{i,j}) \in \mathbb{R}^{D_I \times K}$ and $\boldsymbol{\omega}_{(v)} = (\omega_{(v)}^{j}) \in \mathbb{R}^{1 \times K}$ are the weights of neurons in the hidden and visible layers, respectively; $\mathbf{b} = (b^j) \in \mathbb{R}^{1 \times K}$ are the biases; $\varphi(\cdot)$ is a logistic sigmoid function

without any prior training. We prove that the DQL algorithm converges to a stable state that maximizes the leader's long-term payoff.

The remainder of this article is organized as follows. In Section II, we review the related works on resource management and pricing for the IoT/mobile blockchains. In

Section III, we present a general architecture of the IoT system with BaaS and MEC. In Section IV, we derive the analytical model and delay/energy expressions for IoT-blockchain tasks. In Section V, we define a Stackelberg model of the BaaS-MEC support. In Section VI, we formulate a stochastic game, devise the BRL algorithm for the followers, and propose the BDL algorithm in Section VII. In Section VIII, we develop the game and present the RL and DQL algorithms for the leaders. In Section IX, we conduct a numerical performance evaluation of the framework.

II. Related Work

Most works on blockchain applications in the mobile and IoT systems (e.g., [5], [7], [8] and [18]–[21]) study the implementation and design of the blockchain-based protocols for decentralized and secure data communication, unrelated to the framework of this article. The only works which consider resource management and pricing for mobile/IoT blockchains have been conducted in [22]–[27] where the blockchain peer can offload its blockchain tasks to one of the BSs in the MEC network. The focus here is on the mining under the proof-of-work (PoW) consensus [4], [5] which results in the competition among peers to win a mining reward. In [22] and [23], the interplay between the peers and the MEC service provider (SP) has been represented as a two-stage Stackelberg game. At stage one, a leader/SP assigns the price per unit hash rate with the objective to maximize its one-stage profit. At stage two, given the unit price, the followers/peers select hash rates aiming to maximize their one-stage payoffs. The auction-theoretic resource allocation approaches for blockchains with MEC have been analyzed in [24] and [25]. The objective of the auction in [24] is to maximize a social welfare while ensuring its incentive compatibility, i.e., trustfulness for peers. In [25], the performance of auction designed to maximize the revenues of MEC SPs has been improved by adopting deep learning. In [26] and [27], resource allocation, offloading, and block size adaptation for blockchains in the MEC have been represented as a binary programming problem with the objective to maximize the average reward of peers. To reduce the problem size, we adopt the alternating direction method of multipliers (ADMMs) that yields a distributed algorithm.

To summarize, the works in [22]–[27] study the possibility of enhancing the computing abilities of mobile devices with MEC services. Yet, they still have several limitations. First, the main assumption of these studies is that information about peers' actions, e.g., offloading or hash rate decisions, is public, i.e., known to other peers. However, in the blockchain networks, the peers take actions simultaneously and independently [28]. In addition, in mobile settings, communications among peers lead to additional delay and energy overheads. Second, the game and auction models presented in these studies ignore dynamics and randomness of the wireless environment, e.g., locations of peers and channel quality. As such, the games/auctions are played in a static and deterministic environment and result in short-term (one-stage) solutions. Third, the works in [22]–[27] are designated specially for a PoW consensus and cannot be used with other consensus methods. Although PoW is a popular protocol, it is compute-intensive and has a poor scalability, since each mining task must be computed by all blockchain nodes [29]. As a result, PoW is not suitable for scenarios with large numbers of nodes such as IoT systems.

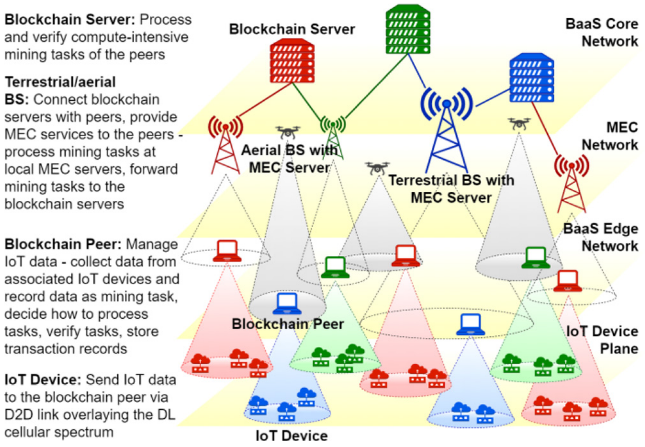
Blockchain Server: Process and verify compute-intensive mining tasks of the peers
Terrestrial/aerial BS: Connect blockchain servers with peers, provide MEC services to the peers - process mining tasks at local MEC servers, forward mining tasks to the blockchain servers
Blockchain Peer: Manage IoT data - collect data from associated IoT devices and record data as mining task, decide how to process tasks, verify tasks, store transaction records
IoT Device: Send IoT data to the blockchain peer via D2D link overlaying the DL cellular spectrum
Blockchain Server
BaaS Core Network
Aerial BS with MEC Server
Terrestrial BS with MEC Server
MEC Network
BaaS Edge Network
Blockchain Peer
IoT Device Plane
IoT Device

Fig. 1. General architecture of the IoT system with BaaS-MEC support.

Similar to the above works, we analyze the pricing and resource management in blockchains supported by MEC. However, our framework is based on more realistic and applicable to the IoT considerations are as follows.

1) The information about actions of blockchain peers is private, i.e., unknown to other peers. As a result, the peers must form their beliefs about unobservable actions and estimate their best response strategies by adopting BRL or BDL.
2) The proposed Stackelberg model is stochastic. That is, it is played in a dynamic and random environment and produces long-term (multistage) solutions.
3) The framework is not linked to a specific consensus scheme, i.e., it can be applied with any arbitrary consensus protocol.

III. IoT System With BaaS-MEC Support

A. Architecture and System Model

The main notations used in this article are listed in Table I.

First, we introduce a general architecture of the IoT system with BaaS-MEC support (as shown in Fig. 1) that categorizes all system participants into four categories as follows.

1) *IoT Devices:* Fixed and mobile IoT devices, such as sensors, that are exploited only for basic functions—sense, collect, and send data to be recorded in the IoT blockchains to the peers operated by the BaaS provider.
2) *Peers:* Semistatic blockchain nodes located at the network edges, each of which can be represented by the desktop computer or mobile terminal (e.g., Laptop, Smartphone, or Tablet). The main function of peers is to manage the IoT data collected from the IoT devices in their transmission ranges, i.e., record data as mining tasks, process and verify tasks, and store verified tasks in the form of transaction records.

3) *MEC Network:* It can be formed by terrestrial BSs with links to the blockchain servers and aerial BSs, i.e., UAVs. The function of the MEC network is to connect peers at the network edges with blockchain servers at the network core. In addition, the MEC servers installed at BSs can be used to process some mining tasks.
4) *Blockchain Servers:* Resource-rich core servers managed by the BaaS provider. Unlike peers, servers are only accessed through the terrestrial BSs.

We now present a general model of the IoT system with BaaS-MEC support where IoT devices are served by the set $\mathbf{N} = \{1, \ldots, N\}$ of peers labeled as $P_1, \ldots, P_N$ and a blockchain server. The main function of peers is to manage the IoT data, e.g., sensing records, collected from the IoT devices in their transmission ranges. In addition, each peer P_n, $n \in \mathbf{N}$, has a local processor of the computing power ρ^n in CPU cycles per second to process mining tasks. We assume that the peers do not communicate with each other, because such information exchange introduces heavy signaling over a wireless medium that increases delays and transmission costs for the peers. The MEC network is formed by $M = M_T + M_A$ BSs, including M_T terrestrial BSs connected to the blockchain server via fiber backhaul links, and M_A aerial BSs, i.e., the UAVs. For notation consistency, terrestrial BSs are labeled as $\text{BS}_{N+1}, \ldots, \text{BS}_{N+MT}$; and aerial BSs are labeled as $\text{BS}_{N+MT+1}, \ldots, BS_{N+MT+MA}$. Thus, the set of terrestrial BSs is defined by $\mathbf{M}_T = \{N+1, \ldots, N+M_T\}$, the set of aerial BSs is $\mathbf{M}_A = \{N+M_T+1, \ldots, N+M_T+M_A\}$, the total set of BSs is $\mathbf{M} = \mathbf{M}_T \cup \mathbf{M}_A = \{N+1, \ldots, N+M\}$.

Since BSs can be privately owned or controlled by different operators, in general, there is no communication among the BSs. Each BS_m, $m \in \mathbf{M}$, operates on an orthogonal full-duplex (FD) band denoted by B_m which may overlap with the spectra of other BSs. The MEC server of BS_m represents a processor of a computing power ρ^m in CPU cycles per second to perform some mining tasks. On the contrary, a resource-rich blockchain server includes several processors to run multiple blockchain operations in parallel and confirm IoT-blockchain transactions. We consider that the blockchain server has M_T processors, each of which has the computing power of ρ^0 in CPU cycles per second and is connected to a terrestrial BS to support parallel processing of the tasks forwarded by BSs.

B. Mining Process and System Parameters

The process of blockchain mining consists of the number of stages, denoted as $t = 0, \ldots, T$. The peers and BSs remain in the system for indefinitely long time, i.e., $T \to \propto$. During each stage, the network parameters (e.g., channel quality, locations of peers, IoT devices, and UAVs) remain fixed, although the parameters can change, as we move to the next stage. At the beginning of stage t, each BS_m, $m \in \mathbf{M}$, assigns the stage cost $c_t^m = (c_t^{m(I)}, c_t^{m(P)}) \in \mathbf{C}^m$ for its MEC services per mining task and announces the cost to peers in its service range. The cost c_t^m comprises: 1) forwarding cost $c_t^{m(I)}$, i.e., cost of forwarding the task to the blockchain server via BS_m and 2) processing cost $c_t^{m(P)}$, i.e., cost of processing the task at the MEC server of BS_m. Forwarding services are provided only by the terrestrial BSs that have links to the blockchain server. Hence, $c_t^{m(I)} = 0$, for an aerial BS_m, $m \in \mathbf{M}_A$. All costs are counted in terms of units of a certain currency (e.g., bitcoin). Hence, the set of possible costs, $\mathbf{C}^m = \{(0, 0), (0, 1), \ldots, (C_{\max}^{m(I)}, C_{\max}^{m(P)})\}$, where $C_{\max}^{m(I)}$ and $C_{\max}^{m(P)}$ are the maximal processing and forwarding costs, is discrete and finite. Meanwhile, each peer P_n, $n \in \mathbf{N}$, collects IoT data and records these data as a mining task. In order to be recorded in the IoT blockchain, after the task is processed, its output must be confirmed by other peers. Once all the task's outputs are received and confirmed, the current stage t of the mining process concludes and the next stage $t+1$ begins.

Let $l_t^n \in \mathbf{L}^n \subset \mathbf{R}$, $n \in \mathbf{N}$, denote the 2-D location of peer P_n at stage t, where $\mathbf{L}^n$ is the set of possible 2-D locations of peer P_n; and $\mathbf{R}$ is the service area of the network. Let $l_t^m = (h_t^m, \ell_t^m) \in \mathbf{L}^m$, $m \in \mathbf{M}$, be the 3-D location of BS_m at stage t, i.e., the altitude h_t^m and the projected (to the ground) 2-D location ℓ_t^m of BS_m, where $\mathbf{L}^m$ is the set of possible 3-D locations of BS_m. Note that $h_t^m = 0$ and $l_t^m = l_0^m$, for the terrestrial BS_m, $m \in \mathbf{M}_T$. To be compatible with the framework in this article, all distances in the network are discretized, i.e., the distance axis is partitioned into equal intervals of a small length Δl (in meters). Hence, the sets $\mathbf{L}^n$, $\mathbf{L}^m$, and $\mathbf{R}$ are finite. Given that the service range $\mathbf{R}_m(l_t^m) \subset \mathbf{R}$ of BS_m is bounded by a circle of a radius $R_m \in \mathbb{R}$, we obtain the following logical relation:

$$l_t^n \in \mathbf{R}_m\left(l_t^m\right) \Longleftrightarrow \sqrt{\left|h_t^m\right|^2 + \left\|l_t^n - \ell_t^m\right\|^2} \leq R_m \quad \forall n \in \mathbf{N} \tag{1a}$$

for all $m \in \mathbf{M}$, at any stage t, where the 3-D location l_t^m of BS_m is such that

$$\sqrt{\left|h_t^m - h_{t-1}^m\right|^2 + \left\|\ell_t^m - \ell_{t-1}^m\right\|^2} \leq v_{\max}^m \Delta t$$

where $v_{\max}^m$ is a maximal speed of BS_m in meters per second ($v_{\max}^m = 0$, for a terrestrial BS_m, $m \in \mathbf{M}_T$), Δt is the stage duration (in seconds). Then, at any stage t, given the past 3-D location $l_{t-1}^m = (h_{t-1}^m, \ell_{t-1}^m)$ of BS_m, the set $\mathbf{L}^m$ is defined by

$$\mathbf{L}^m = \left\{ l_t^m = \left(h_t^m, \ell_t^m\right) \middle| \begin{array}{c} \left|h_t^m - h_{t-1}^m\right|^2 + \left\|\ell_t^m - \ell_{t-1}^m\right\|^2 \\ \leq \left(v_{\max}^m \Delta t\right)^2 \end{array} \right\} \tag{1b}$$

for all $m \in \mathbf{M}_A$, and $\mathbf{L}^m = \{l_t^m = (h_t^m, \ell_t^m) = (0, l_0^m)\}$, for all $m \in \mathbf{M}_T$.

Unlike terrestrial BSs, aerial BSs or UAVs have limited fly duration and endurance, e.g., the average endurance of rotary-wind UAVs is 30 min. Hence, no UAV can be constantly active and must be recalled for battery recharging or swapping [15]–[17]. Let $\mathbf{z}_t = \{z_t^m \in \{0, 1\}, z_t^i = 1 | m \in \mathbf{M}_A, i \in \mathbf{M}_T\}$ be the BSs' activity vector, such that $z_t^m = 1$, if BS_m is active at stage t. The activity of aerial BSs can be controlled to achieve various objectives. For example, to prolong the endurance and reduce the energy consumption, the UAV can be deactivated when there are no users in its service ranges. As such, at any stage t, we have $\Delta t \sum_{\tau=t-T_{\max}^m/\Delta t}^{t} z_\tau^m \leq T_{\max}^m$, for $m \in \mathbf{M}_A$, where $T_{\max}^m$ is the maximal endurance of aerial BS_m (in seconds). In words, BS_m can remain active in the air as long as its "ON period"

$\Delta t \sum_{\tau=t-T^m_{\max}/\Delta t}^{t} z^m_\tau$ does not exceed $T^m_{\max}$. Thus, at any stage t, given the activity history $z^m_{t-1}, \ldots, z^m_{t-T^m_{\max}/\Delta t}$, the set $\mathbf{Z}^m$ of possible values of z^m_t is defined by

$$\mathbf{Z}^m = \left\{ z^m_t \in \{0, 1\} \middle| z^m_t \le \frac{T^m_{\max}}{\Delta t} - \sum_{\tau=t-T^m_{\max}/\Delta t}^{t-1} z^m_\tau \right\} \quad \forall m \in \mathbf{M}_A \tag{1c}$$

and $\mathbf{Z}^m = \{z^m_t = 1\}$, for $m \in \mathbf{M}_T$.

IV. Analytical Model of the System

A. Blockchain Mining Model

To be recorded as a confirmed transaction in the blockchain, IoT data must be: 1) collected by the peers from IoT devices in their transmission ranges and 2) processed and verified. Note that as the number of IoT devices is commonly very large, it is rather inefficient (e.g., in terms of spectrum costs/utilization) to allocate them with the separate cellular spectrum [1]. Instead, data transmissions between the IoT devices and the peers can be realized over device-to-device (D2D) links overlaying the downlink (DL) spectrum of the MEC network. The DL cellular spectrum is favored because it eliminates interference between cellular and D2D transmissions, since the uplink (UL) channel load in the MEC network is usually much higher than the DL load [14], [15]. As such, at any stage t, data collected by peer P_n, $n \in \mathbf{N}$, from IoT devices represents a mining task defined by the tuple $\theta^n_t = (\theta^{n(I)}_t, \theta^{n(P)}_t, \theta^{n(D)}_t, \theta^{n(O)}_t) \in \mathbf{\Theta}^n$, where $\theta^{n(I)}_t$ is the input size in bits; $\theta^{n(P)}_t$ is the processing size, i.e., the number of CPU cycles required to finish the task; $\theta^{n(D)}_t$ is the completion deadline in seconds; and $\theta^{n(O)}_t$ is the output size in bits. The set $\mathbf{\Theta}^n = \{(0, 0, 0, 0), (0, 0, 0, 1), \ldots, (\theta^{n(I)}_{\max}, \theta^{n(P)}_{\max}, \theta^{n(D)}_{\max}, \theta^{n(O)}_{\max})\}$ contains all possible parameters of the task θ^n_t, where $\theta^n_t = (0, 0, 0, 0)$, if no task is recorded at stage t; $\theta^{n(I)}_{\max}$, $\theta^{n(P)}_{\max}$, $\theta^{n(D)}_{\max}$, and $\theta^{n(O)}_{\max}$ are the maximal input size, processing size, completion deadline, and output size, respectively. To estimate the parameters $\theta^{n(I)}_t$, $\theta^{n(P)}_t$, $\theta^{n(D)}_t$, $\theta^{n(O)}_t$, one of the analysis methods, e.g., call graph analysis [30], can be applied.

Upon recording its task θ^n_t, peer P_n has the following options.

1) *Option 1:* Offload the task processing to one of the active BSs in the transmission range of the peer.
2) *Option 2:* Offload the task processing to a blockchain server via a terrestrial BS in the transmission range of the peer.
3) *Option 3:* Process the task locally.

Once the task θ^n_t is completed, its output $\theta^{n(O)}_t$ that represents the unconfirmed transaction must be verified by other peers, i.e., the peers different from peer P_n. The transaction is verified only if the total transaction delay, i.e., the sum of processing and transmission delay, for the task θ^n_t does not exceed its deadline $\theta^{n(D)}_t$. If confirmed, the output $\theta^{n(O)}_t$ is appended to the blockchain and stored by the peer for future references, e.g., data analytics or system control. Otherwise, the output $\theta^{n(O)}_t$ is discarded, which is called orphaning [23]–[25].

Now, consider peer P_n that has recorded the mining task θ^n_t at stage t. Let $\mathbf{M}_n = \{m \in \mathbf{M} | z^m_t = 1, l^n_t \in \mathbf{R}_m(l^m_t)\}$ be the set of active BSs within the transmission range of peer P_n. The offloading decision $a^n_t = (y^n_t, x^n_t) \in \mathbf{A}^n$ for the task θ^n_t of peer P_n at stage t consists of the processing decision $y^n_t \in \{0, 1\}$ and forwarding decision $x^n_t \in \{0\} \cup \mathbf{M}_n$. The decisions y^n_t and x^n_t are such that: 1) $y^n_t = 0$, if the task is processed locally (at the MEC server of a respective BS or at the local processor of peer P_n); 2) $y^m_t = 1$, if the task is computed at the blockchain server; 3) $x^n_t = 0$, if the task θ^n_t is not forwarded; and 4) $x^n_t = m \in \mathbf{M}_n$, if the task θ^n_t is forwarded to BS$_m$. Note that the task can be offloaded to the server only if there is at least one terrestrial BS in the set $\mathbf{M}_n$, i.e., $\mathbf{M}_n \cap \mathbf{M}_T \neq \emptyset$. Thus, a set $\mathbf{A}^n$ of possible offloading decisions (y^n_t, x^n_t) of peer P_n is given by

$$\mathbf{A}^n = \left\{ a^n_t = \left(y^n_t, x^n_t\right) \middle| \begin{array}{l} y^n_t \in \{0, 1\}, x^n_t \in \{0\} \cup \mathbf{M}_n \\ y^n_t = 0 \Longrightarrow x^n_t \in \{0\} \cup \mathbf{M}_n \\ y^n_t = 1 \Longrightarrow x^n_t \in \mathbf{M}_n \cap \mathbf{M}_T \end{array} \right\} \quad \forall n \in \mathbf{N}. \tag{2}$$

Then, in Option 1, $y^n_t = 0$ and $x^n_t = m \in \mathbf{M}_n$. That is, peer P_n offloads the task input $\theta^{n(I)}_t$ to BS$_m$ through a UL cellular channel of some data rate $R^{n,m}_U(\mathbf{x}_t)$ that can be estimated, as in

$$R^{n,m}_U(\mathbf{x}_t) = \beta^{n,m}_U(\mathbf{x}_t) \log_2\left(1 + \frac{p^n G^{n,m}}{\sum_{i \in \mathbf{M}\setminus\{m\}} b^{m,i}_U \tilde{p}^i \tilde{G}^{i,m} + \sigma^2}\right) \tag{3a}$$

where $\beta^{n,m}_U(\mathbf{x}_t)$ is the UL bandwidth of a cellular link between peer P_n and BS$_m$ assumed to be distributed equally among all peers transmitting to BS$_m$ (for the fairness purpose), i.e.,

$$\beta^{n,m}_U(\mathbf{x}_t) = \frac{B_m}{\sum_{j \in \mathbf{N}} \mathbf{1}_{x^j_t = m}} \tag{3b}$$

p^n is a transmit power of peer P_n; $G^{n,m}$ is the gain of a cellular link between peer P_n and BS$_m$; $b^{m,i}_U \in \{0,\ 1\}$ is a UL spectrum overlap indicator, such that $b^{m,i}_U = 1$, if a UL spectrum of BS$_m$ overlaps with a UL spectrum of BS$_i$; $\tilde{p}^i$ is an average transmit power of peers transmitting to BS$_i$; $\tilde{G}^{i,m}$ is an average channel gain of cellular links between BS$_i$ and the peers transmitting to BS$_i$; σ^2 is a variance of a zero-mean additive white Gaussian noise (AWGN) power. In Option 2, $y^n_t = 1$ and $x^n_t = m \in \mathbf{M}_n$ $\cap \mathbf{M}_T$. In this case, a terrestrial BS$_m$ receives the task θ^n_t from peer P_n over the UL cellular channel with data rate $R^{n,m}_U(\mathbf{x}_t)$ and forwards it to the blockchain server via a fiber backhaul link of a capacity $R^{m,0}_f$. In Option 3, $y^n_t = 0$ and $x^n_t = 0$, i.e., the peer processes the task itself.

B. Transaction Delay and Energy Consumption

Since each peer P_n and each BS$_m$ are equipped with the local processors, the processing of tasks by the peer and the BS can be represented by the virtual processor queues denoted by Q^n and Q^m, respectively. On the contrary, the blockchain server has M_T parallel processors, each of which is connected to the terrestrial BS. Therefore, the processing of tasks at the server can be defined by M_T parallel queues $\{Q^{m,0}\}_{m \in \mathbf{M}_T}$, where $Q^{m,0}$ is the queue connected to a terrestrial BS$_m$. Let Q^n_t, Q^m_t, and $Q^{m,0}_t$ be the backlogs of queues Q^n, Q^m, and $Q^{m,0}$ at stage t. Propositions 1 and 2 below establish the relationships between the offloading decisions $\mathbf{a}_t = \{a^n_t\}_{n \in \mathbf{N}} = \{(y^n_t, x^n_t)\}_{n \in \mathbf{N}}$

made by all the peers and the transaction delay D^n and energy E^n spent on transacting, i.e., completing, the task θ_t^n of any peer P_n given the parameters $(\theta_t^n, l_t^n, \mathbf{l}_t, \mathbf{z}_t, \mathbf{Q}_t^n)$, for $\mathbf{l}_t = \{l_t^m\}_{m\in\mathbf{M}}$ and $\mathbf{Q}_t^n = \{Q_t^n\} \cup \{Q_t^m\}_{m\in\mathbf{M}} \cup \{Q_t^{m,0}\}_{m\in\mathbf{M}_T}$.

Proposition 1: The transaction delay D^n for the mining task θ_t^n recorded by peer P_n at stage t is the sum of transmission delay $D^{n(I)}$ and processing delay $D^{n(P)}$. That is

$$\begin{aligned} D^n\left(\mathbf{a}_t \middle| \theta_t^n, l_t^n, \mathbf{l}_t, \mathbf{z}_t, \mathbf{Q}_t^n\right) &= D^{n(I)}\left(\mathbf{a}_t|\theta_t^n, l_t^n, \mathbf{l}_t, \mathbf{z}_t\right) \\ &\quad + D^{n(P)}\left(\mathbf{a}_t|\theta_t^n, l_t^n, \mathbf{l}_t, \mathbf{z}_t, \mathbf{Q}_t^n\right) \end{aligned} \tag{4a}$$

where

$$\begin{aligned} &D^{n(I)}\left(\mathbf{a}_t|\theta_t^n, l_t^n, \mathbf{l}_t, \mathbf{z}_t\right) \\ &= \sum_{m\in\mathbf{M}} z_t^m \mathbf{1}_{l_t^n\in\mathbf{R}_m(l_t^m)} \left(\mathbf{1}_{x_t^n=m} \frac{\theta_t^{n(I)}}{R_U^{n,m}(\mathbf{x}_t)} + \mathbf{1}_{m\in\mathbf{M}_T} \frac{\widetilde{\theta}^{(I)}}{R_f^{m,0}} \sum_{j\in\mathbf{N}} y_t^j \mathbf{1}_{x_t^j=m}\right) \end{aligned} \tag{4b}$$

$$\begin{aligned} &D^{n(P)}\left(\mathbf{a}_t|\theta_t^n, l_t^n, \mathbf{l}_t, \mathbf{z}_t, \mathbf{Q}_t^n\right) \\ &= \frac{1}{\Delta t}\left(\frac{\left(1-y_t^n\right)\left(1-x_t^n\right)}{\rho^n}\left(Q_t^n + \theta_t^{n(P)}\right) + \sum_{m\in\mathbf{M}} z_t^m \mathbf{1}_{l_t^n\in\mathbf{R}_m(l_t^m)} \mathbf{1}_{x_t^n=m}\right. \\ &\qquad \times \left(\frac{y_t^n}{\rho^0}\left(Q_t^{m,0} + \theta_t^{n(P)} + \frac{\widetilde{\theta}^{(P)}}{2} \sum_{i\in\mathbf{N}\setminus\{n\}} \mathbf{1}_{x_t^i=m} y_t^i\right) + \frac{1-y_t^n}{\rho^m}\right. \\ &\qquad \left.\left.\times \left(Q_t^m + \theta_t^{n(P)} + \frac{\widetilde{\theta}^{(P)}}{2} \sum_{i\in\mathbf{N}\setminus\{n\}} \mathbf{1}_{x_t^i=m}\left(1-y_t^i\right)\right)\right)\right) \end{aligned} \tag{4c}$$

where $\widetilde{\theta}^{(I)}$ and $\widetilde{\theta}^{(P)}$ are the average input and processing sizes, respectively, of the peers' mining tasks.

Proposition 2: The energy E^n on transacting the mining task θ_t^n recorded by peer P_n at stage t is the sum of energy $E^{n(I)}$ on transmitting and energy $E^{n(P)}$ on processing the task. That is

$$\begin{aligned} E^n\left(\mathbf{a}_t|\theta_t^n, l_t^n, \mathbf{l}_t, \mathbf{z}_t\right) &= E^{n(I)}\left(\mathbf{a}_t|\theta_t^n, l_t^n, \mathbf{l}_t, \mathbf{z}_t\right) \\ &\quad + E^{n(P)}\left(a_t^n|\theta_t^n, l_t^n, \mathbf{l}_t, \mathbf{z}_t\right) \end{aligned} \tag{5a}$$

where

$$\begin{aligned} E^{n(I)}\left(\mathbf{a}_t|\theta_t^n, l_t^n, \mathbf{l}_t, \mathbf{z}_t\right) &= E^{n,0(I)}\left(a_t^n|\theta_t^n, l_t^n, \mathbf{l}_t, \mathbf{z}_t\right) \\ &\quad + \sum_{m\in\mathbf{M}_T} E^{n,m(I)}\left(\mathbf{a}_t|l_t^n, l_t^m\right) \end{aligned} \tag{5b}$$

and

$$\begin{aligned} E^{n(P)}\left(a_t^n|\theta_t^n, l_t^n, \mathbf{l}_t, \mathbf{z}_t\right) &= E^{n,0(P)}\left(a_t^n|\theta_t^n, l_t^n, \mathbf{l}_t\right) \\ &\quad + \sum_{m\in\mathbf{M}} E^{n,m(P)}\left(a_t^n|\theta_t^n, l_t^n, l_t^m, z_t^m\right). \end{aligned} \tag{5c}$$

In (5b), $E^{n,0(I)}$ is the energy spent by peer P_n on transmitting the task, given by

$$E^{n,0(I)}\left(a_t^n|\theta_t^n, l_t^n, \mathbf{l}_t, \mathbf{z}_t\right) = p^n \theta_t^{n(I)} \sum_{m\in\mathbf{M}} \frac{z_t^m \mathbf{1}_{l_t^n\in\mathbf{R}_m(l_t^m)} \mathbf{1}_{x_t^n=m}}{R_U^{n,m}(\mathbf{x}_t)} \tag{6a}$$

and $E^{n,m(I)}$ is the energy consumed by the terrestrial BS_m on forwarding the task to the blockchain server, given by

$$E^{n,m(I)}\left(\mathbf{a}_t|l_t^n, l_t^m\right) = \mathbf{1}_{l_t^n\in\mathbf{R}_m(l_t^m)} \frac{\widetilde{\theta}^{(I)}}{R_f^{m,0}} \sum_{j\in\mathbf{N}} y_t^j p_f^{m,0} \mathbf{1}_{x_t^j=m} \tag{6b}$$

where $p_f^{m,0}$ is the transmit power of a signal sent by BS_m to the blockchain server. In (5c), $E^{n,0(P)}$ is the energy spent by peer P_n or the blockchain server on processing the task, given by

$$\begin{aligned} E^{n,0(P)}\left(a_t^n|\theta_t^n, l_t^n, \mathbf{l}_t\right) &= \left(1-y_t^n\right)\left(1-x_t^n\right)\varphi^n \theta_t^{n(P)} \\ &\quad + y_t^n \sum_{m\in\mathbf{M}_T} \mathbf{1}_{l_t^n\in\mathbf{R}_m(l_t^m)} \mathbf{1}_{x_t^n=m} \varphi^0 \theta_t^{n(I)} \end{aligned} \tag{7a}$$

where φ^n and φ^0 are the energies spent per CPU cycle by peer P_n and a blockchain server, respectively; $E^{n,m(P)}$ is the energy spent by BS_m on processing the task, given by

$$\begin{aligned} &E^{n,m(P)}\left(a_t^n|\theta_t^n, l_t^n, l_t^m, z_t^m\right) \\ &\quad = z_t^m \mathbf{1}_{l_t^n\in\mathbf{R}_m(l_t^m)} \mathbf{1}_{x_t^n=m}\left(1-y_t^n\right)\varphi^m \theta_t^{n(P)} \end{aligned} \tag{7b}$$

where φ^m is the energy spent per CPU cycle by BS_m.

The proofs of Propositions 1 and 2 are given in Appendixes A and B, in the supplementary material, respectively.

V. Stackelberg Model of the BaaS-MEC Support

A. Payoffs of the Peers from the BaaS Support

For any verified task output $\theta_t^{n(O)}$, peer P_n receives a certain transaction fee $\xi^n > 0$. The output $\theta_t^{n(O)}$ is confirmed only if the transaction delay for the task θ_t^n in (4a) does not exceed its completion deadline, i.e., $D^n(\mathbf{a}_t|\theta_t^n, l_t^n, \mathbf{l}_t, \mathbf{z}_t, \mathbf{Q}_t^n) \leq \theta_t^{n(D)}$. Else, if output $\theta_t^{n(O)}$ is discarded due to orphaning, the transaction fee for the task is nullified. If Option 1 is selected for the task θ_t^n, i.e., peer P_n offloads the task processing to some BS_m in its transmission range, the peer must pay a processing cost $c_t^{m(P)}$ to BS_m. If Option 2 is selected, i.e., peer P_n offloads the task processing to the blockchain server via a terrestrial BS_m in its transmission range, the peer must pay a forwarding cost $c_t^{m(P)}$ to BS_m. As such, the instantaneous payoff that peer P_n receives for the task θ_t^n recorded at stage t is given by

$$\begin{aligned} u^n\left(\mathbf{a}_t|\theta_t^n, l_t^n, \mathbf{c}_t, \mathbf{l}_t, \mathbf{z}_t, \mathbf{Q}_t^n\right) &= \mathbf{1}_{D^n\left(\mathbf{a}_t|\theta_t^n, l_t^n, \mathbf{l}_t, \mathbf{z}_t, \mathbf{Q}_t^n\right) \leq \theta_t^{n(D)}} \xi^n \\ &\quad - C^n\left(a_t^n|\theta_t^n, l_t^n, \mathbf{c}_t, \mathbf{l}_t, \mathbf{z}_t, \mathbf{Q}_t^n\right) \end{aligned} \tag{8a}$$

where $\mathbf{c}_t = \{c_t^m\}_{m\in\mathbf{M}} \in \mathbf{C} = \times_{m\in\mathbf{M}} \mathbf{C}^m$ are the stage costs for MEC services of BSs

$$\begin{aligned} &C^n\left(a_t^n|\theta_t^n, l_t^n, \mathbf{c}_t, \mathbf{l}_t, \mathbf{z}_t, \mathbf{Q}_t^n\right) \\ &\quad = \sum_{m\in\mathbf{M}} z_t^m \mathbf{1}_{l_t^n\in\mathbf{R}_m(l_t^m)} \mathbf{1}_{x_t^n=m}\left(\left(1-y_t^n\right)c_t^{m(P)} + y_t^n c_t^{m(I)}\right) \\ &\qquad - \zeta^n E^{n,0}\left(a_t^n|\theta_t^n, l_t^n, \mathbf{l}_t, \mathbf{z}_t\right) \end{aligned} \tag{8b}$$

are the total stage costs paid by peer P_n at stage t; ζ^n is the cost per energy unit paid by peer P_n

$$\begin{aligned} E^{n,0}\left(a_t^n|\theta_t^n, l_t^n, \mathbf{l}_t, \mathbf{z}_t\right) &= E^{n,0(I)}\left(a_t^n|\theta_t^n, l_t^n, \mathbf{l}_t, \mathbf{z}_t\right) \\ &\quad + E^{n,0(P)}\left(a_t^n|\theta_t^n, l_t^n, \mathbf{l}_t\right) \end{aligned} \tag{8c}$$

is the total energy spent by peer P_n or the blockchain server on transacting the task, i.e., the sum of energy consumed by peer P_n on transmitting the task and energy spent by peer P_n or the blockchain server on processing the task.

The payoff u^n in (8a) is a function of: 1) offloading option $a_t^n \in \mathbf{A}^n$ which is selected by peer P_n at the beginning of stage t; 2) task parameters and location $(\theta_t^n, l_t^n) \in \mathbf{\Theta}^n \times \mathbf{L}^n$ of peer P_n at stage t; 3) BSs' parameters $(\mathbf{c}_t, \mathbf{l}_t, \mathbf{z}_t, \mathbf{Q}_t^n)$ observable by peer P_n at the beginning of stage t; and 4) offloading decisions $\mathbf{a}_t^{-n} = \{a_t^i\}_{i \in \mathbf{N} \setminus \{n\}} = \mathbf{A}^{-n} = \times_{i \in \mathbf{N} \setminus \{n\}} \mathbf{A}^i$ of other peers that are unobservable by peer P_n. Since peer P_n is uncertain about the values of $\mathbf{a}_t^{-n}$, it cannot calculate its payoff in (8a) when making its decision a_t^n. However, the peer can construct its own estimate of $\mathbf{a}_t^{-n}$. This estimate or "belief" represents the joint probability distribution $B_t^n = \{B_t^n(\mathbf{a}^{-n})\}_{\mathbf{a}^{-n} \in \mathbf{A}^{-n}}$, where $B_t^n(\mathbf{a}^{-n}) = \Pr_n\{\mathbf{a}_t^{-n} = \mathbf{a}^{-n}\} = \prod_{i \in \mathbf{N} \setminus \{n\}} B_t^n(a^i)$ is the probability that peer P_n assigns to other peers having the offloading profile $\mathbf{a}_t^{-n} = \mathbf{a}^{-n} \in \mathbf{A}^{-n}$ at stage t, and $B_t^n(a^i) = \Pr_n\{a_t^i = a^i\}$ is the marginal probability. Let $s_t^n = (\theta_t^n, l_t^n, \mathbf{c}_t, \mathbf{l}_t, \mathbf{z}_t, \mathbf{Q}_t^n) \in \mathbf{S}^n$ be the parameters that are observable by peer P_n at the beginning of stage t, where $\mathbf{S}^n = \mathbf{\Theta}^n \times \mathbf{L}^n \times_{m \in \mathbf{M}} (\mathbf{C}^m, \mathbf{L}^m, \mathbf{Z}^m) \times \mathbf{\Omega}^n$ is a finite set of possible values of s_t^n; and $\mathbf{\Omega}^n$ is a finite set of possible queue backlogs observable by peer P_n. Then, at any stage t, upon observing the parameters s_t^n, the peer can compute the expected (with respect to its current belief B_t^n) stage payoff from its offloading decision $a_t^n \in \mathbf{A}^n$, as in

$$\begin{aligned} U^n\left(B_t^n, a_t^n | s_t^n\right) &= \sum_{\mathbf{a}^{-n} \in \mathbf{A}^{-n}} B_t^n\left(\mathbf{a}^{-n}\right) u^n\left(a_t^n, \mathbf{a}^{-n} | s_t^n\right) \\ &= \sum_{\mathbf{a}^{-n} \in \mathbf{A}^{-n}} B_t^n\left(\mathbf{a}^{-n}\right) \mathbf{1}_{D^n\left(a_t^n, \mathbf{a}^{-n} | s_t^n\right) \geq \theta_t^{n(D)}} \xi^n \\ &\quad - C^n\left(a_t^n | s_t^n\right). \end{aligned} \tag{9}$$

Since peer P_n is selfish, i.e., it aims to maximize the payoffs from its offloading decisions. If the peer is myopic or short sighted [31], [32], then at any stage t, it selects a decision

$$a_t^n = \underset{a^n \in \mathbf{A}^n}{\operatorname{argmax}}\, U^n\left(B_t^n, a^n | s_t^n\right) \quad \forall n \in \mathbf{N} \tag{10a}$$

which maximizes its expected stage payoff in (9). However, if the peer is fully rational, i.e., long-visional [31], [32], it selects a decision

$$a_t^n = \underset{a^n \in \mathbf{A}^n}{\operatorname{argmax}}\, V^n\left(B_t^n, a^n | s_t^n\right) \quad \forall n \in \mathbf{N} \tag{10b}$$

that maximizes its expected long-term payoff $V^n\left(B_t^n, a^n | s_t^n\right)$. Since the peer remains in the system for indefinitely long time, its long-term payoff is defined by an infinite discounted sum of its stage payoffs in (8a) [32], i.e.,

$$\begin{aligned} V^n\left(B_t^n, a_t^n | s_t^n\right) &= \mathrm{E}\left\{\sum_{\tau=t}^{\infty} \gamma^{\tau-t} u^n\left(a_\tau^n, \mathbf{a}_\tau^{-n} | s_\tau^n\right) | B_t^n, s_t^n, a_t^n\right\} \\ &= \mathrm{E}\left\{\sum_{\tau=t}^{\infty} \gamma^{\tau-t} U^n\left(B_\tau^n, a_\tau^n | s_\tau^n\right) | B_t^n, s_t^n, a_t^n\right\} \\ &= U^n\left(B_t^n, a_t^n | s_t^n\right) + \gamma \sum_{\acute{s}^n \in \mathbf{S}^n} \Pr\left\{\acute{s}^n | s_t^n, a_t^n, B_t^n\right\} \\ &\quad \times V^n\left(B_t^n, a_t^n | \acute{s}^n\right) \\ &= \sum_{\mathbf{a}^{-n} \in \mathbf{A}^{-n}} B_t^n\left(\mathbf{a}^{-n}\right) V^n\left(a_t^n, \mathbf{a}^{-n} | s_t^n\right) \end{aligned} \tag{11a}$$

where $\gamma \in (0, 1]$ is a given discount rate; $\Pr\{\acute{s}^n | s_t^n, a_t^n, B_t^n\}$ is the probability of transiting to $\acute{s}^n \in \mathbf{S}^n$ from (s_t^n, a_t^n, B_t^n); $V^n(a_t^n, \mathbf{a}^{-n} | s_t^n)$ is the expected long-term payoff of peer P_n from the decision profile $(a_t^n, \mathbf{a}^{-n})$ given s_t^n, defined as

$$\begin{aligned} V^n\left(a_t^n, \mathbf{a}^{-n} | s_t^n\right) &= \mathrm{E}\left\{\sum_{\tau=t}^{\infty} \gamma^{\tau-t} u^n\left(a_\tau^n, \mathbf{a}_\tau^{-n} | s_\tau^n\right) | s_t^n, a_t^n, \mathbf{a}^{-n}\right\} \\ &= u^n\left(a_t^n, \mathbf{a}^{-n} | s_t^n\right) + \gamma \sum_{\acute{s}^n \in \mathbf{S}^n} \Pr\left\{\acute{s}^n | s_t^n, a_t^n, \mathbf{a}^{-n}\right\} \\ &\quad \times V^n\left(a_t^n, \mathbf{a}^{-n} | \acute{s}^n\right) \end{aligned} \tag{11b}$$

where $\Pr\{\acute{s}^n | s_t^n, a_t^n, \mathbf{a}^{-n}\}$ is the probability of transiting to $\acute{s}^n \in \mathbf{S}^n$ from $(s_t^n, a_t^n, \mathbf{a}^{-n})$.

B. Payoffs of the Base Stations from MEC Services

Since each BS_m is selfish and fully rational, at any stage t, it aims at maximizing its long-term payoff $V_t^m = \sum_{\tau=t}^{\infty} \gamma^{\tau-t} u_\tau^m$ defined as an infinite discounted sum of stage payoffs (as BS_m stays in the system indefinitely long). Let $a_t^m = (c_t^m, l_t^m, z_t^m) \in \mathcal{A}^m = \mathbf{C}^m \times \mathbf{L}^m \times \mathbf{Z}^m$ denote the stage parameters of BS_m, i.e., cost c_t^m, 3-D location l_t^m, and activity z_t^m at stage t. Then, the instantaneous payoff u_t^m that BS_m receives from its MEC services with parameters $a_t^m \in \mathcal{A}^m$ is given by

$$\begin{aligned} u_t^m &= u^m\left(a_t^m | \mathbf{a}_t, \boldsymbol{\theta}_t, \mathbf{l}_t^p\right) \\ &= z_t^m \sum_{n \in \mathbf{N}} \mathbf{1}_{l_t^n \in \mathbf{R}_m(l_t^m)} \mathbf{1}_{x_t^n = m} \\ &\quad \times \Big(\left(1 - y_t^n\right) c_t^{m(P)} + y_t^n c_t^{m(I)} \\ &\qquad - \zeta^m\left(E_a^m + E^{n,m}\left(\mathbf{a}_t | \theta_t^n, l_t^n, l_t^m, z_t^m\right)\right)\Big). \end{aligned} \tag{12a}$$

In (12a), $\boldsymbol{\theta}_t = \{\theta_t^n\}_{n \in \mathbf{N}} \in \mathbf{\Theta} = \times_{n \in \mathbf{N}} \mathbf{\Theta}^n$ and $\mathbf{l}_t^p = \{l_t^n\}_{n \in \mathbf{N}} \in \mathbf{L}^p = \times_{n \in \mathbf{N}} \mathbf{L}^n$ are, respectively, the task parameters and 2-D locations of peers at stage t; ζ^m is the cost per energy unit payable by BS_m; E_a^m is the energy consumption of active aerial BS_m per mining stage (for a terrestrial BS_m, $m \in \mathbf{M}_T$, we have $E_a^m = 0$); and

$$\begin{aligned} E^{n,m}\left(\mathbf{a}_t | \theta_t^n, l_t^n, l_t^m, z_t^m\right) &= E^{n,m(I)}\left(\mathbf{a}_t | l_t^n, l_t^m\right) \\ &\quad + E^{n,m(P)}\left(a_t^n | \theta_t^n, l_t^n, l_t^m, z_t^m\right) \end{aligned} \tag{12b}$$

is the total energy spent by BS_m on transacting the task, i.e., the sum of energy consumed by BS_m on forwarding the task to the blockchain server and energy spent by peer BS_m on processing the task at its local MEC server.

The payoff in (12a) is a function of: 1) stage parameters a_t^m of BS_m and 2) offloading decisions $\mathbf{a}_t$, task parameters $\boldsymbol{\theta}_t$, and 2-D locations $\mathbf{l}_t^p$ of peers. Note that BS_m does not know the exact values of $(\mathbf{a}_t, \boldsymbol{\theta}_t, \mathbf{l}_t^p)$ prior to selecting its parameters a_t^m at the beginning of stage t, although it can observe these values once all the peers have decided on their offloading options, e.g., at the end of stage t. As such, no BS_m can directly estimate its stage payoff in (12a) when selecting a_t^m. On the other hand, as each peer P_n is selfish, its offloading decision $a_t^n \in \mathbf{A}^n$ should satisfy (10a) for myopic peers or (10b) for fully rational peers. As such, we obtain the following optimization problem that reflects the decision process of BS_m

$$\max_{a_t^m \in \mathcal{A}^m} \quad V^m\left(a_t^m|\mathbf{a}_t\right)$$

$$\text{subject to:} \begin{cases} \text{(10a)}, & \text{for myopic peers} \\ \text{(10b)}, & \text{for fully-rational peers} \end{cases} \tag{13a}$$

where $V^m(a_t^m|\mathbf{a}_t)$ is the expected long-term payoff of BS_m from parameters $a_t^m \in \mathcal{A}^m$ given the decision profile $\mathbf{a}_t$ of the peers, expressed by

$$V^m\left(a_t^m|\mathbf{a}_t\right) = \left\{ \sum_{\tau=t}^{\infty} \gamma^{\tau-t} u^m\left(a_\tau^m \middle| \mathbf{a}_\tau, \boldsymbol{\theta}_\tau, \mathbf{l}_\tau^p\right) \middle| a_t^m, \mathbf{a}_t \right\}. \tag{13b}$$

To solve the problem in (13a), BS_m must be able to compute the expected stage payoff $U^n(B_t^n, a^n|s_t^n)$ or expected long-term payoff $V^n(B_t^n, a^n|s_t^n)$ of each peer P_n. Both the stage and long-term payoffs depend on the peer's belief B_t^n and parameters $s_t^n = (\theta_t^n, l_t^n, \mathbf{c}_t, \mathbf{l}_t, \mathbf{z}_t, \mathbf{Q}_t^n) = (\theta_t^n, l_t^n, a_t^m, \boldsymbol{a}_t^{-m}, \mathbf{Q}_t^n)$, where $\boldsymbol{a}_t^{-m} = \{a_t^i\}_{i \in \mathbf{M} \setminus \{m\}} = \{c_t^i, l_t^i, z_t^i\}_{i \in \mathbf{M} \setminus \{m\}} \in \mathcal{A}^{-m}$ are the parameters of all BSs other than BS_m. Note that BS_m knows neither the peer's beliefs B_t^n nor the parameters $\boldsymbol{a}_t^{-m}$ selected by other BSs, as there is no information exchange among BSs. As a result, no BS_m can directly estimate the expected payoffs of peers and, thus, their decisions $\mathbf{a}_t$. Therefore, (13a) casts as a stochastic optimization problem where the decisions $\mathbf{a}_t$, task parameters $\boldsymbol{\theta}_t$, and 2-D locations $\mathbf{l}_t^p$ of peers are the random variables—the values of $(\mathbf{a}_t, \boldsymbol{\theta}_t, \mathbf{l}_t^p)$ can be observed by BS_m only after it has selected its own parameters a_t^m.

C. Stackelberg Model of the BaaS-MEC Support

Based on the provided description of the BaaS-MEC model, the interactions between the BSs and peers can be modeled as a stochastic Stackelberg game with multiple leaders. The game is played repeatedly at each stage of the mining process. In the game, both the leaders/BSs and followers/peers act selfishly and independently. At the beginning of stage t, each leader/BS_m independently, without coordinating its actions with other BSs, selects its stage parameters $a_t^m = (c_t^m, l_t^m, z_t^m) \in \mathcal{A}^m$ maximizing its expected long-term payoff V^m by solving the problem in (13a). Upon observing the stage parameters $\boldsymbol{a}_t = \{a_t^m\}_{m \in \mathbf{M}} \in \mathcal{A}$ of BSs, each follower/peer P_n independently, without coordinating its actions with other peers, selects the offloading option $a_t^n = (y_t^n, x_t^n) \in \mathbf{A}^n$ which satisfies (10a) if the peer is myopic or (10b) if the peer is fully rational. As such, the action a_t^m of BS_m is observable only by peers, but not by other BSs. On the other hand, the action a_t^n of peer P_n is observable only by BSs, but not by other peers. Hence, both BS_m and peer P_n operate under incomplete information about actions $\boldsymbol{a}_t^{-m} \in \mathcal{A}^{-m}$ and $\mathbf{a}_t^{-n} \in \mathbf{A}^{-n}$ of other BSs and peers, respectively. Note that the game is stochastic, since the payoffs of leaders/BSs and followers/peers are defined by the stochastic parameters with unknown transitions. In particular, the payoff u^n of peer P_n depends on parameters $s_t^n = (\theta_t^n, l_t^n, \boldsymbol{a}_t, \mathbf{Q}_t^n)$ which include not only the actions $\boldsymbol{a}_t$ of BSs but also on the peer's task parameters, location (θ_t^n, l_t^n), and queue backlogs $\mathbf{Q}_t^n$. The payoff u^m of BS_m depends on the peers' actions, task parameters, and locations $(\mathbf{a}_t, \boldsymbol{\theta}_t, \mathbf{l}_t^p)$. The data flow between leaders and followers in the proposed Stackelberg model is shown in Fig. 2.

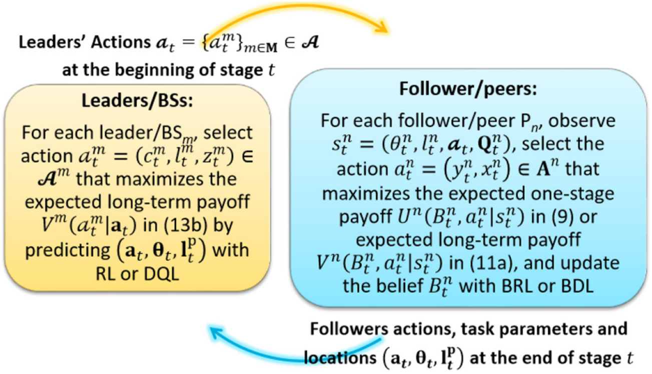
Leaders' Actions $\boldsymbol{a}_t = \{a_t^m\}_{m\in\mathbf{M}} \in \mathcal{A}$ at the beginning of stage t
Leaders/BSs:
For each leader/BS_m, select action $a_t^m = (c_t^m, l_t^m, z_t^m) \in \mathcal{A}^m$ that maximizes the expected long-term payoff $V^m(a_t^m|\mathbf{a}_t)$ in (13b) by predicting $(\mathbf{a}_t, \boldsymbol{\theta}_t, \mathbf{l}_t^{\mathrm{p}})$ with RL or DQL
Follower/peers:
For each follower/peer P_n, observe $s_t^n = (\theta_t^n, l_t^n, \boldsymbol{a}_t, \mathbf{Q}_t^n)$, select the action $a_t^n = (y_t^n, x_t^n) \in \mathbf{A}^n$ that maximizes the expected one-stage payoff $U^n(B_t^n, a_t^n|s_t^n)$ in (9) or expected long-term payoff $V^n(B_t^n, a_t^n|s_t^n)$ in (11a), and update the belief B_t^n with BRL or BDL
Followers actions, task parameters and locations $(\mathbf{a}_t, \boldsymbol{\theta}_t, \mathbf{l}_t^{\mathrm{p}})$ at the end of stage t

Fig. 2. Data flow between leaders and followers in the Stackelberg model.

Recall that in the conventional Stackelberg games, a leader/BS_m selects its action a_t^m by predicting actions $\mathbf{a}_t$ of its followers/peers by backward induction [33]. However, in our game, the action a_t^n selected by peer P_n depends on the actions $\boldsymbol{a}_t^{-m}$ of other BSs and the peer's belief B_t^n, unknown to BS_m. Thus, no BS_m can directly estimate the peers' actions $\mathbf{a}_t$. Instead, it can learn to predict these actions during their repeated interactions with the followers. In the following, we formulate the game to describe the interactions of followers/peers in the Stackelberg model, develop the POMDP model to find the game solution, and propose the novel unsupervised BRL and BDL algorithms that allow each follower to update its belief and estimate its optimal strategy in response to actions of other followers and leaders. Next, we represent the decisions of each leader/BS in the Stackelberg model as an MDP and devise the unsupervised RL and DQL algorithms for the leader to select its actions in the way maximizing its long-term payoff.

VI. Game of Followers in the Stackelberg Model

A. Stochastic Game of Followers With Unobservable Actions

The interactions of followers/peers in the Stackelberg model can be modeled as a stochastic noncooperative game with unobservable players' actions, denoted as $\boldsymbol{\Gamma}$. The game is played repeatedly by a set $\mathbf{N}$ of players, i.e., peers, at each stage of the mining process. At any stage t, the action $a_t^n \in \mathbf{A}^n$ of player P_n represents its offloading decision. As no player P_n observes the actions $\mathbf{a}^{-n} \in \mathbf{A}^{-n}$ of other players, information about $\mathbf{a}^{-n}$ is incomplete and the player constructs its belief $B^n(\mathbf{a}^{-n})$. At the beginning of stage t, player P_n is in state $s_t^n = (\theta_t^n, l_t^n, \boldsymbol{a}_t, \mathbf{Q}_t^n) \in \mathbf{S}^n$ that combines all parameters known to the player: 1) task parameters and location $(\theta_t^n, l_t^n) \in \boldsymbol{\Theta}^n \times \mathbf{L}^n$; 2) actions $\boldsymbol{a}_t \in \mathcal{A}^n$ of leaders/BSs; and 3) queue backlogs $\mathbf{Q}_t^n \in \boldsymbol{\Omega}^n$. As such, the current state s_t^n is observable by the player before it decides on its action a_t^n, but the next state $s_{t+1}^n \in \mathbf{S}^n$ is unknown, i.e., the state is fully observable, but stochastic. As a result, game $\boldsymbol{\Gamma}$ is defined as follows.

Definition 1: A stochastic game with unobservable players' actions is the tuple $\boldsymbol{\Gamma} = (\mathbf{N}, \boldsymbol{\Psi}^n, \mathbf{S}^n, \mathbf{A}^n, \Pr, u^n)$ defined by the following elements: 1) set of N players $\mathbf{N}$ and, for each player P_n; 2) partially observable state space $\boldsymbol{\Psi}^n = \mathbf{S}^n \times \mathbf{A}^{-n}$ which combines observable space $\mathbf{S}^n$ and unobservable space

$\mathbf{A}^{-n}$; 3) set of actions $\mathbf{A}^n$ defined in (2); 4) transition dynamics $\Pr\{\acute{s}^n, \acute{\mathbf{a}}^{-n}|s^n, a^n, \mathbf{a}^{-n}\}$, i.e., probability of transiting to state $(\acute{s}^n, \acute{\mathbf{a}}^{-n}) \in \mathbf{\Psi}^n$ given action $a^n \in \mathbf{A}^n$ taken in state $(s^n, \mathbf{a}^{-n}) \in \mathbf{\Psi}^n$ which can be factored into two conditional distributions, $\Pr\{\acute{s}^n|s^n, a^n, \mathbf{a}^{-n}\}$ and $\Pr\{\acute{\mathbf{a}}^{-n}|\mathbf{a}^{-n}\}$; and 5) instantaneous payoff $u^n(a^n, \mathbf{a}^{-n}|s^n)$ from action $a^n \in \mathbf{A}^n$ taken in state $(s^n, \mathbf{a}^{-n}) \in \mathbf{\Psi}^n$ defined in (8a).

Note that in conventional stochastic games with observable actions, the player's pure strategy $\alpha^n : \mathbf{\Psi}^n \to \mathbf{A}^n$ is a mapping from its state space $\mathbf{\Psi}^n$ to action space $\mathbf{A}^n$ [33]. However, in game $\mathbf{\Gamma}$, state space $\mathbf{\Psi}^n$ comprises two subsets, $\mathbf{S}^n$ and $\mathbf{A}^{-n}$, where $\mathbf{A}^{-n}$ is unobservable. As such, the player's pure strategy $\alpha^n(B^n, s^n) = a^n \in \mathbf{A}^n$ is the mapping from its belief B^n and observable state $s^n \in \mathbf{S}^n$ to the action $a^n \in \mathbf{A}^n$ [32], [34]. The game proceeds as follows: 1) at the beginning of stage t, upon observing state s_t^n, player P_n selects its best-response strategy $\alpha^n(B_t^n, s_t^n) = a_t^n$ and 2) after taking action a_t^n, the player receives a payoff $u^n(a^n, \mathbf{a}^{-n}|s^n)$. The procedure is repeated at each game stage.

One of the possible solution concepts applicable to game $\mathbf{\Gamma}$ is a Bayesian Nash equilibrium (BNE) [33]. In words, in the BNE, no player believes that it can increase its expected stage payoff by deviating from its pure strategy. Formally, a BNE for game $\mathbf{\Gamma}$ can be defined as follows.

Definition 2: A BNE for game $\mathbf{\Gamma}$ is the tuple $(\overline{\boldsymbol{\alpha}}, \overline{\mathbf{B}})$ defined by the players' pure strategy profile $\overline{\boldsymbol{\alpha}} = \{\overline{\alpha}^n\}_{n \in \mathbf{N}} \in \mathbf{A}$ and the beliefs $\overline{\mathbf{B}} = \{\overline{B}^n\}_{n \in \mathbf{N}}$, such that

$$U^n\left(\overline{B}^n, \overline{\alpha}^n|s^n\right) \geq U^n\left(\overline{B}^n, a^n|s^n\right) \quad \forall a^n \in \mathbf{A}^n \quad \forall s^n \in \mathbf{S}^n \quad (14)$$

for all $n \in \mathbf{N}$, where $U^n(B^n, a^n|s^n)$ is an expected stage payoff of player P_n from the action a^n taken in state s^n with belief B^n defined in (9).

Note that as the number of players N and their state–action space $\mathbf{S} \times \mathbf{A}$ are finite, at any stage, game $\mathbf{\Gamma}$ has a BNE [33]. Unfortunately, a BNE concept 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 [32]–[34]. Hence, in order to refine a solution generated by the BNE, the notion of an MPBE [32] can be utilized. In words, MPBE is a BNE where at any stage t, the players' beliefs are consistent, i.e., updated with the Bayesian inference, and the players' 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 strategies of players maximize their expected payoffs. Formally, the MPBE for game $\mathbf{\Gamma}$ can be defined as follows.

Definition 3: The MPBE for the game $\mathbf{\Gamma}$ is the tuple $(\overline{\boldsymbol{\alpha}}, \overline{\mathbf{B}})$ defined by the players' pure strategy profile $\overline{\boldsymbol{\alpha}} = \{\overline{\alpha}^n\}_{n \in \mathbf{N}} \in \mathbf{A}$ and the beliefs $\overline{\mathbf{B}} = \{\overline{B}^n\}_{n \in \mathbf{N}}$, satisfying (14), where at every stage t, the beliefs are updated according to the Bayes' rule based on the last observed transition $(s^n, a^n, \acute{s}^n)$, for all $n \in \mathbf{N}$, as in

$$\begin{aligned} B^n\left(\mathbf{a}^{-n}\right) &= B_{a^n}^{s^n, \acute{s}^n}\left(\mathbf{a}^{-n}\right) \\ &= \frac{\Pr\left\{\acute{s}^n|s^n, a^n, \mathbf{a}^{-n}\right\} B^n\left(\mathbf{a}^{-n}\right)}{\sum_{\acute{\mathbf{a}}^{-n} \in \mathbf{A}^{-n}} \Pr\left\{\acute{s}^n|s^n, a^n, \acute{\mathbf{a}}^{-n}\right\} B^n\left(\acute{\mathbf{a}}^{-n}\right)}. \end{aligned} \quad (15)$$

Similar to the BNE, game $\mathbf{\Gamma}$ admits an MPBE, as the number of players and their state–action space are finite [33]. Note that in the MPBE, players are myopic or short sighted, since their strategies maximize one-stage (rather than long-term) payoffs. In this regard, the PBE concept [32], [33] is more suitable for game $\mathbf{\Gamma}$. In words, PBE is the MPBE in which the players' pure strategies maximize their expected long-term payoffs. Formally, a PBE for game $\mathbf{\Gamma}$ can be defined as follows.

Definition 4: A PBE for game $\mathbf{\Gamma}$ is the tuple $(\overline{\boldsymbol{\alpha}}, \overline{\mathbf{B}})$ defined by the players' pure strategy profile $\overline{\boldsymbol{\alpha}} = \{\overline{\alpha}^n\}_{n \in \mathbf{N}} \in \mathbf{A}$ and the beliefs $\overline{\mathbf{B}} = \{\overline{B}^n\}_{n \in \mathbf{N}}$, such that

$$V^n\left(\overline{B}^n, \overline{\alpha}^n|s^n\right) \geq V^n\left(\overline{B}^n, a^n|s^n\right) \quad \forall a^n \in \mathbf{A}^n \quad \forall s^n \in \mathbf{S}^n \quad (16a)$$

for all $n \in \mathbf{N}$, where $V^n(B^n, a^n|s^n)$ is the expected long-term payoff of player P_n from the action a^n taken in state s^n with belief B^n, defined by

$$\begin{aligned} V^n\left(B^n, a^n|s^n\right) &= U^n\left(B^n, a^n|s^n\right) + \gamma \sum_{\acute{s}^n \in \mathbf{S}^n} \Pr\left\{\acute{s}^n|s^n, a^n, B^n\right\} \\ &\quad \times V^n\left(B_{a^n}^{s^n, \acute{s}^n}, a^n|\acute{s}^n\right) \\ &= \sum_{\mathbf{a}^{-n} \in \mathbf{A}^{-n}} B^n\left(\mathbf{a}^{-n}\right) V^n\left(a^n, \mathbf{a}^{-n}|s^n\right) \end{aligned} \quad (16b)$$

where for any transition $(s^n, a^n, \acute{s}^n)$, the player's belief $B_{a^n}^{s^n, \acute{s}^n}$ is updated with Bayes' rule in (15); and $V^n(a^n, \mathbf{a}^{-n}|s^n)$ is the expected long-term payoff of player P_n from the action profile $(a^n, \mathbf{a}^{-n})$ given state s^n defined in (11b).

Similar to the BNE and MPBE, game $\mathbf{\Gamma}$ admits a PBE, as the number of players and their state–action space are finite [33]. A quantitative comparison between the BNE, PBE, and MPBE is provided, e.g., [33].

B. Bayesian Reinforcement Learning by the Followers

In the repeated game settings, any player can deduce the unknown state transition dynamics and update its belief with stochastic dynamic programming and RL [34]–[36]. In the following, we develop the BRL framework that allows every player/peer to compute its best response strategy and estimate the distributions of observable states and unobservable actions via repeated interactions with each other and with a stochastic environment. We begin with the definition of POMDPs which will be further used to develop our RL model. Formally, a POMDP is defined as follows [32], [36].

Definition 5: POMDP is a tuple $(\mathbf{\Psi}, \mathbf{S}, \mathbf{A}, \Pr, u, \gamma)$ defined by the following elements: 1) partially observable state space $\mathbf{\Psi} = \mathbf{S} \times \mathbf{\Theta}$ which combines observable space $\mathbf{S}$ and unobservable space $\mathbf{\Theta}$; 2) action set $\mathbf{A}$; 3) transition dynamics $\Pr\{\acute{s}, \acute{\theta}|s, a, \theta\} = \Pr\{\acute{s}|s, a, \theta\}\Pr\{\acute{\theta}|\theta\}$—probability of transiting to state $(\acute{s}, \acute{\theta}) \in \mathbf{\Psi}$ given action $a \in \mathbf{A}$ taken in state $(s, \theta) \in \mathbf{\Omega}$; and 4) payoff $u(a|s)$ from action $a \in \mathbf{A}$ taken in state $s \in \mathbf{S}$; v) discount rate $\gamma \in (0, 1]$.

As the transition $\Pr\{\acute{\theta}|\theta\}$ of unknown state $\theta \in \mathbf{\Theta}$ cannot be observed directly, a learner must rely on an observed transition $(s, a, \acute{s})$ to infer an underlying probability distribution or belief $B = \{B(\theta)\}_{\theta \in \mathbf{\Theta}}$, for $B(\theta) = \Pr\{\theta\}$. The BRL problem is to find a pure strategy $\alpha(B) \in \mathbf{A}$ that maximizes the value $V^\alpha(B) = \sum_{t=0}^{\infty} \gamma^t u(\alpha(B_t)|s_t)$, i.e., long-term payoff of a learner. As such, the optimal pure strategy $\overline{\alpha}$ maximizes the value in all beliefs states, i.e., $V^{\overline{\alpha}}(B) \geq V^{\alpha}(B)$, for all α, and satisfies the Bellman optimality equation [32], [36]

$$V^{\overline{\alpha}}(B) = \max_{a\in\mathbf{A}} \left(u(a|s) + \gamma \sum_{\acute{s}\in\mathbf{S}} \Pr\left\{\acute{s}|s, a, B\right\} V^{\overline{\alpha}}\left(B_a^{s,\acute{s}}\right) \right) \quad (17)$$

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

Apparently, the problem of determining the best response of each player in a stochastic game under incomplete information can be cast as a POMDP [34], [35]. In particular, when applied to game $\mathbf{\Gamma}$, the POMDP for player P_n is the tuple $(\mathbf{\Psi}^n, \mathbf{S}^n, \mathbf{A}^n, \Pr, u^n, \gamma)$ with elements defined in Section VI-A. The player's strategy $\alpha^n(B^n, s^n) \in \mathbf{A}^n$ maps from its belief B^n and state s^n to action a^n. Hence, for any player P_n, the BRL problem is to find the best response strategy $\overline{\alpha}^n$ which maximizes the value or expected long-term payoff $V^n(B^n, a^n|s^n)$. We can solve this problem by stochastic dynamic programming [36], [37]. Then, we obtain a dynamic programming algorithm where, at every stage t of mining/learning process, we store the set of $|\mathbf{S}^n \times \mathbf{A}|$ updated values $V_{t+1}^n(a^n, \mathbf{a}^{-n}|s^n)$ indexed by the state $s^n \in \mathbf{S}^n$ and actions $(a^n, \mathbf{a}^{-n}) \in \mathbf{A}$.

The algorithm is as follows. At stage $t = 0$, we initialize the belief B_0^n and value V_0^n. At stage $t = 0, \ldots, T$ (where $T_{rd} \leq T$ is a recursion depth [38] which, in general, can be infinite, i.e., $T_{rd} = T \to \infty$), we repeat the following two steps.

1) *Step 1:* Observe the current state s_t^n and update the belief $B_t^n(\mathbf{a}^{-n}) = B_{a^n}^{s^n,\acute{s}^n}(\mathbf{a}^{-n})$ with Bayes' rule in (15) for the last observed transition $(s^n, a^n, \acute{s}^n)$, where $s^n = s_{t-1}^n$, $a^n = a_{t-1}^n$, $\acute{s}^n = s_t^n$.
2) *Step 2:* Based on the stored values $V_t^n(a^n, \mathbf{a}^{-n}|s^n)$, find the best response strategy $\alpha^n(B_t^n, s_t^n) = a_t^n$ for the current belief B_t^n and state s_t^n, and compute and store values $V_{t+1}^n(a^n, \mathbf{a}^{-n}|s^n)$ by solving the Bellman equation

$$a_t^n = \operatorname*{argmax}_{a^n\in\mathbf{A}^n} V_{t+1}^n\left(B_t^n, a^n|s_t^n\right) \quad (18a)$$

for

$$\begin{aligned} V_{t+1}^n\left(B_t^n, a^n|s_t^n\right) &= U^n\left(B_t^n, a^n|s_t^n\right) \\ &\quad + \gamma \sum_{\acute{s}^n\in\mathbf{S}^n} \Pr\left\{\acute{s}^n|s_t^n, a^n, B_t^n\right\} \\ &\quad \times V_t^n\left(B_{a^n}^{s^n,\acute{s}^n}, a^n|\acute{s}^n\right) \\ &= \sum_{\mathbf{a}^{-n}\in\mathbf{A}^{-n}} B_t^n\left(\mathbf{a}^{-n}\right) V_{t+1}^n\left(a^n, \mathbf{a}^{-n}|s_t^n\right) \\ &\quad \forall a^n \in \mathbf{A}^n \end{aligned} \quad (18b)$$

where

$$\begin{aligned} V_{t+1}^n\left(a^n, \mathbf{a}^{-n}|s_t^n\right) &= u^n\left(a^n, \mathbf{a}^{-n}|s_t^n\right) \\ &\quad + \gamma \sum_{\acute{s}^n\in\mathbf{S}^n} \Pr\left\{\acute{s}^n|s_t^n, a^n, \mathbf{a}^{-n}\right\} \\ &\quad \times V_t^n\left(a^n, \mathbf{a}^{-n}|\acute{s}^n\right) \end{aligned} \quad (18c)$$

is computed and stored for all $(a^n, \mathbf{a}^{-n}) \in \mathbf{A}$.

The above BRL algorithm does not require any additional exploration, as it is implicit in the computation of the Bellman equation in (18b) and (18c). Nonetheless, when the exploration ability is limited, e.g., due to a finite recursion depth T_{rd}, it can be useful to utilize some explicit exploration, e.g., a simple ε-greedy exploration [38] in which, for any arbitrary small $\varepsilon \in [0, 1)$, the best response strategy a_t^n is taken with probability $1 - \varepsilon$, and any other strategy $\acute{a}_t^n \in \mathbf{A}^n \backslash \{a_t^n\}$ is selected with probability ε. The practical implementation of the algorithm, such as the estimation of Bayesian inference in step 1 and state transition probabilities in step 2, is discussed in Appendix C, in the supplementary material. Proposition 3 shows that the proposed BRL algorithm defined by the sequence of iterations $\{V_t^n\}_{t\in\mathbb{N}}$ in (18a)–(18c) converges to the optimal value $\overline{V}^n$, i.e., the value of the best response strategy $\overline{\alpha}^n$ that represents a solution of the Bellman optimality equation.

Proposition 3: The BRL algorithm defined by the sequence of iterations $\{V_t^n\}_{t\in\mathbb{N}}$ in (18a)–(18c) converges to the optimal value $\overline{V}^n$, for all $n \in \mathbf{N}$, with probability one as time tends to infinity.

The proof of Proposition 3 is provided in Appendix D, in the supplementary material. From Proposition 1, we obtain Corollary 1 which shows that the BRL algorithm $\{V_t^n\}_{t\in\mathbb{N}}$ converges to a stable state where the pure strategies and beliefs or the players form the PBE of game $\mathbf{\Gamma}$.

Corollary 1: With probability one as time tends to infinity, the BRL algorithm defined by the sequence of iterations $\{V_t^n\}_{t\in\mathbb{N}}$ in (18a)–(18c), for all $n \in \mathbf{N}$, converges to a stable state where the players' pure strategies and beliefs $(\overline{\boldsymbol{\alpha}}, \overline{\mathbf{B}})$ are in the PBE of game $\mathbf{\Gamma}$.

The proof of Corollary 1 is provided in Appendix E, in the supplementary material. The worst-case computational complexity and convergence rate of the BRL algorithm is established in Proposition 4.

Proposition 4: The BRL algorithm defined by the sequence of iterations $\{V_t^n\}_{t\in\mathbb{N}}$ in (18a)–(18c) has the worst-case time complexity equal to $\mathcal{O}(c^{M+N})$, for $c = \max\{\max_{m\in\mathbf{M}} |\mathcal{A}^m|, \max_{n\in\mathbf{N}} |\mathbf{A}^n|\}$, and the asymptotic rate of convergence equal to $\mathcal{O}(1/t^{1-\gamma})$, for $\gamma > 0.5$ and $\mathcal{O}(\sqrt{\log(\log t)/t})$, otherwise.

The proof of Proposition 4 is provided in Appendix F, in the supplementary material.

VII. Bayesian Deep Learning by the Followers

From Proposition 4, the proposed BRL algorithm has the sublinear convergence rate and the time complexity $\mathcal{O}(c^{M+N})$ exponential in the total number of leaders/BSs and followers/peers $M+N$. Therefore, the BRL algorithm is intractable if the number $M+N$ is large and, to be practically realizable, it requires an approximation. Consequently, in this section, we propose a novel unsupervised polynomial-time BDL algorithm where uncertainties about unobservable states are modeled by BNNs. This allows us to reduce the size of the unobservable state space and, hence, the algorithm complexity. The rest of this section is structured as follows. In Section VII-A, we present the existing BDL framework. In Section VII-B, we devise the BNN model of the decisions of followers/peers. In Section VII-C, we develop the BDL algorithm for the peers.

A. Deep Learning With Bayesian Neural Networks

Although DL algorithms have a proven record in producing accurate solutions of MDPs for various problems (e.g., [39] and [40]), they are not efficient when applied to the POMDPs. The main reason is that the DL methods cannot model epistemic uncertainties about unobservable model parameters which are inherent in POMDPs where such uncertainties are formalized as probability distributions over unobservable states or beliefs [41]–[43]. However, epistemic uncertainties can be efficiently modeled by the BDL methods where these uncertainties are captured with BNNs—NNs with random weights distributed according to a predefined prior [41]. Then, instead of directly optimizing the network weights, as in DL, we average over all possible weights, which is referred to as a marginalization [43]. A brief description of the BDL methods is provided below (for more details, see [41]–[43]).

Consider a data set $(\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1,\ldots,F}$ which 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 observed outputs or observation targets with D_O elements. Let $\Pr\{\mathbf{y}_t|f^{\mathbf{W}}(\mathbf{x}_t)\}$ be the likelihood of 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 the random row output vector of the BNN with some prior distribution over its weights $\mathbf{W}$, e.g., standard normal distribution $\Pr\{\mathbf{W}\} = \mathcal{N}(0, I)$. In regression, the likelihood is usually represented by a Gaussian distribution $\Pr\{\mathbf{y}_t|f^{\mathbf{W}}(\mathbf{x}_t)\} = \mathcal{N}(f^{\mathbf{W}}(\mathbf{x}_t), \sigma^2)$, with mean $f^{\mathbf{W}}(\mathbf{x}_t)$ defined by the model and some given observation noise variance σ^2. In classification, the observed target output $\mathbf{y}_t \in \mathbf{Y} = \{1, \ldots, D_O\}$ is a label in the set $\mathbf{Y} = \{1, \ldots, D_O\}$. As such, a random output $f^{\mathbf{W}}(\mathbf{x}_t) = \widehat{\mathbf{y}}_t = (\hat{y}_t^1, \ldots, \hat{y}_t^{D_O}) \in \mathbb{R}^{1\times D_O}$ of the BNN model is predicted being classified with the label $\mathbf{y}_t = i \in \mathbf{Y}$. To use the model for classification, we pass the BNN output through an elementwise softmax function to obtain the likelihood of the output label $\mathbf{y}_t = i \in \mathbf{Y}$, defined as [41], [43]

$$\Pr\Big\{\mathbf{y}_t = i|f^{\mathbf{W}}(\mathbf{x}_t)\Big\} = \text{Softmax}\Big(i, f^{\mathbf{W}}(\mathbf{x}_t)\Big) = \frac{e^{\hat{y}_t^i}}{\sum_{j=1}^{D_O} e^{\hat{y}_t^j}}. \quad (19)$$

Given the likelihood $\Pr\{\mathbf{y}_t|f^{\mathbf{W}}(\mathbf{x}_t)\}$, we must estimate the posterior $\Pr\{\mathbf{W}|\mathbf{X}, \mathbf{Y}\}$ over weights $\mathbf{W}$ given $(\mathbf{X}, \mathbf{Y})$. In a model-based BRL, the posterior is estimated with the Bayesian inference $\Pr\{\mathbf{W}|\mathbf{X}, \mathbf{Y}\} = \Pr\{\mathbf{W}\}$ $\Pr\{\mathbf{Y}|\mathbf{X}, \mathbf{W}\}/\Pr\{\mathbf{Y}|\mathbf{X}\}$. However, with the BNNs, the computation of inference is intractable, because the marginal probability $\Pr\{\mathbf{Y}|\mathbf{X}\}$ cannot be evaluated analytically [43]. Hence, instead of direct inference, in BDL, a variational inference [44] is utilized where a posterior $\Pr\{\mathbf{W}|\mathbf{X}, \mathbf{Y}\}$ is fitted with a simple distribution $q^\theta(\mathbf{W})$ parameterized by θ, so that $\Pr\{\mathbf{W}|\mathbf{X}, \mathbf{Y}\} \sim \overline{q}^\theta(\mathbf{W})$, where $\overline{q}^\theta(\mathbf{W})$ is an optimal distribution. This replaces the intractable problem of averaging over model weights $\mathbf{W}$ with a simpler problem of optimizing parameters θ. In particular, the optimal parameters $\overline{\theta}$ minimize the Kullback–Leibler (KL) divergence to the true posterior $\Pr\{\mathbf{W}|\mathbf{X}, \mathbf{Y}\}$, i.e.,

$$\overline{q}^\theta(\mathbf{W}) = \min_\theta D_{\text{KL}}\big(q^\theta(\mathbf{W})|\Pr\{\mathbf{W}|\mathbf{X}, \mathbf{Y}\}\big)\big) \quad (20a)$$

where

$$D_{\text{KL}}\Big(q^\theta(\mathbf{W})\Big|\Pr\{\mathbf{W}|\mathbf{X}, \mathbf{Y}\}\big)\Big) = \int q^\theta(\mathbf{W})\log\frac{q^\theta(\mathbf{W})}{\Pr\{\mathbf{W}|\mathbf{X}, \mathbf{Y}\}}\text{d}\mathbf{W} \quad (20b)$$

is the KL divergence, i.e., difference between the distribution $q^\theta(\mathbf{W})$ and the posterior $\Pr\{\mathbf{W}|\mathbf{X}, \mathbf{Y}\}$. Note that minimization of the KL divergence in (20b) is equivalent to maximization of the evidence lower bound (ELBO), given by

$$\begin{aligned}\text{ELBO}(\theta) &= \int q^\theta(\mathbf{W})\log\Pr\{\mathbf{Y}|\mathbf{X}, \mathbf{W}\}\text{d}\mathbf{W} \\ &\quad - D_{\text{KL}}\big(q^\theta(\mathbf{W})|\Pr\{\mathbf{W}\}\big)\big) \\ &= \sum_{t=1}^{F}\int q^\theta(\mathbf{W})\log\Pr\Big\{\mathbf{y}_t|f^{\mathbf{W}}(\mathbf{x}_t)\Big\}\text{d}\mathbf{W} \\ &\quad - D_{\text{KL}}\big(q^\theta(\mathbf{W})|\Pr\{\mathbf{W}\}\big)\big) \\ &\leq \log\Pr\{\mathbf{Y}|\mathbf{X}\} \end{aligned} \quad (20c)$$

where $\log\Pr\{\mathbf{Y}|\mathbf{X}\}$ is the log-evidence. By maximizing the first term in (20c), we maximize the expected log-likelihood. By maximizing the second term, we minimize the KL divergence between the distribution $q^\theta(\mathbf{W})$ and a prior $\Pr\{\mathbf{W}\}$.

A practical approach to approximate inference in the large and complex BNNs is a dropout variational inference [39], [45]–[47] where dropout is interpreted as a variational Bayesian approximation and the approximating distribution $q^\theta(\mathbf{W})$ is a mixture of two Gaussians with small variances and the mean of one of Gaussians fixed at zero. The objective is to minimize the approximation loss, given by [41]

$$\begin{aligned}\mathcal{L}(\theta) &= -\text{ELBO}(\theta) = -\sum_{t=1}^{F}\int q^\theta(\mathbf{W})\log\Pr\Big\{\mathbf{y}_t|f^{\mathbf{W}}(\mathbf{x}_t)\Big\}\text{d}\mathbf{W} \\ &\quad + D_{\text{KL}}\big(q^\theta(\mathbf{W})|\Pr\{\mathbf{W}\}\big)\big) \\ &= -\sum_{t=1}^{F}\log\Pr\Big\{\mathbf{y}_t|f^{\widehat{\mathbf{W}}_t}(\mathbf{x}_t)\Big\} + \int q^\theta(\mathbf{W})\log\frac{q^\theta(\mathbf{W})}{\Pr\{\mathbf{W}\}}\text{d}\mathbf{W} \end{aligned} \quad (21a)$$

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 classification, a log-likelihood of each observed output label $\mathbf{y}_t = i \in \mathbf{Y}$ takes the form

$$\log\Pr\Big\{\mathbf{y}_t = i|f^{\widehat{\mathbf{W}}_t}(\mathbf{x}_t)\Big\} = \log\Big(\text{Softmax}\Big(i, f^{\widehat{\mathbf{W}}_t}(\mathbf{x}_t)\Big)\Big) \quad (21b)$$

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 must estimate the derivatives of loss $\mathcal{L}(\theta)$ with respect to θ, which can be done with the Monte Carlo (MC) estimation [45].

A rather efficient MC estimator is a pathwise derivative estimator (PDE), also called an infinitesimal perturbation, a reparameterization trick, or stochastic backpropagation [41], [43]. In the PDE, $q^\theta(\mathbf{W})$ is reparameterized as a parameter-free distribution $q(\epsilon)$ subject to $\widehat{\mathbf{W}}_t = g(\theta, \epsilon)$, where $\epsilon > 0$ is an infinitesimal number; and $g(\cdot, \cdot)$ is a deterministic differentiable bivariate transformation. For example, for a Gaussian $q^\theta(\mathbf{W}) = \mathcal{N}(\mu, \sigma^2)$, 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, for the normal prior $\Pr\{\mathbf{W}\} = \mathcal{N}(0, I)$, the loss $\mathcal{L}(\theta)$ takes the form [41], [43]

$$\mathcal{L}(\theta, \lambda) = \frac{1}{2}(1-\lambda)\|\theta\|^2 - \sum_{t=1}^{F}\log\Pr\Big\{\mathbf{y}_t|f^{g(\theta,\epsilon)}(\mathbf{x}_t)\Big\} \quad (22a)$$

where $\lambda \in [0,\ 1)$ is a dropout probability or rate, which also must be optimized. In classification, a log-likelihood of every observed output label $\mathbf{y}_t = i \in \mathbf{Y}$ is given by

$$\log \Pr\Big\{\mathbf{y}_t = i | f^{g(\theta,\epsilon)}(\mathbf{x}_t)\Big\} = \log\Big(\mathrm{Softmax}\Big(i, f^{g(\theta,\epsilon)}(\mathbf{x}_t)\Big)\Big) \tag{22b}$$

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

Then, based on the theoretical findings in [41], if the number of data points F in the data set $(\mathbf{X}, \mathbf{Y})$ is sufficiently large, then:

1) loss $\mathcal{L}(\theta, \lambda)$ in (22a) approaches the loss $\mathcal{L}(\theta)$ in (21a), i.e.,

$$\begin{aligned}\mathcal{L}(\theta, \lambda) \underset{F\rightarrow\infty}{\longrightarrow} \mathcal{L}(\theta) &= -\mathrm{ELBO}(\theta)\\ &= D_{KL}\big(q^{\theta}(\mathbf{W}) | \Pr\{\mathbf{W}|\mathbf{X}, \mathbf{Y}\}\big)\big)\end{aligned}$$

2) 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 the Bayesian inference, i.e., $\widehat{\mathbf{W}}_t \longrightarrow \mathbf{W}_t$ and, hence

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

B. Bayesian Neural Network Model for the Followers

Based on the existing BDL framework presented above, in this section, we formulate the BNN model of the decision taken by each follower/peer P_n. To approximate unobservable state transitions, the model adopts a feedforward BNN $f^{\mathbf{W}}(\mathbf{x})$ defined by random weights $\mathbf{W}$ that follow a standard normal prior distribution $\Pr\{\mathbf{W}\} = \mathcal{N}(0, I)$. The data set $(\mathbf{X}_t, \mathbf{Y}_t)= \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=t-T,\ldots,t-1}$ available at stage t of the learning/mining process represents the history about the past $T = \min(t, F)$ consecutive observations or data points $(\mathbf{x}_i, \mathbf{y}_i)$, where F is the size of the observation window. A row input vector $\mathbf{x}_i = (x_i^0, \ldots, x_i^{D_I-1}) = (a_i^n, s_i^n) \in \mathbb{R}^{1\times D_I}$ has $D_I = 7M+5$ elements that represent the action $a_i^n = (y_i^n, x_i^n) \in \mathbb{R}^{1\times 2}$ and state $s_i^n = (\theta_t^n, l_t^n, \boldsymbol{a}_t, \mathbf{Q}_t^n) \in \mathbb{R}^{1\times(7M+3)}$ of peer P_n at stage i. The output or observation target $\mathbf{y}_i \in \mathbf{Y} = \{0, 1\}$ represents the payoff $u_i^n \in \mathbf{U}^n = \{\xi^n - C^n(a_t^n|s_t^n), -C^n(a_t^n|s_t^n)\}$, such that

$$\begin{aligned}u_i^n &= u^n\big(a_i^n, \mathbf{a}_i^{-n}|s_i^n\big)\\ &= \begin{cases}\xi^n - C^n\big(a_i^n|s_i^n\big), D^n\big(a_i^n, \mathbf{a}_i^{-n}|s_i^n\big) \leq \theta_i^{n(D)}\\ -C^n\big(a_i^n|s_i^n\big), D^n\big(a_i^n, \mathbf{a}_i^{-n}|s_i^n\big) > \theta_i^{n(D)}\end{cases}\end{aligned}$$

received by the player at the end of stage i, that can take only two values, $\xi^n - C^n(a_t^n|s_t^n)$ or $-\ C^n(a_t^n|s_t^n)$. In particular, the output $\mathbf{y}_i$ is labeled as

$$\mathbf{y}_i = \begin{cases}1,\ \ u^n\big(a_t^n, \mathbf{a}_t^{-n}|s_t^n\big) = \xi^n - C^n\big(a_t^n|s_t^n\big)\\ 0,\ \ u^n\big(a_t^n, \mathbf{a}_t^{-n}|s_t^n\big) = -\ C^n\big(a_t^n|s_t^n\big).\end{cases} \tag{23a}$$

As such, we have a classification task where, given the data set $(\mathbf{X}_t, \mathbf{Y}_t)$ at stage t, we predict the random output $f^{\mathbf{W}}(\mathbf{x}_t) = \widehat{\mathbf{y}}_t = (\hat{y}_t^0, \hat{y}_t^1) \in \mathbb{R}^{1\times D_O}$, for $D_O = 2$, being classified as 0 or 1. In a feedforward 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 row output vector is given by [41]

$$f^{\mathbf{W}}(\mathbf{x}_t) = \varphi\big(\mathbf{x}_t\mathbf{W}_{(h)} + \mathbf{b}\big)\mathbf{W}_{(v)}^T = \widehat{\mathbf{y}}_t = \Big(\hat{y}_t^0, \hat{y}_t^1\Big) \in \mathbb{R}^{1\times D_O}. \tag{23b}$$

In (23b), the weights represent the tuple $= (\mathbf{W}_{(h)}, \mathbf{W}_{(v)}, \mathbf{b}) \in \mathbb{R}^{(D_I+D_O+1)\times K}$, where $\mathbf{W}_{(h)} = (w_{(h)}^{i,j}) \in \mathbb{R}^{D_I\times K}$ and $\mathbf{W}_{(v)} = (w_{(v)}^{i,j}) \in \mathbb{R}^{D_O\times K}$ are the weights in the hidden and visible layers, respectively; $\mathbf{b} = (b^j) \in \mathbb{R}^{1\times K}$ are the biases; and $\varphi(\cdot)$ is a logistic sigmoid function. Similar to the existing works (e.g., [41]–[43]), we assume the standard normal prior distribution over weights $\mathbf{W}$, i.e., $\Pr\{\mathbf{W}\} = \mathcal{N}(0, I)$, where the prior weight distribution $\Pr\{\mathbf{W}\}$ approximates the belief $B^n(\mathbf{a}^{-n})$ about unobservable actions $\mathbf{a}^{-n}$, i.e., $B^n\ \sim \Pr\{\mathbf{W}\}$. Then, the likelihood of the data point $(\mathbf{x}_t, \mathbf{y}_t)$, given by

$$\Pr\Big\{\mathbf{y}_t|f^{\mathbf{W}}(\mathbf{x}_t)\Big\} = \begin{cases}\mathrm{Softmax}\big(0, f^{\mathbf{W}}(\mathbf{x}_t)\big) = \frac{e^{\hat{y}_t^0}}{e^{\hat{y}_t^0}+e^{\hat{y}_t^1}},\ \ \mathbf{y}_t = 0\\ \mathrm{Softmax}\big(1, f^{\mathbf{W}}(\mathbf{x}_t)\big) = \frac{e^{\hat{y}_t^1}}{e^{\hat{y}_t^0}+e^{\hat{y}_t^1}},\ \ \mathbf{y}_t = 1\end{cases} \tag{24}$$

is equivalent to the probability $\Pr\{u_t^n|s_t^n, a_t^n,\ B_t^n\}$ of the payoff $u_t^n \in \mathbf{U}^n$ received from the action a_t^n executed by the player P_n in state s_t^n, i.e., $\Pr\{u_t^n|s_t^n, a_t^n, B_t^n\} \sim \Pr\{\mathbf{y}_t|f^{\mathbf{W}}(\mathbf{x}_t)\}$. Accordingly, the expected stage payoff $U^n(B_t^n, a^n|s_t^n)$ is approximated by the function $U^n(\mathbf{x}_t|\mathbf{W})$, as in

$$\begin{aligned}U^n\big(B_t^n, a^n|s_t^n\big) &= \sum_{\mathbf{a}^{-n}\in\mathbf{A}^{-n}} B_t^n\big(\mathbf{a}^{-n}\big)u^n\big(a^n, \mathbf{a}^{-n}|s_t^n\big)\\ &= \sum_{u_t^n\in\mathbf{U}^n} u_t^n\Pr\big\{u_t^n|s_t^n, a_t^n, B_t^n\big\}\sim U^n(\mathbf{x}_t|\mathbf{W})\\ &= \sum_{\mathbf{y}_t\in\mathbf{Y}} \mathbf{y}_t\Pr\Big\{\mathbf{y}_t|f^{\mathbf{W}}(\mathbf{x}_t)\Big\}.\end{aligned} \tag{25}$$

Recall that in the model-based BRL, the beliefs are updated with the Bayesian inference, i.e., at any stage t, given the prior $B_t^n\ \sim\ \Pr\{\mathbf{W}\}$, we must estimate a posterior $B_{a_t^n}^{s_t^n,u_t^n}\ \sim\ \Pr\{\mathbf{W}|\mathbf{x}_t, \mathbf{y}_t\}$ for every observed transition $(a_t^n, s_t^n, u_t^n) = (\mathbf{x}_t, \mathbf{y}_t)$ with Bayes' rule. That is, we must compute

$$\begin{aligned}B_{a_t^n}^{s_t^n,u_t^n} &= \frac{\Pr\big\{u_t^n|s_t^n, a_t^n, \mathbf{a}^{-n}\big\}B_t^n\big(\mathbf{a}^{-n}\big)}{\Pr\big\{u_t^n|s_t^n, a_t^n\big\}}\\ &= \frac{\Pr\big\{u_t^n|s_t^n, a_t^n, B_t^n\big(\mathbf{a}^{-n}\big)\big\}B_t^n\big(\mathbf{a}^{-n}\big)}{\sum_{\acute{\mathbf{a}}^{-n}\in\mathbf{A}^{-n}}\Pr\big\{u_t^n|s_t^n, a_t^n, B_t^n\big(\acute{\mathbf{a}}^{-n}\big)\big\}B_t^n\big(\acute{\mathbf{a}}^{-n}\big)}\\ &\quad\sim \Pr\{\mathbf{W}|\mathbf{x}_t, \mathbf{y}_t\}\\ &= \frac{\Pr\big\{\mathbf{y}_t|f^{\mathbf{W}}(\mathbf{x}_t)\big\}\Pr\{\mathbf{W}\}}{\Pr\{\mathbf{y}_t|\mathbf{x}_t\}} = \frac{\Pr\big\{\mathbf{y}_t|f^{\mathbf{W}}(\mathbf{x}_t)\big\}\Pr\{\mathbf{W}\}}{\int\Pr\big\{\mathbf{y}_t|f^{\mathbf{W}}(\mathbf{x}_t)\big\}\Pr\{\mathbf{W}\}\mathrm{d}\mathbf{W}}\end{aligned} \tag{26}$$

which is intractable, as the marginal probability $\Pr\{\mathbf{y}_t|\mathbf{x}_t\} = \int\Pr\{\mathbf{y}_t|f^{\mathbf{W}}(\mathbf{x}_t)\}\Pr\{\mathbf{W}\}\mathrm{d}\mathbf{W}$ cannot be evaluated analytically. Hence, instead of direct inference, we utilize the variational inference. Then, instead of integrating over the weights $\mathbf{W}$, we find the parameters $\overline{\theta}$ of an approximating distribution $q^{\theta}(\mathbf{W})$ which minimize the KL divergence in (20b) or, equivalently, maximize the ELBO in (20c), so that a posterior $\Pr\{\mathbf{W}|\mathbf{x}_t, \mathbf{y}_t\}$

approaches the optimal distribution $\overline{q}^{\theta}(\mathbf{W}) = q^{\overline{\theta}}(\mathbf{W})$ as time tends to infinity, i.e., $\Pr\{\mathbf{W}|\mathbf{x}_t, \mathbf{y}_t\}\xrightarrow[t\to\infty]{}\overline{q}^{\theta}(\mathbf{W})$.

To estimate variational inference, we exploit the PDE. For this, we reparameterize the approximating distribution $q^{\theta}(\mathbf{W})$ by parameter-free distribution $q(\epsilon)$ subject to $\widehat{\mathbf{W}}_t = g(\theta, \epsilon)$. Then, for the Gaussian approximating distribution $q^{\theta}(\mathbf{W}) = \mathcal{N}(\mu, \sigma^2)$, with $\theta = (\mu, \sigma)$, $\widehat{\mathbf{W}}_t = g(\theta, \epsilon) = \mu + \sigma\epsilon$, and $q(\epsilon) = \mathcal{N}(0, I)$, the BDL problem is to find the parameters $(\overline{\theta}, \overline{\lambda})$ which minimize the classification loss $\mathcal{L}(\theta, \lambda)$, given by

$$\begin{aligned}\mathcal{L}(\theta, \lambda) = -\sum_{i=t-T}^{t-1}\Big(&\mathbf{1}_{\mathbf{y}_i=0}\log\Big(\mathrm{Softmax}\Big(0, f^{g(\theta,\epsilon)}(\mathbf{x}_i)\Big)\Big)\\ &+ \mathbf{1}_{\mathbf{y}_i=1}\log\Big(\mathrm{Softmax}\Big(1, f^{g(\theta,\epsilon)}(\mathbf{x}_i)\Big)\Big)\Big)\\ &+ \frac{1-\lambda}{2}\|\theta\|^2.\end{aligned} \tag{27}$$

Then, the BDL problem can be solved by stochastic gradient descent [41], [42]. That is, at any stage t, given the data set $(\mathbf{X}_t, \mathbf{Y}_t)$, we minimize the current loss $\mathcal{L}_t = \mathcal{L}(\theta_t, \lambda_t)$ by updating parameters (θ_t, λ_t) in the direction of a negative gradient until (θ_t, λ_t) converges, as in

$$\theta_{t+1} = \theta_t - \eta_t\nabla_{\theta_t}\mathcal{L}_t \text{ and } \lambda_{t+1} = \lambda_t - \eta_t\nabla_{\lambda_t}\mathcal{L}_t \tag{28a}$$

where η_t is the learning rate of a stochastic gradient descent, such that $\sum_{t=1}^{\infty}\eta_t = \infty$ and $\sum_{t=1}^{\infty}\eta_t^2 < \infty$, typically set as

$$\eta_t = 1/(t+\eta_1)^{\eta_2} \quad \forall\, \eta_1 \in (0, 1) \quad \forall \eta_2 \in (0, 1). \tag{28b}$$

C. Bayesian Deep Learning by the Followers

Based on the presented BNN model, in this section, we develop a novel BDL algorithm for each peer P_n. The proposed algorithm is unsupervised, i.e., it enables a self-organized "online" learning which does not require a prior "off-line" training and/or pre-existing training data sets. To employ the PDE, we utilize the parameter $\theta_t = (\theta_{t(h)}, \theta_{t(v)}, \theta_{t(\mathbf{b})}) \in \mathbb{R}^{(D_I+D_O+1)\times K}$, where $\theta_{t(h)} = (\theta_{t(h)}^{i,j}) \in \mathbb{R}^{D_I\times K}$, $\theta_{t(v)} = (\theta_{t(v)}^{i,j}) \in \mathbb{R}^{D_O\times K}$, and $\theta_{t(\mathbf{b})} = (\theta_{t(\mathbf{b})}^{j}) \in \mathbb{R}^{1\times K}$. The elements $\theta_{t(h)}$ and $\theta_{t(v)}$ define the weights $\widehat{\mathbf{W}}_{(h)} \in \mathbb{R}^{D_I\times K}$ and $\widehat{\mathbf{W}}_{(v)} \in \mathbb{R}^{D_O\times K}$ in hidden and visible layers of a BNN model, respectively. The element $\theta_{t(\mathbf{b})}$ defines the biases $\widehat{\mathbf{b}} \in \mathbb{R}^{1\times K}$ of the BNN. As such, at any stage t, the Gaussian approximating distribution

$$q^{\theta}(\mathbf{W}_t) = \prod_{i,j} q^{\theta_{(h)}^{i,j}}\left(w_{t(h)}^{i,j}\right)\prod_{i,j} q^{\theta_{(v)}^{i,j}}\left(w_{t(v)}^{i,j}\right)\prod_{j} q^{\theta_{(\mathbf{b})}^{i}}\left(b_t^{j}\right)$$

is reparameterized by the standard normal parameter-free distribution, as

$$q(\epsilon_t) = \prod_{i,j} q\Big(\epsilon_{t(h)}^{i,j}\Big)\prod_{i,j} q\Big(\epsilon_{t(v)}^{i,j}\Big)\prod_{j} q\Big(\epsilon_t^{j}\Big) \tag{29a}$$

$$\text{subject to: }\begin{cases}\hat{w}_{(h)}^{i,j} = g\Big(\theta_{t(h)}^{i,j}, \epsilon_{t(h)}^{i,j}\Big) & \forall i \in \{0, \ldots,\ D_I - 1\}\\ \hat{w}_{(v)}^{i,j} = g\Big(\theta_{t(v)}^{k,j}, \epsilon_{t(v)}^{k,j}\Big) & \forall k \in \{0, D_O - 1\}\\ \hat{b}^{j} = g\Big(\theta_{t(\mathbf{b})}^{j}, \epsilon_{t(\mathbf{b})}^{j}\Big)\end{cases} \tag{29b}$$

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

Based on the above, we obtain the BDL algorithm for each peer P_n where, at any stage t, we store the updated parameters (θ_t, λ_t) and data set $(\mathbf{X}_t, \mathbf{Y}_t) = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=t-T,\ldots,t-1}$. At stage $t = 0$, we initialize the parameter θ_0 according to a distribution $\Pr\{\theta_0\} = \Pr\{\widehat{\mathbf{W}}\} = \mathcal{N}(0, I)$, i.e., for all $i \in \{0, \ldots,\ D_I - 1\}$, $k \in \{0, \ldots,\ D_O - 1\}$, $j \in \{0, \ldots,\ K - 1\}$, draw $(D_I + D_O + 1)K$ random samples $\theta_{0(h)}^{i,j}$, $\theta_{0(v)}^{k,j}$, $\theta_{0(\mathbf{b})}^{j}$ from a standard normal distribution. At stage $t = 0, \ldots, T_{rd}$, we repeat the following five steps.

1) *Step 1:* Generate parameter ϵ_t according to $q(\epsilon) = \mathcal{N}(0, I)$, i.e., for all $i \in \{0, \ldots,\ D_I - 1\}$, $k \in \{0, \ldots,\ D_O - 1\}$, $j \in \{0, \ldots, K-1\}$, draw $(D_I+D_O+1)K$ random samples $\epsilon_{t(h)}^{i,j}$, $\epsilon_{t(v)}^{k,j}$, $\epsilon_{t(\mathbf{b})}^{j}$ from the standard normal distribution.
2) *Step 2:* Given the stored parameters (θ_t, λ_t) and the data set $(\mathbf{X}_t, \mathbf{Y}_t) = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=t-T,\ldots,t-1}$, estimate the weights $\widehat{\mathbf{W}} = g(\theta_t, \epsilon_t)$ according to (29b).
3) *Step 3:* Observe the current state s_t^n and find a pure strategy $\alpha_t^n(\theta_t, \epsilon_t) = \operatorname{argmax}_{a^n\in\mathbf{A}^n} U^n(a^n, s_t^n|g(\theta_t, \epsilon_t))$ for weights $\widehat{\mathbf{W}} = g(\theta_t, \epsilon_t)$, that maximizes the expected stage payoff of the player, given by
$$U^n\big(a^n, s_t^n|g(\theta_t, \epsilon_t)\big) = \sum_{\mathbf{y}_t\in\mathbf{Y}}\mathbf{y}_t\Pr\Big\{\mathbf{y}_t|f^{\widehat{\mathbf{W}}}\big(a^n, s_t^n\big)\Big\} \tag{30a}$$
where $f^{\widehat{\mathbf{W}}}(a^n, s_t^n)$ and $\Pr\{\mathbf{y}_t|f^{\widehat{\mathbf{W}}}(a^n, s_t^n)\}$ are given by (23b) and (24), respectively;
4) *Step 4:* Take action $a_t^n = \alpha_t^n(\theta_t, \epsilon_t)$, observe payoff u_t^n, and update the data point as $(\mathbf{x}_t, \mathbf{y}_t) = (a_t^n, s_t^n, u_t^n)$ and the data set as $(\mathbf{X}_{t+1}, \mathbf{Y}_{t+1}) = (\mathbf{X}_t, \mathbf{Y}_t)\cup(\mathbf{x}_t, \mathbf{y}_t)\backslash(\mathbf{x}_{t-T}, \mathbf{y}_{t-T})$.
5) *Step 5:* Given weights $\widehat{\mathbf{W}}$, update the parameters (θ_t, λ_t) in the direction of negative gradient, i.e., for all $j \in \{0, \ldots, K-\ 1\}$, compute
$$\begin{cases}\theta_{t+1(h)}^{i,j} = \theta_{t(h)}^{i,j} - \eta_t\partial\mathcal{L}_t/\partial\theta_{t(h)}^{i,j} & \forall i \in \{0, \ldots,\ D_I - 1\}\\ \theta_{t+1(v)}^{i,k} = \theta_{t(h)}^{i,k} - \eta_t\partial\mathcal{L}_t/\partial\theta_{t(v)}^{i,k} & \forall k \in \{0,\ D_O - 1\}\\ \theta_{t+1(\mathbf{b})}^{j} = \theta_{t(\mathbf{b})}^{j} - \eta_t\partial\mathcal{L}_t/\partial\theta_{t(\mathbf{b})}^{j}\\ \lambda_{t+1} = \lambda_t - \eta_t\partial\mathcal{L}_t/\partial\lambda_t\end{cases} \tag{30b}$$
where the learning rate η_t is defined by (28b).

Similar to the BRL, the above BDL algorithm does not need any additional exploration, as it is implicit in the computation of an output $f^{\widehat{\mathbf{W}}}(a^n, s_t^n)$. Proposition 5 shows that this BDL algorithm defined by the sequence of iterations $\{U_t^n\}_{t\in\mathbb{N}}$ in (30a) and (30b), for $U_t^n = U^n(a^n, s_t^n|g(\theta_t, \epsilon_t))$, converges to the optimal value $\overline{U}^n$ that represents the expected stage payoff of the player from the optimal strategy $\overline{\alpha}^n$.

Proposition 5: The BDL algorithm defined by the sequence of iterations $\{U_t^n\}_{t\in\mathbb{N}}$ in (30a) and (30b) converges to the optimal value $\overline{U}^n$, for all $n \in \mathbf{N}$, with probability one as time tends to infinity.

The proof of Proposition 5 is provided in Appendix G, in the supplementary material. From Proposition 5, we obtain

Corollary 2 which shows that the BDL algorithm $\{U_t^n\}_{t\in\mathbb{N}}$ converges to a stable state where the pure strategies of the players are in the MPBE of game $\boldsymbol{\Gamma}$.

Corollary 2: With probability one as time tends to infinity, the BDL algorithm defined by the sequence of iterations $\{U_t^n\}_{t\in\mathbb{N}}$ in (30a) and (30b), for all $n \in \mathbf{N}$, converges to the stable state in which the players' pure strategies $\overline{\boldsymbol{\alpha}}$ are in the MPBE of game $\boldsymbol{\Gamma}$.

The proof of Corollary 2 is given in Appendix H, in the supplementary material. The worst-case computational complexity and convergence rate of the BDL algorithm are established in Proposition 6.

Proposition 6: The BDL algorithm defined by the sequence of iterations $\{U_t^n\}_{t\in\mathbb{N}}$ in (30a) and (30b) has the worst-case time complexity of $\mathcal{O}(M^2KT)$, where K is the number of neurons and T is the number of data points in the BNN model, and the convergence rate equal to $\mathcal{O}(1/t^{1-[1/(2-\lambda)]})$, where $\lambda \in [0, 1)$ is the average dropout rate.

The proof of Proposition 6 is presented in Appendix I, in the supplementary material. From Proposition 6, the proposed BDL algorithm has a sublinear convergence rate and a polynomial time complexity $\mathcal{O}(M^2KT)$ defined by the number of leaders/BSs M, number of neurons K, and number of data points T.

VIII. Game of Leaders in the Stackelberg Model

A. Reinforcement Learning by the Leaders

Each leader/BS$_m$ in the Stackelberg model selects its action $a_t^m = (c_t^m, l_t^m, z_t^m) \in \boldsymbol{\mathcal{A}}^m$ represented by stage parameters of BS$_m$ by solving an optimization problem in (13a). To solve this problem directly, BS$_m$ should predict the task parameters and 2-D location (θ_t^n, l_t^n), and the best response $a_t^n = a^n(B_t^n, s_t^n) = \mathrm{argmax}_{a^n\in\mathbf{A}^n} V^n(B_t^n, a^n|s_t^n)$ of every follower/peer P_n at any stage t, which is a function of: 1) the peer's belief B_t^n and 2) the peer's state $s_t^n = (\theta_t^n, l_t^n, a_t^m, \boldsymbol{a}_t^{-m}, \mathbf{Q}_t^n)$ depending on actions $\boldsymbol{a}_t^{-m} \in \boldsymbol{\mathcal{A}}^{-m}$ of other BSs. Since the values of B_t^n and $\boldsymbol{a}_t^{-m}$ are unobservable by BS$_m$, the direct estimation of best responses $\mathbf{a}_t$ is not possible. Hence, (13a) is the stochastic optimization problem defined by a random state $s_t = (\mathbf{a}_t, \boldsymbol{\theta}_t, \mathbf{l}_t^p) \in \mathcal{S}$ which represents the observable, but stochastic best responses $\mathbf{a}_t$, task parameters $\boldsymbol{\theta}_t$, and locations $\mathbf{l}_t^p$ of peers, as the leader observes the values of s_t only after taking its action a_t^m, e.g., at the end of stage t. As such, the problem in (13a) represents an MDP defined by the observable random state $s \in \mathcal{S}$ with unknown transition dynamics $\Pr\{\acute{s}|s, a^m\}$. The MDP for this problem can be defined as follows.

Definition 6: The MDP for the problem in (13a) is the tuple $(\mathcal{S}, \boldsymbol{\mathcal{A}}^m, \Pr, u^m, \gamma)$ defined by the following elements: 1) fully observable state space $\mathcal{S} = \mathbf{A} \times \boldsymbol{\Theta} \times \mathbf{L}^p$ equivalent to the set of possible actions, task parameters, and 2-D locations of peers; 2) action space $\boldsymbol{\mathcal{A}}^m = \mathbf{C}^m \times \mathbf{L}^m \times \mathbf{Z}^m$ equivalent to the set of possible stage costs, 3-D locations, and activity assignments of BS$_m$; 3) transition dynamics $\Pr\{\acute{s}|s, a^m\}$, i.e., probability of transiting to state $\acute{s} \in \mathcal{S}$ given action $a^m \in \boldsymbol{\mathcal{A}}^m$ taken by BS$_m$ in state $s \in \mathcal{S}$; 4) payoff $u^m(a^m|\acute{s})$ received by BS$_m$ from action a^m when transiting to state $\acute{s} \in \mathcal{S}$ defined in (12a) and 5) discount rate $\gamma \in (0, 1]$.

The RL problem is to find the optimal pure strategy $\overline{\alpha}^m(s) \in \boldsymbol{\mathcal{A}}^m$ mapping from the state space $\mathcal{S}$ to action space $\boldsymbol{\mathcal{A}}^m$, which maximizes the value, i.e., the expected long-term payoff of BS$_m$, given by

$$\begin{aligned} V^m\big(a^m|s\big) &= \mathrm{E}\left\{\sum_{t=0}^{\infty} \gamma^t u^m\big(a_t^m|s_t\big)|s_0 = s, a_0^m = a^m\right\} \\ &= \sum_{\acute{s}\in\mathcal{S}} \Pr\big\{\acute{s}|s, a^m\big\}\big(u^m\big(a^m|\acute{s}\big) + \gamma V^m\big(a^m|\acute{s}\big)\big) \quad (31a)\end{aligned}$$

and represents a solution of the Bellman optimality equation

$$\begin{aligned} \overline{V}^m(s) &= \overline{V}^m\big(\overline{\alpha}^m(s)|s\big) \\ &= \max_{a^m\in\boldsymbol{\mathcal{A}}^m} \sum_{\acute{s}\in\mathcal{S}} \Pr\big\{\acute{s}|s, a^m\big\}\big(u^m\big(a^m|\acute{s}\big) + \gamma \overline{V}^m\big(a^m|\acute{s}\big)\big). \end{aligned} \quad (31b)$$

The above problem can be solved by dynamic programming. In this case, we obtain the RL algorithm where, at every stage t, we store a set of $|\boldsymbol{\mathcal{A}}^m \times \mathcal{S}|$ updated values $V_t^m(a^m|s)$ indexed by action $a^m \in \boldsymbol{\mathcal{A}}^m$ and state $s \in \mathcal{S}$. In the algorithm, to estimate unknown transition dynamics $\Pr\{\acute{s}|s, a^m\}$, we use a counting variable $\mu_t^m(\acute{s}, s, a^m)$ to count the number of times that each transition $(\acute{s}, s, a^m)$ has occurred and approximate $\Pr\{\acute{s}|s, a^m\}$ as

$$\Pr\big\{\acute{s}|s, a^m\big\} \sim \widehat{\Pr}_t\big\{\acute{s}|s, a^m\big\} = \frac{\mu_t^m\big(\acute{s}, s, a^m\big)}{\sum_{\acute{s}\in\mathcal{S}} \mu_t^m\big(\acute{s}, s, a^m\big)} \quad (32a)$$

for any $a^m \in \boldsymbol{\mathcal{A}}^m$, $\acute{s}, s \in \mathcal{S}$, where

$$\mu_t^m\big(\acute{s}, s, a^m\big) = \mu_{t-1}^m\big(\acute{s}, s, a^m\big) + \mathbf{1}_{\acute{s}=s_t, s=s_{t-1}, a^m=a_t^m}. \quad (32b)$$

The algorithm is as follows. At stage $t = 0$, we initialize the values V_0^m and the counting variables $\mu_0^m = 1$. At stage $= 0, \ldots, T_{rd}$, we repeat three steps below.

1) *Step 1:* Based on the stored values $V_t^m(a^m|s)$, find the optimal pure strategy $\alpha^m(s)$ for current state $s = s_{t-1}$ by solving the Bellman equation

$$\alpha^m(s) = \underset{a^m\in\boldsymbol{\mathcal{A}}^m}{\mathrm{argmax}}\, V_{t+1}^m\big(a^m|s\big) \quad (33a)$$

where $V_{t+1}^m(a^m|s)$ is computed and stored for each action $a^m \in \boldsymbol{\mathcal{A}}^m$ and state $s \in \mathcal{S}$, as in

$$V_{t+1}^m\big(a^m|s\big) = \sum_{\acute{s}\in\mathcal{S}} \Pr\big\{\acute{s}|s, a^m\big\}\big(u^m\big(a^m|\acute{s}\big) + \gamma V_t^m\big(a^m|\acute{s}\big)\big). \quad (33b)$$

2) *Step 2:* Take optimal action $a_t^m = \alpha^m(s)$ with probability $1-\varepsilon$; another action $a_t^m \in \boldsymbol{\mathcal{A}}^m \backslash \alpha^m(s)$ with probability ε.

3) *Step 3:* Observe the state $\acute{s} = s_t$ resulting from action a_t^m taken in state $s = s_{t-1}$ and update the transition dynamics $\widehat{\Pr}_t\{\acute{s}|s, a^m\}$ according to (32a) and (32b).

The convergence of the proposed RL algorithm defined by the sequence of iterations $\{V_t^m\}_{t\in\mathbb{N}}$ in (33a) and (33b) to the optimal value $\overline{V}^m$, i.e., the value of optimal pure strategy $\overline{\alpha}^m$ which represents a solution of the Bellman optimality equation in (31b) or, equivalently, a solution of the optimization problem in (15a) is established in Proposition 7.

Proposition 7: The RL algorithm defined by the sequence of iterations $\{V_t^m\}_{t\in\mathbb{N}}$ in (33a) and (33b) converges to the optimal value $\overline{V}^m$, for all $m \in \mathbf{M}$, with probability one as time tends to infinity.

The proof of Proposition 7 is provided in Appendix J, in the supplementary material. The worst-case computational complexity and convergence rate of the RL algorithm is established in Proposition 8.

Proposition 8: The RL algorithm defined by the sequence of iterations $\{V_t^m\}_{t\in\mathbb{N}}$ in (33a) and (33b) has the worst-case time complexity equal to $\mathcal{O}(c^N)$, for $c = \max_{n\in\mathbf{N}} |\mathbf{A}^n \times \mathbf{\Theta}^n \times \mathbf{L}^n|$, and the asymptotic convergence rate equal to $\mathcal{O}(1/t^{1-\gamma})$, for $\gamma > 0.5$ and $\mathcal{O}(\sqrt{\log(\log t)/t})$, otherwise.

The proof of Proposition 8 is provided in Appendix K, in the supplementary material. As such, we obtain the hierarchical RL (HRL) procedure for the leaders/BSs and the followers/peers in the Stackelberg model. In this procedure, at the beginning of stage t, each leader/BS$_m$ realizes the action $a_t^m = \operatorname{argmax}_{a^m\in\mathcal{A}^m} V_{t+1}^m(a^m|s)$ that maximizes its value according to the RL algorithm. Given the leaders' actions $\boldsymbol{a}_t$, each follower/peer P_n selects its best-response action $a_t^n = \operatorname{argmax}_{a^n\in\mathbf{A}^n} V_{t+1}^n(B_t^n, a^n|s_t^n)$ according to the BRL algorithm. Consequently, the HRL is defined by the sequence of iterations $\{\{V_t^m\}_{m\in\mathbf{M}}, \{V_t^n\}_{n\in\mathbf{N}}\}_{t\in\mathbb{N}}$. The convergence of the HRL to the corresponding optimal values $\{\{\overline{V}^m\}_{m\in\mathbf{M}}, \{\overline{V}^n\}_{n\in\mathbf{N}}\}$ follows from the convergence of the RL and BRL algorithms.

B. Deep Q-Learning by the Leaders

From Proposition 8, the proposed RL algorithm has a sublinear convergence rate and the complexity $\mathcal{O}(c^N)$ exponential in the number of followers/peers N. Thus, the RL algorithm is intractable if the number N is large and, in order to reduce its complexity, it is essential to reduce the size of the state space $\mathcal{S}$ in the MDP model. In the following, we develop a polynomial-time DQL algorithm which enables us to decrease state space $\mathcal{S}$ by approximating the value V^m with an output $f^{\boldsymbol{\omega}}$ of the NN model. The proposed DQL algorithm is unsupervised, i.e., it can be implemented "online" without a prior training or pre-existing data sets.

In the DQL algorithm, we utilize a feedforward NN model $f^{\boldsymbol{\omega}}(a^m, s)$ with deterministic weights $\boldsymbol{\omega}$ to approximate the value $V^m(a^m|s)$ of action $a^m \in \mathcal{A}^m$ taken by BS$_m$ in state $s \in \mathcal{S}$. A row input vector $(a^m, s) \in \mathbb{R}^{1\times D_I}$ of the model has $D_I = 4N + 6$ elements to define a current action $a^m = a_t^m = (c_t^m, l_t^m, z_t^m) \in \mathbb{R}^{1\times 4}$ and state $s = s_{t-1} = (\mathbf{a}_{t-1}, \boldsymbol{\theta}_{t-1}, \mathbf{l}_{t-1}^p) \in \mathbb{R}^{1\times 4N}$ of BS$_m$ which is observable at the beginning of stage t. The output $\hat{V}_{t+1}^m(a^m|s, \boldsymbol{\omega}) = f^{\boldsymbol{\omega}}(a^m, s) \in \mathbb{R}$ of the NN model with $K \geq 1$ neurons approximates the future value $V_{t+1}^m(a^m|s)$. Thus

$$V_{t+1}^m\big(a^m|s\big) \sim \hat{V}_{t+1}^m\big(a^m|s, \boldsymbol{\omega}\big) = f^{\boldsymbol{\omega}}\big(a^m, s\big) \\ = \varphi\big(\big(a^m, s\big)\boldsymbol{\omega}_{(h)} + \mathbf{b}\big)\boldsymbol{\omega}_{(v)}^{\mathrm{T}} \quad (34a)$$

where the tuple $\boldsymbol{\omega} = (\boldsymbol{\omega}_{(h)}, \boldsymbol{\omega}_{(v)}, \mathbf{b}) \in \mathbb{R}^{(D_I+2)\times K}$ comprises weights $\boldsymbol{\omega}_{(h)} = (\omega_{(h)}^{i,j}) \in \mathbb{R}^{D_I\times K}$ and $\boldsymbol{\omega}_{(v)} = (\omega_{(v)}^{j}) \in \mathbb{R}^{1\times K}$ in the hidden and visible layers of the NN model, respectively, and biases $\mathbf{b} = (b^j) \in \mathbb{R}^{1\times K}$. A pure strategy $\alpha^m(\boldsymbol{\omega}_t) \in \mathcal{A}^m$ at stage t maps from the current weights $\boldsymbol{\omega}_t$ to action a_t^m In particular, BS$_m$ selects a strategy $\alpha^m(\boldsymbol{\omega}_t)$ that maximizes its approximated value, i.e.,

$$\alpha^m(\boldsymbol{\omega}_t) = \operatorname*{argmax}_{a^m\in\mathcal{A}^m} \hat{V}_{t+1}^m\big(a^m|s, \boldsymbol{\omega}_t\big). \quad (34b)$$

As such, at any stage t, the approximation target represents the future value $V_{t+1}^m(a^m|s) \in \mathbb{R}$. This value can be updated at the end of stage t, after the game transits from the current state $s = s_{t-1}$ and action $a^m = a_t^m$ to the next state $\acute{s} = s_t$ resulting in the payoff $u^m(a^m|\acute{s})$ based on the value iteration [48]–[51]

$$V_{t+1}^m\big(a^m|s\big) = V_t^m\big(a^m|s\big) + \mathbf{1}_{\acute{s}=s_t, a^m=a_t^m}\chi_t \\ \times \big(u^m\big(a^m|\acute{s}\big) + \gamma V_t^m\big(a^m|\acute{s}\big) - V_t^m\big(a^m|s\big)\big) \\ \forall a^m \in \mathcal{A}^m, s = s_{t-1}. \quad (34c)$$

The above iteration represents a weighted average of the old value $V_t^m(a^m|s)$ and future information, where the weight χ_t can be updated similar to the learning rate η_t, as in (28b). The DQL problem is to find the weights $\overline{\boldsymbol{\omega}}$ which minimize the loss function, given by [39], [40], [48]–[51]

$$\mathcal{L}(\boldsymbol{\omega}) = \frac{1}{2}\mathrm{E}\left\{\left(\overline{V}^m - \hat{V}^m(\boldsymbol{\omega})\right)^2\right\} \\ = \frac{1}{2}\mathrm{E}\left\{\left(\max_{a^m\in\mathcal{A}^m}\sum_{\acute{s}\in\mathcal{S}}\Pr\{\acute{s}|s, a^m\}\big(u^m\big(a^m|\acute{s}\big) + \gamma\overline{V}^m\big(a^m|\acute{s}\big)\big) \\ -\hat{V}^m\big(\alpha^m(\boldsymbol{\omega})|s, \boldsymbol{\omega}\big)\right)^2\right\} \quad (35a)$$

where $\overline{V}^m = \overline{V}^m(s) = \overline{V}^m(\overline{\alpha}^m(s)|s)$ is an optimal value, i.e., a solution of the Bellman optimality equation in (31b) and the optimization problem in (13a).

This problem can be solved by stochastic gradient descent [39], [40], [45]–[48]. That is, at any stage t, we minimize the current loss $\mathcal{L}_t = \mathcal{L}(\boldsymbol{\omega}_t)$, given by

$$\mathcal{L}_t = \mathcal{L}(\boldsymbol{\omega}_t) = \frac{1}{2}\left(\max_{a^m\in\mathcal{A}^m} V_{t+1}^m\big(a^m|s\big) - \hat{V}_{t+1}^m\big(\alpha^m(\boldsymbol{\omega}_t)|s, \boldsymbol{\omega}_t\big)\right)^2. \quad (35b)$$

After differentiating the loss $\mathcal{L}_t$ with respect to current weights $\boldsymbol{\omega}_t$, we obtain the gradient

$$\nabla_{\boldsymbol{\omega}_t}\mathcal{L}_t = \left(\hat{V}_{t+1}^m\big(\alpha^m(\boldsymbol{\omega}_t)|s, \boldsymbol{\omega}_t\big) - \max_{a^m\in\mathcal{A}^m} V_{t+1}^m\big(a^m|s\big)\right) \\ \times \nabla_{\boldsymbol{\omega}_t}\hat{V}_{t+1}^m\big(\alpha^m(\boldsymbol{\omega}_t)|s, \boldsymbol{\omega}_t\big). \quad (36a)$$

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

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t - \eta_t\nabla_{\boldsymbol{\omega}_t}\mathcal{L}_t \quad (36b)$$

where η_t is set according to (28b), until $\boldsymbol{\omega}_t$ converges.

Based on the above equation, we obtain the following DQL algorithm where, at every stage t, we store the set of updated weights $\boldsymbol{\omega}_t$ and values $V_t^m(a^m|s)$ indexed by action $a^m \in \mathcal{A}^m$ and state $s \in \mathcal{S}$. At stage $t = 0$, we initialize the weights $\boldsymbol{\omega}_0$ and values V_0^m. At stage $= 0, \ldots, T_{rd}$, we repeat the following four steps.

1) *Step 1:* Given current state $s = s_{t-1}$ and weights $\boldsymbol{\omega} = \boldsymbol{\omega}_t$, estimate the output $\hat{V}_{t+1}^m(a^m|s, \boldsymbol{\omega})$ of the NN and a strategy $\alpha^m(\boldsymbol{\omega}_t)$ according to (34a) and (34b), respectively.

2) *Step 2:* With probability $1 - \varepsilon$, take action $a_t^m = \alpha^m(\boldsymbol{\omega}_t)$; with probability ε, take another action $a_t^m \in \boldsymbol{\mathcal{A}}^m \backslash \alpha^m(\boldsymbol{\omega}_t)$.
3) *Step 3:* Observe the next state $\acute{s} = s_t$ and payoff $u^m(a_t^m|\acute{s})$ and, based on the stored values $V_t^m(a^m|\acute{s})$, compute and store the next approximation target $V_{t+1}^m(a^m|s)$using (34c).
4) *Step 4:* Given $V_{t+1}^m(a^m|s)$, update gradient $\nabla_{\boldsymbol{\omega}_t}\mathcal{L}_t$ in (36a), and compute and store the weights $\boldsymbol{\omega}_{t+1}$ using (36b).

The convergence of the proposed DQL algorithm represented by the sequence of iterations $\{\hat{V}_t^m\}_{t\in\mathbb{N}}$ in (34a)–(34c) and (36a) and (36b), with $\hat{V}_t^m = \hat{V}_t^m(a^m|s, \boldsymbol{\omega})$, to the optimal value $\overline{V}^m$ is established in Proposition 9.

Proposition 9: The DQL algorithm defined by the sequence of iterations $\{\hat{V}_t^m\}_{t\in\mathbb{N}}$ in (34a)–(34c) and (36a)–(36b) converges to the optimal value $\overline{V}^m$, for all $m \in \mathbf{M}$, with probability one as time tends to infinity.

The proof of Proposition 9 is provided in Appendix L, in the supplementary material. The worst-case computational complexity and convergence rate of the BDL algorithm is established in Proposition 10.

Proposition 10: The DQL algorithm represented by the sequence of iterations $\{\hat{V}_t^m\}_{t\in\mathbb{N}}$ in (34a)–(34c) and (36a) and (36b) has the worst-case time complexity of $\mathcal{O}(NKT|\boldsymbol{\mathcal{A}}^m|)$, where K and T are the numbers of neurons and data points in the NN model, respectively, and the convergence rate of $\mathcal{O}(1/t^{1-[1/(1+\beta)]})$, for $\beta = K\omega_{\max}^{(v)} \max\{1, \omega_{\max}^{(v)}\}/\varepsilon$, where $\omega_{\max}^{(v)} > 0$ is the maximal weight of a visible neuron and $\varepsilon > 0$ is some arbitrary small number.

The proof of Proposition 10 is presented in Appendix M, in the supplementary material. From Proposition 10, the proposed BDL algorithm has the sublinear rate of convergence and the polynomial time complexity $\mathcal{O}(NKT|\boldsymbol{\mathcal{A}}^m|)$ determined by the number of followers/peers N, number of neurons K, number of data points T, and the size of the action space $|\boldsymbol{\mathcal{A}}^m|$.

Accordingly, we obtain a hierarchical deep learning (HDL) procedure for the leaders/BSs and the followers/peers in the Stackelberg model. In this procedure, at the beginning of each stage t, every leader/BS$_m$ adopts the DQL algorithm to decide on the action $a_t^m = \text{argmax}_{a^m\in\boldsymbol{\mathcal{A}}^m} \hat{V}_{t+1}^m(a^m|s, \boldsymbol{\omega}_t)$ which maximizes its value. Each follower/peer P_n selects the best response action $a_t^n = \text{argmax}_{a^n\in\mathbf{A}^n} U^n(a^n, s_t^n|g(\theta_t, \epsilon_t))$ with the BDL algorithm. As such, the HDL is defined by the sequence of iterations $\{\{\hat{V}_t^m\}_{m\in\mathbf{M}}, \{U_t^n\}_{n\in\mathbf{N}}\}_{t\in\mathbb{N}}$. The convergence of the HDL to the corresponding optimal values $\{\{\overline{V}^m\}_{m\in\mathbf{M}}, \{\overline{U}^n\}_{n\in\mathbf{N}}\}$ follows from the convergence of the DQL and BDL algorithms.

IX. Performance Evaluation

A simulation model of the BaaS-MEC system is developed with OPNET package [52]. The MEC model is realized in the long-term evolution advanced (LTE-A) time division duplex (TDD) [53] network. The terrestrial BSs are represented by $M_T = 3$ LTE evolved NodeBs (eNBs): the macrocell eNB labeled as BS$_{N+1}$, microcell eNB labeled as BS$_{N+2}$, and femtocell eNB labeled as BS$_{N+3}$. The aerial BSs are represented by $M_A = 3$ UAVs labeled as BS$_{N+4}$, BS$_{N+5}$ and BS$_{N+6}$. Each eNB is connected to the blockchain server via a 100-km long ITU-T G.657 brand B optical fiber of the capacity 100 wavelengths. The default number of blockchain peers is $N = 10$. Each peer represents a Laptop to serve 100 IoT devices. The IoT devices are represented by the smoke, temperature, motion detector, and image sensors. The system presumes intercell interference and operates in a typical urban environment. The MEC network service area is co-located with the service area of a macrocell eNB. The peers and small-cell eNBs are positioned evenly in the network area. All system payments are counted in units of a digital currency, i.e., bitcoin. The main default parameters of the simulation model are summarized in Table II. The parameters of the LTE-A model, e.g., path loss, antenna gain, noise, and shadowing, are set based on Third Generation Partnership Project (3GPP) specifications [53]. UAV-related parameters

TABLE II
Default Parameters of the Simulation Model

Parameter	**Value**
Numbers of peers N, terrestrial BSs M_{T}, and aerial BSs M_{A}	$N = 10, M_{\text{T}} = 3, M_{\text{A}} = 3$
Service radius of BS$_m$, R_m	$R_{N+1} = 3000$ m (macro-cell BS), $R_{N+2} = 500$ m (micro-cell BS), $R_{N+3} = 50$ m (femto-cell BS), $R_{N+4} = R_{N+5} = R_{N+6} = 200$ m (UAV)
Fiber link settings	100 km long ITU-T G.657 brand B optical fiber of the capacity 100 wavelengths
Computing powers of the blockchain server, MEC server and peer	$\rho^0 = 100$ Gc/s (blockchain server), $\rho^{N+1} = 100$ Gc/s (macro-cell BS), $\rho^{N+2} = 50$ Gc/s (micro-cell BS), $\rho^{N+3} = 20$ Gc/s (femto-cell BS), $\rho^{N+4} = \rho^{N+5} = \rho^{N+6} = 20$ Gc/s (UAV), $\rho^n = 5$ Gc/s, $\forall n \in \mathbf{N}$ (peer)
Energy per CPU cycle consumed by the blockchain server, MEC server and peer	$\varphi^0 = \varphi^n = \varphi^m = 8.2$ nJ, $\forall n \in \mathbf{N}, m \in \mathbf{M}$
Bandwidth of BS$_m$, B_m	$B_{N+1} = 20$ MHz (macro-cell BS), $B_{N+2} = 10$ MHz (micro-cell BS), $B_{N+3} = 5$ MHz (femto-cell BS), $B_{N+4} = B_{N+5} = B_{N+6} = 5$ MHz (UAV)
UL spectrum overlap indicator between BSs, $b_U^{m,i}$	$b_U^{N+1,N+2} = \cdots = b_U^{N+1,N+6} = 1$ (macro-cell BS), $b_U^{N+2,N+3} = b_U^{N+2,N+4} = 0$, $b_U^{N+2,N+5} = b_U^{N+2,N+6} = 1$ (micro-cell BS) $b_U^{N+3,N+4} = b_U^{N+3,N+5} = 0$, $b_U^{N+3,N+6} = 1$ (femto-cell BS) $b_U^{N+4,N+5} = b_U^{N+4,N+6} = 0$, $b_U^{N+5,N+6} = 0$ (UAV)
Transmit powers of peer P$_n$ and BS$_m$, p^n and p^m	$p^n = 23$ dBm, $\forall n \in \mathbf{N}$ (peer), $p^{N+1} = 42$ dBm (macro-cell BS), $p^{N+2} = \cdots = p^{N+6} = 23$ dBm (small-cell BSs and UAVs)
Maximal speed of aerial BS$_m$, $v_{\max}^m$	$v_{\max}^m = 10$ m/s, $\forall m \in \mathbf{M}_{\text{A}}$
Maximal endurance of aerial BS$_m$, $T_{\max}^m$	follows a Poisson distribution with mean of 60 min
Energy consumption per stage of active aerial BS$_m$, E_a^m	$E_a^m = 60$ J, $\forall m \in \mathbf{M}_{\text{A}}$
Mining tasks of peer P$_n$, θ_t^n	Task input size $\theta_t^{n(I)}$, processing size $\theta_t^{n(P)}$, completion deadlines $\theta_t^{n(D)}$, and output size $\theta_t^{n(O)}$ are distributed uniformly in the intervals [1 Mb, 10 Mb], [0.5 Gc, 5 Gc], [0.1 s, 1 s], and [1 kb, 10 kb], respectively. Task inter-arrival times are distributed exponentially with the mean of 5 minutes
Transaction fee per mining task of peer P$_n$, ξ^n	$\xi^n = 500$ bitcoins, $\forall n \in \mathbf{N}$
Cost per energy unit paid by peer P$_n$ and BS$_m$, ζ^n and ζ^m	$\zeta^n = \zeta^m = 1$ bitcoin, $\forall n \in \mathbf{N}, m \in \mathbf{M}$
Discount rate γ, dropout rate λ, learning rate η	$\gamma = 0.9, \lambda = 0.1, \eta_t = 1/(t + 0.5)^{0.7}$
Number of data points in the dataset F and number of neurons in the BNN/NN models K	$F = 100, K = 10$

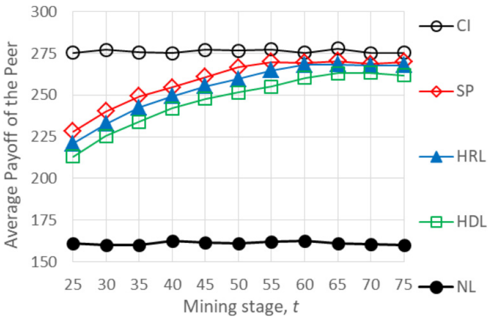

Fig. 3. Time dynamics of the average peer's payoff.

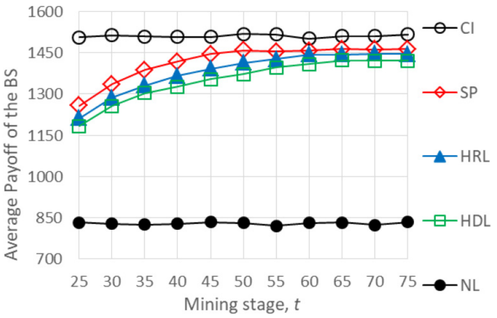

Fig. 4. Time dynamics of the average BS's payoff.

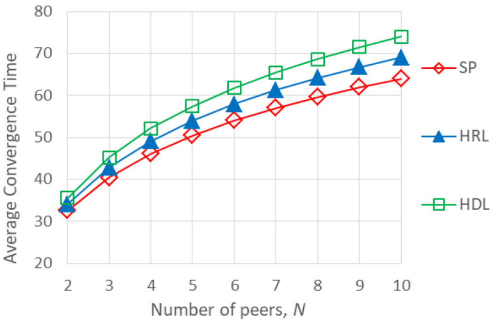

Fig. 5. Convergence time depending on the number of peers.

are based on the common settings (listed, e.g., [15]–[17]). The parameters related to the learning models are based on the recommendations in [38].

In the following, we evaluate the performance of the HRL and HDL procedures for the Stackelberg game by comparing it with the performance of the following models.

1) Stackelberg game under complete information denoted as "CI," where each peer P_n knows current actions $\mathbf{a}_t^{-n}$ of other peers. That is, at any stage t, the peer takes action $\overline{a}_t^n$ which maximizes payoff $u^n(a^n, \mathbf{a}_t^{-n}|s_t^n)$. Each BS$_m$ selects action $\overline{a}_t^m$ maximizing the payoff $u^m(a^m|\overline{\mathbf{a}}_t, \boldsymbol{\theta}_t, \mathbf{l}_t^p)$ by estimating the peers' responses $\overline{\mathbf{a}}_t$ based on the known actions $\overline{\boldsymbol{a}}_t^{-m}$ of other BSs. As such, the actions $\overline{\boldsymbol{a}}_t$ of the BSs form the Nash equilibrium (NE); the peers' actions $\overline{\mathbf{a}}_t$ are the best responses to the BSs' actions. Note that as the CI scenario is infeasible in practical blockchains [15], we use it only to benchmark the performance of our algorithms.
2) Simultaneous play [54] Stackelberg game denoted as "SP," where the BSs and peers select their actions simultaneously and independently. Any peer P_n can observe the past actions $\mathbf{a}_{t-1}^{-n}, \dots, \mathbf{a}_0^{-n}$ of other peers, but does not know their current actions $\mathbf{a}_t^{-n}$. Similarly, any BS$_m$ can observe the past actions $\boldsymbol{a}_{t-1}^{-m}, \dots, \boldsymbol{a}_0^{-m}$ of other BSs, but does not know their current actions $\boldsymbol{a}_t^{-n}$. As such, the decision making of each BS and each peer is modeled as an MDP with fully observable but stochastic state spaces. Note that although this scenario is more realistic than CI, it is still infeasible in practical IoT blockchains because of the massive amount of information exchange required in this case.
3) Stackelberg game played under incomplete information with no learning denoted as "NL." In NL, the assumptions are the same as in HRL and HDL, i.e., no peer P_n and no BS$_m$ know the current and past actions $\mathbf{a}_t^{-n}, \dots, \mathbf{a}_0^{-n}$ and $\boldsymbol{a}_t^{-m}, \dots, \boldsymbol{a}_0^{-m}$ of other peers and BSs, respectively, but no learning is used. Instead, each peer P_n and each BS$_m$ operate under a fixed initial belief B_0^m and about unobservable states. As a result, peer P_n and BS$_m$ select actions $\overline{a}_t^n$ and $\overline{a}_t^m$ which maximize their expected stage payoffs $U^n(B_0^n, a^n|s_t^n)$ and $U^m(a^m|s) = \sum_{\acute{s}\in\mathcal{S}} B_0^n(\acute{s}) u^m(a^m|\acute{s})$. As such, the BSs' actions $\overline{a}_t$ are in the BNE; the peers' actions $\overline{\mathbf{a}}_t$ are best responses (with respect to their initial beliefs) to the BSs' actions.

The time dynamics of the average payoffs of a peer and a BS are shown in Figs. 3 and 4, respectively. Observe that the average payoffs of a peers and a BS are stable in CI and NL. The reason is that in CI and NL, the actions of peers and BSs' are in the equilibrium states. In particular, in CI, the BSs' actions form the NE; the peers' actions are best responses to BSs' actions. On the other hand, in NL, the BSs' actions are in a BNE; the peers' actions are the best responses (with respect to their initial beliefs) to BSs' actions. The results also show that the payoffs of peers and BSs in SP, HRL, and HDL increase consistently with time converging to the stable close-to-optimal levels after around 65, 70, and 75 iterations, respectively. We observe that SP converges faster than HRL and HDL. The results follow from the fact that in SP, the stochastic state s^n of peer P_n has an observable state transition $\Pr\{\acute{s}^n|s^n, a^n\}$. On the contrary, in HRL and HDL, in addition to state s^n, there is also an unobservable state $\mathbf{a}^{-n}$ that is estimated based on the observable state transitions by updating peers' beliefs with the Bayes' rule (in HRL) or its BNN approximation (in HDL). This means that the peers' beliefs (in HRL) and the weights of the BNN (in HDL) must converge before convergence of the best responses. Thus, SP converges faster than HRL and HDL. We also observe that HRL converges faster than HDL. The reason is that in HDL, loss functions are nonconvex with respect to weights of NN or BNN. Thus, in general, a stochastic gradient descent goes through several local minima prior to reach its global optimum. In Fig. 5, convergence times in SP, HRL, and HDL are shown as the functions of the number of peers N. Observe that the convergence time increases with N, because when N is large, it is difficult to accurately estimate the peers' best responses and the algorithms need more time to converge.

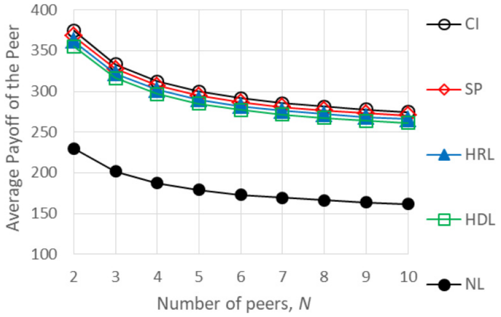

Fig. 6. Average peer's payoff depending on the number of peers.

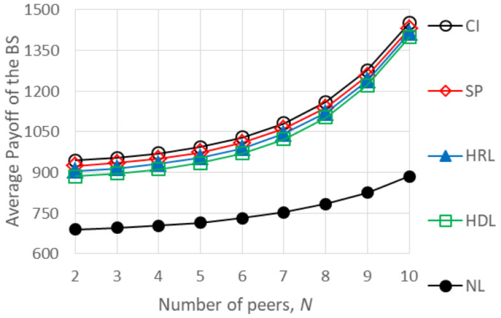

Fig. 7. Average BS's payoff depending on the number of peers.

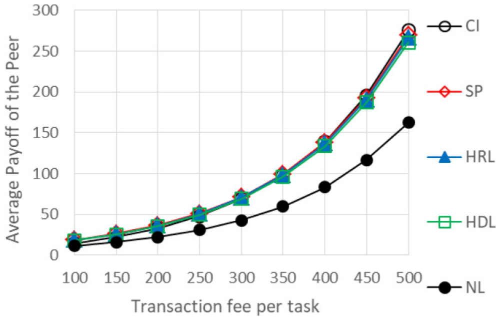

Fig. 8. Average peer's payoff depending on the transaction fee per task.

Figs. 6 and 7 show the average payoffs of a peer and a BS in stable states as functions of the number of peers N. Observe that the payoffs of peers decrease, whereas the payoffs of BSs increase with N. The reason is that when the number of peers is big, more tasks are recorded, forwarded, and offloaded in the system. Hence, some tasks could experience delays exceeding their completion deadlines, in which case the transaction fees for the tasks are nullified leading to a reduced average payoff of a peer. On the contrary, BSs receive payments for MEC services provided to a task regardless of the task completion deadlines. As a result, their payoffs increase when the number of peers and, thus, the number of forwarded and offloaded tasks in the system is big. Figs. 8 and 9 show the average payoffs of a peer and a BS in the stable states as functions of the task transaction fee. Observe that the payoffs of both the peers and BSs increase with the transaction fee, but the growth rates are different—exponential for peers and logarithmic for BSs. The reason is that the peer's payoff is affected directly by the task transaction fee [see (8a)], while the BS's payoff is affected indirectly [see (12a)]. In particular, given the same offloading costs, the peer prefers to process the task locally if the task transaction fee is small (as energy costs are lower than offloading costs) and offload the task to a faster server if the fee is large, in order to increase its payoffs in (8a). Thus, if the task transaction fee is small, more tasks are processed locally by peers leading to reduced BSs' payoffs. On the other hand, if the task transaction fee is high, more tasks are forwarded and offloaded leading to increased BSs' payoffs.

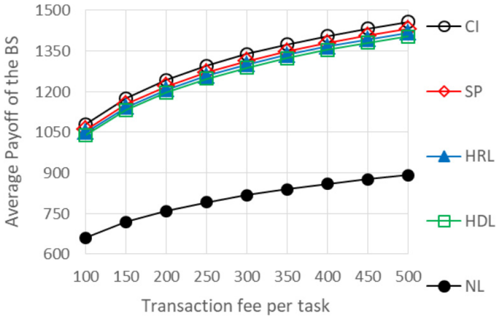

Fig. 9. Average BS's payoff depending on the transaction fee per task.

X. Conclusion

We have proposed a novel unsupervised hierarchical RL and deep learning framework for a stochastic Stackelberg game with multiple leaders under incomplete information. The game models the interactions between the BSs as leaders and peers as followers in the IoT system with BaaS-MEC support. We have developed a hierarchical RL algorithm based on the MDP and POMDP models of the decisions of BSs and peers. We have formulated an HDL algorithm that combines: 1) DQL, where the value of a BS is approximated by the NN and 2) BDL, where uncertainties about unobservable states of the POMDP model for peers are modeled by the BNN. We have shown that the proposed algorithms converge to the stable states in which the peers' actions are the best responses to optimal actions of BSs.

References

[1] J. Cao, D. Zhang, H. Zhou, and P.-J. Wan, "Guest editorial emerging computing offloading for IoTs: Architectures, technologies, and applications," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 3987–3993, Jun. 2019.

[2] S. Fu, Q. Fan, Y. Tang, H. Zhang, X. Jian, and X. Zeng, "Cooperative computing in integrated blockchain based Internet of Things," *IEEE Internet Things J.*, to be published.

[3] O. Novo, "Scalable access management in IoT using blockchain: A performance evaluation," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4694–4701, Jun. 2019.

[4] K. Yeow, A. Gani, R. W. Ahmad, J. J. P. C. Rodrigues, and K. Ko, "Decentralized consensus for edge-centric Internet of Things: A review, taxonomy, and research issues," *IEEE Access*, vol. 6, pp. 1513–1524, 2018.

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

[6] J. Kang, R. Yu, X. Huang, S. Maharjan, Y. Zhang, and E. Hossain, "Enabling localized peer-to-peer electricity trading among plug-in hybrid electric vehicles using consortium blockchains," *IEEE Trans. Ind. Informat.*, vol. 13, no. 6, pp. 3154–3164, Dec. 2017.

[7] J. Kang, Z. Xiong, D. Niyato, D. Ye, D. I. Kim, and J. Zhao, "Toward secure blockchain-enabled Internet of vehicles: Optimizing consensus management using reputation and contract theory," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2906–2920, Mar. 2019.

[8] 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.

[9] W. Chen *et al.*, "Cooperative and distributed computation offloading for blockchain-empowered industrial Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8433–8446, Oct. 2019.

[10] *Amazon Blockchain* . Accessed: Dec. 1, 2019. [Online]. Available: https://docs.aws.amazon.com/managed-blockchain/?id=docs_gateway

[11] *Ali-Baba Cloud BaaS*. Accessed: Dec. 1, 2019. [Online]. Available: https://www.alibabacloud.com/help/doc-detail/85263.htm?spm=a2c63.l28256.a3.1.3abc14a4QOeVRL

[12] B. Omoniwa, R. Hussain, M. A. Javed, S. H. Bouk, and S. A. Malik, "Fog/edge computing-based IoT (FECIoT): Architecture, applications, and research issues," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4118–4149, Jun. 2019.

[13] R. Yang, F. R. Yu, P. Si, Z. Yang, and Y. Zhang, "Integrated blockchain and edge computing systems: A survey, some research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1508–1532, 2nd Quart., 2019.

[14] 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.*, to be published.

[15] A. Asheralieva and D. Niyato, "Hierarchical game-theoretic and reinforcement learning framework for computational offloading in UAV-enabled mobile edge computing networks with multiple service providers," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8753–8769, Oct. 2019.

[16] 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.

[17] M. Mozaffari, W. Saad, M. Bennis, Y.-H. Nam, and M. Debbah, "A tutorial on UAVs for wireless networks: Applications, challenges, and open problems," *arXiv preprint arXiv:1803.00680*, Aug. 2018.

[18] T. Jin, X. Zhang, Y. Liu, and K. Lei, "BlockNDN: A bitcoin blockchain decentralized system over named data networking," in *Proc. 9th Int. Conf. Ubiquitous Future Netw. (ICUFN)*, Milan, Italy, Jul. 2017, pp. 75–80.

[19] W. Wang, D. Niyato, P. Wang, and A. Leshem, "Decentralized caching for content delivery based on blockchain: A game theoretic perspective," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kansas City, MO, USA, May 2018, pp. 1–6.

[20] K. Kotobi and S. G. Bilén, "Blockchain-enabled spectrum access in cognitive radio networks," in *Proc. Wireless Telecommun. Symp. (WTS)*, Chicago, IL, USA, Apr. 2017, pp. 1–6.

[21] A. Lei, H. Cruickshank, Y. Cao, P. Asuquo, C. P. A. Ogah, and Z. Sun, "Blockchain-based dynamic key management for heterogeneous intelligent transportation systems," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 1832–1843, Dec. 2017.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

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

[29] S. Li, M. Yu, S. Avestimehr, S. Kannan, and P. Viswanath, "PolyShard: Coded sharding achieves linearly scaling efficiency and security simultaneously," *arXiv preprint arXiv:1809.10361*, Sep. 2018.

[30] L. Yang, J. Cao, Y. Yuan, Y. Li, A. Han, and A. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," *Perform. Eval. Rev.*, vol. 40, no. 4, pp. 23–32, Aug. 2013.

[31] A. Asheralieva, T. Q. S. Quek, and D. Niyato, "An asymmetric evolutionary Bayesian coalition formation game for distributed resource sharing in a multi-cell device-to-device enabled cellular network," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 3752–3767, Jun. 2018.

[32] 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.

[33] M. J. Osborne and A. Rubenstein. *A Course in Game Theory*. Cambridge, MA, USA: MIT Press, 1994.

[34] S. Sorin, "Stochastic games with incomplete information," in *Stochastic Games and Applications*. Dordrecht, The Netherlands: Springer, 2003, pp. 375–395.

[35] D. Rosenberg, E. Solan, and N. Vieille, "Stochastic games with a single controller and incomplete information," *SIAM J. Control Optim.*, vol. 43, no. 1, pp. 86–110, 2004.

[36] 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.

[37] P. Poupart, N. Vlassis, J. Hoey, and K. Regan, "An analytic solution to discrete Bayesian reinforcement learning," in *Proc. ACM Int. Conf. Mach. Learn.(ICML)*, Jun. 2006, pp. 697–704.

[38] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[39] 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. 24, no. 4, pp. 3039–3071, 4th Quart., 2019.

[40] 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.

[41] Y. Gal, "Uncertainty in deep learning," Ph.D. dissertation, Dept. Eng., Univ. Cambridge, Cambridge, U.K., 2016.

[42] S. Depeweg, J. M. Hernández-Lobato, F. Doshi-Velez, and S. Udluft, "Learning and policy search in stochastic dynamical systems with Bayesian neural networks," *arXiv preprint arXiv:1605.07127*, May 2016.

[43] A. Kendall and Y. Gal, "What uncertainties do we need in Bayesian deep learning for computer vision?" in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5574–5584.

[44] M. D. Hofman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic variational inference," *J. Mach. Learn. Res.*, vol. 14, pp. 1303–1347, May 2013.

[45] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[46] D. Krueger *et al.*, "Zoneout: Regularizing RNNs by randomly preserving hidden activations," *arXiv preprint arXiv:1606.01305*, Jun. 2016.

[47] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Weinberger, "Deep networks with stochastic depth," in *Proc. Eur. Conf. Comput. Vis.*, Oct. 2016, pp. 646–661.

[48] T. Hester *et al.*, "Deep Q-learning from demonstrations," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 3223–3230.

[49] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," in *Proc. ACM ICML*, Jun. 2011, pp. 2829–2838.

[50] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[51] L. Deng and D. Yu, "Deep learning: Methods and applications," *Found. Trends Signal Process.*, vol. 7, nos. 3–4, pp. 197–387, 2014.

[52] *OPNET Simulation and Development Tool*. Accessed: Dec. 1, 2019. [Online]. Available: http://www.opnet.com

[53] *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.

[54] T. Imai, "Essays in revealed preference theory and behavioral economics," Ph.D. dissertation, Humanities Soc. Sci., California Inst. Technol., Pasadena, CA, USA, 2016.

[55] C. Szepesvári, "The asymptotic convergence-rate of Q-learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 1998, pp. 1064–1070.

[56] M. Hardt, B. Recht, and Y. Singer, "Train faster, generalize better: Stability of stochastic gradient descent," *arXiv preprint arXiv:1509.01240*, Sep. 2015.

[57] X. Xu, Y. Zeng, Y. L. Guan, and R. Zhang, "Overcoming endurance issue: UAV-enabled communications with proactive caching," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1231–1244, Jun. 2018.