

# Cooperative Task Offloading and Block Mining in Blockchain-Based Edge Computing With Multi-Agent Deep Reinforcement Learning

Dinh C. Nguyen<sup>1</sup>, Member, IEEE, Ming Ding<sup>2</sup>, Senior Member, IEEE, Pubudu N. Pathirana<sup>3</sup>, Senior Member, IEEE, Aruna Seneviratne<sup>4</sup>, Senior Member, IEEE, Jun Li<sup>5</sup>, Senior Member, IEEE, and H. Vincent Poor<sup>6</sup>, Fellow, IEEE

**Abstract**—The convergence of mobile edge computing (MEC) and blockchain is transforming the current computing services in mobile networks, by offering task offloading solutions with security enhancement empowered by blockchain mining. Nevertheless, these important enabling technologies have been studied separately in most existing works. This article proposes a novel cooperative task offloading and block mining (TOBM) scheme for a blockchain-based MEC system where each edge device not only handles data tasks but also deals with block mining for improving the system utility. To address the latency issues caused by the blockchain operation in MEC, we develop a new Proof-of-Reputation consensus mechanism based on a lightweight block verification strategy. A multi-objective function is then formulated to maximize the system utility of the blockchain-based MEC system, by jointly optimizing offloading decision, channel selection, transmit power allocation, and computational resource allocation. We propose a novel distributed deep reinforcement learning-based approach by using a multi-agent deep deterministic policy gradient algorithm. We then develop a game-theoretic solution to model the offloading and mining competition among edge devices as a potential game, and prove the existence of a pure Nash equilibrium. Simulation results demonstrate the significant system utility improvements of our proposed scheme over baseline approaches.

**Index Terms**—Blockchain, mobile edge computing, task offloading, block mining, deep reinforcement learning

## 1 INTRODUCTION

RECENT advances in Internet of Things (IoT) have promoted the proliferation of numerous mobile applications that mostly rely on edge devices (EDs), e.g., laptops, tablets, and smartphones, to collect data from IoT sensors to serve end users. To meet the increasing computation demands of such users, mobile edge computing (MEC) has been proposed as a promising technique to improve the computation experience of EDs, by offloading computationally intensive tasks to

a nearby MEC server located at a base station (BS) [1]. Mapping each offloading process to a specific application, multiple distributed EDs naturally share computation and communication resources of the BS to handle data tasks without depleting the users' batteries. Task offloading with MEC thus becomes a viable solution to satisfy various EDs' computation demands and enhances the quality of experience (QoE) of end users.

However, the design of an efficient task offloading scheme for MEC systems still faces non-trivial challenges. Each ED always aims to maximize its individual utility by occupying as many edge resources (e.g., channel spectrum, CPU frequency) as possible, which is likely to cause network traffic congestion and user interference. The heterogeneous resource requirements of multiple IoT data tasks, e.g., different resource allocations needed for handling different data tasks, and the heterogeneous features of real-time IoT data tasks, e.g., computation deadlines and data task sizes, pose challenges for the design of offloading strategies for all EDs. Moreover, the lack of prior information on system statistics in practical multi-user MEC systems, e.g., channel state and edge computational resource state, makes it challenging to derive an optimal offloading solution for each ED. Therefore, it is important to develop an intelligent and self-organized offloading scheme to guide offloading actions of all EDs in distributed MEC systems.

Furthermore, the dynamic communications among IoT devices, EDs, and the MEC server and the migration of IoT data tasks across the MEC network potentially cause security vulnerabilities. Recent works [1], [2], [3] have mainly

- Dinh C. Nguyen and Pubudu N. Pathirana are with the School of Engineering, Deakin University, Waurn Ponds, VIC 3216, Australia. E-mail: {cdnguyen, pubudu.pathirana}@deakin.edu.au.
- Ming Ding is with the Data61, CSIRO, Eveleigh, NSW 2015, Australia. E-mail: ming.ding@data61.csiro.au.
- Aruna Seneviratne is with the School of Electrical Engineering and Telecommunications, University of New South Wales (UNSW), Sydney, NSW 2052, Australia. E-mail: a.seneviratne@unsw.edu.au.
- Jun Li is with the School of Electrical and Optical Engineering, Nanjing University of Science and Technology, Nanjing 210094, China. E-mail: jun.li@njjust.edu.cn.
- H. Vincent Poor is with the Department of Electrical and Computer Engineering, Princeton University, Princeton, NJ 08544 USA. E-mail: poor@princeton.edu.

Manuscript received 22 Mar. 2021; revised 17 Sept. 2021; accepted 28 Sept. 2021. Date of publication 14 Oct. 2021; date of current version 6 Mar. 2023.

This work was supported in part by CSIRO Data61, Australia, and in part by the U.S. National Science Foundation under Grant CCF-1908308. The work of Jun Li was supported by the National Natural Science Foundation of China under Grant 61872184.

(Corresponding author: Dinh C. Nguyen.)

Digital Object Identifier no. 10.1109/TMC.2021.3120050

focused on computation offloading designs for task scheduling and resource allocation, with considering security aspects in MEC networks. Fortunately, blockchain has been envisioned as a strong candidate to enhance security of MEC systems [4]. In fact, blockchain is able to provide high degrees of security and trust for MEC by employing community verification among edge nodes via mining mechanisms such as Delegated Proof of Stake (DPoS) [4] without requiring any central authority.

In this context, the use of blockchain is highly desirable to support edge computing systems [5], [6], [7]. Specifically, blockchain decentralizes the MEC system such that edge nodes can communicate with each other via a peer-to-peer network over a decentralized data ledger. Different from traditional MEC systems that often rely on a central server to coordinate MEC operation, blockchain helps build decentralized edge communications without the need for a single authority, which eliminates the risks of single-point failure. This feature is very useful in practical application scenarios, e.g., decentralized edge data sharing and decentralized edge data caching in MEC networks. Another motivation behind the integration of blockchain in MEC is its immutability that makes edge data records, e.g., IoT data, unchangeable once they are stored on the ledger [8]. By deploying immutable transaction ledgers, EDs can establish reliable communications to perform heterogeneous networking and computation, such as large-scale IoT collaborations or mobile edge computing over trustless IoT environments. Moreover, blockchain provides transparency for MEC networks, where blockchain allows the copy of data records to replicate across edge nodes for public validation, which in return enhances data integrity. This feature is particularly suitable for MEC ecosystems where openness and fairness are required. For example, blockchains can offer transparent ledger solutions to support open and secure data delivery and payment for EDs in a fashion such that EDs can trace and monitor transactions.

Moreover, each ED joins the block mining process to maintain the operation of blockchain in MEC. The key purpose of mining is to verify the data transactions, aiming to guarantee the security for the involved edge networks. Accordingly, in the blockchain-based MEC system, EDs perform the mining, and data blocks are secured and chained via an immutable ledger. With more devices mining the blockchain, the security of the edge network increases accordingly.

### 1.1 Related Works

Recently, many edge task offloading solutions have been proposed [9], [10], [11], but these works mostly have considered offloading scenarios with a single agent using traditional convex optimization tools. Deep reinforcement learning (DRL) techniques such as deep Q-learning (DQN) have emerged as a promising alternative, by modelling the offloading problem as a Markov decision process (MDP) and using a deep neural network (DNN) for function approximation [12], [13], [14]. However, these works use only a single agent to handle the entire offloading process which may not work well in large-scale distributed MEC environments. An interesting alternative is to use multi-

agent-DRL (MA-DRL) [15] for supporting intelligent task offloading in MEC networks [16]. The work in [17] proposed a non-cooperative MA-DRL scheme where EDs could build their offloading policies independently. Another study in [18] also suggested an MA-DRL approach for joint data offloading and resource allocation in multiple independent edge clouds. Furthermore, a multi-agent Q-learning algorithm was developed in [19] for a joint computation offloading and resource allocation scheme in edge computing.

In terms of reputation-based DPoS mining design, the work in [4] focused primarily on addressing secure block verification issues in the DPoS mechanism using contract theory. The paper in [20] suggested a fair voting scheme for the DPoS mechanism via vague set theory. Specifically, this work leveraged a general model of transforming vague sets into fuzzy sets to calculate the comprehensive evaluation indices for agent node selection. The paper in [21] proposed a contract theory-based optimization scheme for transaction relaying and DPoS based block verification. The authors of this work focused on formulating two mathematical models: value of transaction relaying and value of block verification, and developed an optimal contract to maximize utility of the miners. Further, the work in [8] proposed a lightweight blockchain-based information trading framework to model interactions between traffic administration and vehicles in a reputation-based DPoS mechanism via a budgeted auction approach. This study considered the optimization of the mining profit by using a truthful budgeted selection and pricing algorithm. However, a design for low-latency block verification in the reputation-based DPoS mechanism has not been developed.

Moreover, research related to task offloading and blockchain mining in MEC networks has been conducted recently. A blockchain-empowered computation offloading scheme was presented in [16] where blockchain was mainly used for data integrity in the offloading. The authors in [22], [23], [24] studied edge offloading schemes for blockchain mining tasks with edge clouds, aiming to enhance the quality of service (QoS) for efficient block mining. Blockchain was also utilized in [25] to support resource trading for the edge task offloading, while the work in [26] considered a cooperative blockchain-MEC system with an actor-critic DRL algorithm. The study in [27] optimized edge computation offloading and resource allocation via a double-dueling deep Q network. The authors in [28] considered a cooperative computation offloading framework for blockchain-based IoT networks. An MA-DRL algorithm was designed which could allow IoT devices to collaboratively explore the offloading environments in order to minimize long-term offloading costs. Another work [29] considered a blockchain-based energy trading scheme to manage the energy trading process toward building a secure energy trading system in Industry 4.0. In [30], the problem of resource trading for blockchain-based IoT was studied by using a two-level Stackelberg game with a credit-based payment with smart contracts. Game theory was also applied in [31] to minimize the economic cost of industrial IoT devices. However, in most existing works [22], [23], [24], [25], [26], [27], [29], [30], [31], the design and optimization of task offloading and blockchain mining were implemented separately, which will result in a sub-optimal performance.

TABLE 1  
The Comparison of the Existing Works and Our Scheme

Design features	Schemes								
	[13]	[16]	[17]	[18]	[26]	[27]	[28]	[31]	Our scheme
Task offloading with blockchain		✓			✓	✓	✓	✓	✓
Intelligent edge task offloading	✓		✓	✓	✓	✓	✓	✓	✓
Cooperative edge task offloading							✓	✓	✓
Lightweight blockchain design									✓
Joint offloading and mining design									✓

## 1.2 Motivations and Our Key Contributions

Despite the recent research efforts in blockchain-MEC designs, there are several limitations in existing works, as highlighted below:

- In distributed blockchain-based MEC systems, traditional single-agent DRL algorithms like DQN [13], [14], [22], [26], [32] face critical challenges caused by diversified and time-varying local environments. In particular, during the training process of DQN, each agent only observes its local information and cannot know the updates from other agents due to non-collaboration. This makes it hard to ensure the stability and convergence of the agents' algorithm [33]. Moreover, this breaks the Markov properties required by the Q-learning algorithm and thus, DQN may not be capable of learning the cooperative offloading policies of EDs. Moreover, the non-cooperative multi-agent DRL solutions [17], [18], [19] may not be able to learn the cooperative policy; and thus resource usage over the edge network is not efficient which limits the overall offloading performance, e.g., offloading utility.
- In addition, in most existing blockchain-based MEC schemes [22], [23], [24], [25], [26], the design and optimization of task offloading and blockchain mining are done separately, which can result in sub-optimal performance. Moreover, the problem of high network latency caused by blockchain mining in the edge offloading system has not been addressed so far [27], [28], [30], [31]. To improve the overall performance, a joint offloading and blockchain design is needed for realizing efficient blockchain-based MEC systems.

Motivated by the aforementioned limitations, we propose a novel cooperative task offloading and blockchain mining (TOBM) scheme for blockchain-based MEC systems enabled by a new MA-DRL solution. Different from existing works [23], [24], [25], [26], [27], [29], [30], [31], we here focus on maximizing the overall system utility as the sum of offloading utility and mining utility. More specifically, each ED handles data tasks collected from its IoT sensors and deals with block mining tasks simultaneously. To reduce the network latency caused by the blockchain integration in the MEC system, we design a new Proof-of-Reputation (PoR) mining mechanism enabled by a lightweight block verification solution. In particular, we develop a novel distributed DRL-based algorithm using a multi-agent deep deterministic policy gradient (MA-DDPG) approach to optimize the overall system utility. The proposed MA-DDPG approach enables the efficient learning of the mutual policy

among cooperative EDs in dynamic environments and high-dimensional system state spaces. Indeed, the proposed MA-DDPG scheme allows EDs to learn mutually the cooperative offloading and mining policy which helps enhance the computation efficiency and thus improves the system utility. To enhance the convergence performance in model training and solve the nonstationary issues caused by the concurrent learning process of all EDs in the multi-agent environment, a centralized learning and decentralized execution solution is adopted. As such, the proposed MA-DRL algorithm is first trained at the centralized MEC server, and the learned model is then executed at EDs in a distributed manner. In fact, the benefits of the MA-DDPG algorithm in edge computing have been proved in recent works for MEC-based Industry 4.0 [34], smart ocean federated learning IoTs [35], and smart grid [36]. However, its potential in blockchain-MEC systems has not been explored so far. The comparison of our paper and the related works via some key features is summarized in Table 1. In a nutshell, the unique contributions of this work are highlighted as follows:

- 1) We propose a novel cooperative TOBM scheme in a blockchain-based MEC system to enable a joint design of task offloading and blockchain mining for improving the overall system utility.
- 2) The details of task offloading are presented, where EDs cooperatively offload their IoT data tasks to the MEC server. Moreover, we propose a new PoR mining mechanism enabled by a lightweight block verification strategy, in order to solve latency issues caused by blockchain adoption in the MEC system.
- 3) In the TOBM scheme, each ED as an intelligent agent to learn cooperatively policies, by jointly considering the offloading decision, channel selection, transmit power allocation, and computational resource allocation with respect to both offloading and mining states for maximizing the system utility. Then, we propose a novel distributed DRL-based approach using an MA-DDPG algorithm to solve the proposed problem based on a centralized learning and decentralized execution strategy.
- 4) We further develop a game-theoretic solution to model the competition among EDs in offloading and mining as a potential game. We then analyze the properties of the formulated game and prove the existence of a pure Nash equilibrium (NE).
- 5) We conduct extensive numerical simulations and compare with the existing schemes to verify the effectiveness of the proposed scheme.

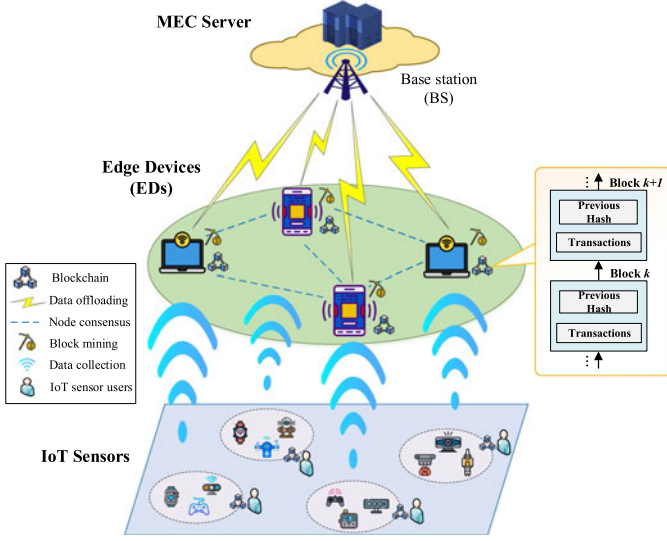


Fig. 1. The cooperative task offloading and block mining architecture in the blockchain-based MEC system.

### 1.3 Paper Organization

The remainder of this paper is organized as follows. Section 2 introduces the system model along with the analysis of network model edge task offloading model. The PoR blockchain consensus mechanism is proposed in Section 3. Based on the offloading and mining design, a joint system utility problem is formulated in Section 4 which is then modelled by a cooperative offloading game. A new MA-DRL algorithm is proposed to solve the formulated offloading game by using an MA-DDPG algorithm. The simulation results are provided in Section 5 and the comparison with other related offloading schemes is also discussed. Finally, Section 6 concludes this article and highlights possible future directions.

## 2 SYSTEM MODEL

In this section, we introduce the network model of the blockchain-based MEC system, and then present the task offloading model.

### 2.1 Network Model

We consider a cooperative TOBM architecture in the blockchain-based MEC system as illustrated in Fig. 1. The BS is equipped with an MEC server to provide computation services for EDs. We denote the set of EDs as  $\mathcal{N} = \{1, 2, \dots, N\}$ . For the sake of simplicity, we assume that each ED  $n \in \mathcal{N}$  has an IoT data task  $Y_n$  to be executed [9], [10], which can be defined by a tuple  $Y_n = (C_n, D_n, \tau_n)$ ,  $n \in \mathcal{N}$ . Herein,  $C_n$  denotes the total computational resource (i.e., the number of the CPU cycles) to accomplish the task  $Y_n$ . Also,  $D_n$  expresses the size of the input data, and  $\tau_n$  specifies the maximum permissible latency to accomplish task  $Y_n$ . In addition to the task offloading function, each ED also participates in the block mining by using a PoR consensus mechanism. The key network components of the blockchain-based MEC system are described as follows:

- **IoT Sensors:** IoT sensors such as cameras, smart meters, and wearables are responsible for sensing physical environments, e.g., entertainment, logistics,

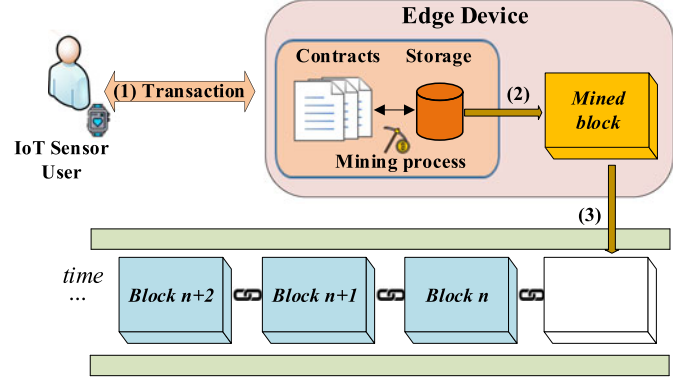


Fig. 2. Illustration of the blockchain operation in our MEC network: (1) An IoT sensor user sends a transaction to its associated ED for triggering the IoT data task offloading, (2) A new block is created to represent the verified transaction, (3) A mined block is appended to the blockchain.

transportation and healthcare monitoring, and generating data which need to be computed to serve end users. IoT sensors can act as lightweight blockchain nodes to securely communicate and transmit data to their nearby ED via the blockchain network.

- **Edge Devices:** Each ED such as a laptop or a powerful smartphone manages a group of IoT sensors under its coverage. Based on the QoE requirements, EDs can use their computational capability to process data tasks locally or offload to a nearby MEC server via wireless links. EDs also participate in block mining, i.e., transaction verification and block generation, via a PoR consensus mechanism where IoT sensor users vote to select representative EDs to run the mining process. The details of our blockchain mining design are explained in Section 3.
- **MEC Server:** In our considered blockchain-based MEC system, there is a single MEC server located at a BS to handle computationally extensive data tasks offloaded from EDs. By analyzing the task profile such as task sizes, channel conditions, and available resource, EDs can make offloading decisions so that the MEC server can allocate its resources for computation under QoE requirements.
- **Blockchain:** A blockchain network is deployed over the MEC system where each ED acts as a blockchain miner. In this paper, we propose a PoR framework and focus on analyzing the block verification latency that is a significant factor in evaluating the efficiency of a blockchain network. The use of our proposed PoR scheme allows EDs to participate in the block mining with an enhanced mining utility which contributes to the system utility improvement in the blockchain-based MEC system.

The blockchain framework for our MEC network is illustrated in Fig. 2, and its operational concept is explained via the following steps:

- **Step 1:** The IoT sensor user first creates a transaction with metadata (i.e., user ID), user signature and timestamp from its wallet account and sends it to the associated ED. The user then submits a transaction to the ED for a certain request, such as IoT data task offloading.

- *Step 2:* The ED releases its available resources and processes the request from the user. For example, an ED can use a smart contract [37], a self-executing software running on blockchain, to automatically perform transaction authentication, user verification, or resource trading. Moreover, the ED collaborates with other mining members to aggregate the transactions offloaded from IoT users to build a block after a certain period of time. Then, the EDs participate in the mining to verify the block using a consensus mechanism, e.g., PoR.
- *Step 3:* After the mining, if all miners achieve an agreement on the verified block, this block with its signature is then appended to the chain of blocks in chronological order. Finally, all network entities receive this block and synchronize the copy of the blockchain.

## 2.2 Edge Task Offloading Model

Here, we present the communication model and the computing model for the edge task offloading.

### 2.2.1 Communication Model

We denote  $\mathcal{K} = \{1, \dots, K\}$  as the set of available sub-bands at the BS. We define a task offloading policy, which also incorporates the uplink sub-band scheduling, by a binary variable  $x_n^k$ , ( $n \in \mathcal{N}, k \in \mathcal{K}$ ). Here,  $x_n^k = 1$  indicates that the task  $Y_n$  from ED  $n$  is offloaded to the MEC server via sub-band  $k$ , and  $x_n^k = 0$  otherwise. Each computation task can be either executed locally at the ED or offloaded to the MEC server under a feasible offloading policy

$$\sum_{k \in \mathcal{K}} x_n^k \leq 1, n \in \mathcal{N}. \quad (1)$$

In line of the above discussion, we define the task offloading policy  $X$  that contains all the task offloading variables  $x_n^k$  as  $X = \{x_n^k | x_n^k = 1, n \in \mathcal{N}, k \in \mathcal{K}\}$ . Besides, we denote  $\mathcal{N}_n = \{n \in \mathcal{N} | \sum_{k \in \mathcal{K}} x_n^k = 1\}$  as the set of EDs offloading their tasks to the MEC server.

Moreover, we consider that each ED and the BS have a single antenna for uplink communications. We denote  $h_n^k$  as the uplink channel gain between the ED  $n$  and the BS on sub-band  $k$ . Let  $P = \{p_n^k | 0 < p_n^k \leq P_n^k, n \in \mathcal{N}_n\}$  denote the transmit power policy of EDs, where  $p_n^k$  is the transmit power of ED  $n$  when offloading the task  $Y_n$  to the BS via the channel  $k$ , subject to a maximum budget  $P_n^k$ . We also assume that the MEC system has a total operational frequency band  $B_{MEC}$  that is divided into  $K$  sub-bands of an equal size  $W = B_{MEC}/K$  [Hz]. Then, the transmission data rate of the ED  $n$  can be calculated as

$$R_n = W \log_2 \left( 1 + \frac{p_n^k h_n^k}{\sigma^2 + \sum_{j \in \mathcal{N}_n, j \neq n} (x_j^k p_j^k h_j^k)} \right), \quad (2)$$

where  $\sigma^2$  is the background noise variance and the second term at the denominator is the interference among mobile users in the same channel. We also denote  $x_n = \sum_{k \in \mathcal{K}} x_n^k, \forall n \in \mathcal{N}$ . Thus, the required time that the ED  $n$  upload its task input  $D_n$  via the uplink is specified as

$$T_n^{up} = \frac{D_n}{R_n}, \forall n \in \mathcal{N}. \quad (3)$$

Accordingly, the energy consumption of ED  $n$  for offloading the task  $Y_n$  is specified as

$$E_n^{up} = p_n T_n^{up} = p_n \frac{D_n}{R_n}, \forall n \in \mathcal{N}, \quad (4)$$

where  $p_n = \sum_{k \in \mathcal{K}} p_n^k, \forall n \in \mathcal{N}$ .

### 2.2.2 Computing Model

We consider two computing modes: local execution and edge offloading.

- *Local execution:* Let  $f_n^l$  denote the computational resource of ED  $n$  (in CPU cycles/s) allocated to execute the data task, which should not exceed its total computation capacity  $F_n$ . Thus, we can define the policy of computational resource allocation of EDs as  $F = \{f_n^l | 0 < f_n^l \leq F_n, n \in \mathcal{N}\}$ . The time consumed to execute the task input  $D_n$  (with  $C_n$  in CPU cycles) at an ED  $n$  is expressed as

$$T_n^l = \frac{C_n}{f_n^l}. \quad (5)$$

Moreover, the energy consumption of an ED  $n$  when executing its task locally is specified as

$$E_n^l = \kappa (f_n^l)^2 C_n, \quad (6)$$

where  $\kappa$  is the energy coefficient depending on the chip architecture [9] and  $C_n$  is the CPU workload of ED  $n$ .

- *Edge offloading:* For the offloading case, the MEC server at the BS can provide computation services to multiple EDs concurrently. Compared to the local device, the MEC server has much more powerful computation capacity  $f^e$  (in CPU cycles/s) and more stable power supply. The execution time of task  $Y_n$  at the MEC server can be calculated as

$$T_n^{ex} = \frac{C_n}{f^e}, \forall n \in \mathcal{N}. \quad (7)$$

Similar to [9], [10], we do not model the downloading part due to the small size of data results compared to the offloading data.

In summary, the latency cost consumed by the ED  $n$  when offloading its task  $Y_n$  is given by

$$T_n^{off} = T_n^{up} + T_n^{ex} = \left( \frac{D_n}{R_n} + \frac{C_n}{f^e} \right), \forall n \in \mathcal{N}. \quad (8)$$

Moreover, the energy cost consumed by the ED  $n$  when offloading its task  $Y_n$  is only associated with the data transmission, which is given by

$$E_n^{off} = E_n^{up} = p_n \frac{D_n}{R_n}, \forall n \in \mathcal{N}. \quad (9)$$

## 3 BLOCKCHAIN CONSENSUS DESIGN

In the blockchain-based MEC system, a crucial component is blockchain consensus that aims to mine the blocks of transactions (i.e., IoT data records) and add them to the blockchain. To handle the transactions, the EDs work as



blockchain miners to perform mining. In the blockchain-based edge offloading environment, latency is one of the most important factors determining the efficiency of a blockchain system. Given a consensus algorithm, when the number of transactions to the blockchain increases, the consensus workload to validate and append them into the blockchain will increase significantly. In current consensus schemes, e.g., DPoS [38], each miner node must implement a repeated verification process across the miner network, which results in unnecessary consensus latency and network bandwidth waste. A possible solution is to reduce the number of miner nodes to reduce the consensus latency, but it potentially compromises the security of blockchain because of the high probability of adding compromised transactions from malicious nodes [39]. To solve these mining issues, here we propose a new lightweight Proof of Reputation (PoR) consensus mechanism for our blockchain system. Compared to the DPoS scheme, we make an improvement in the miner selection based on a reputation score evaluation approach. Moreover, instead of using a repeated verification among miner nodes, we implement a lightweight block verification solution that allows each miner only needs to verify once with another node during the consensus process, which would significantly reduce the verification latency and save network bandwidth. There are two main parts of our PoR consensus, including miner node selection and block verification, as illustrated in Fig. 3.

### 3.1 Miner Node Selection

In this phase, the IoT users first calculate the reputation score of EDs and then select the miner nodes to implement the mining process.

#### 3.1.1 Reputation Calculation

In our MEC system, IoT sensors' users participate in the delegate selection process to vote the mining candidates among EDs for performing blockchain consensus. In this regard, each IoT user votes its preferred ED with the most reputation. Here, the reputation of an ED is measured by its mining utility with respect to mining latency. That is, an ED exhibits a lower mining latency will have a better mining utility which increases its reputation. To this end, we define a mining utility function of each ED as

$$J_n^{\text{mine}} = \left[ e^{1 - \frac{T_n^{\text{PoR}}}{\tau_n}} - 1 \right]^+, \quad (10)$$

where  $T_n^{\text{PoR}}$  is the mining latency of the ED  $n$  (its detail will be explained in the following sub-section),  $\tau_n$  denotes the task execution latency constraint.  $[y]^+ = \max\{y, 0\}$  implies that the reputation of an ED is set to 0 if the mining latency  $T_n^{\text{PoR}}$  is exceeded to its task execution constraint  $\tau_n$ .

#### 3.1.2 Miner Selection

Based on the calculated reputation score, each sensor user will vote for the ED candidates as the miners based on their reputation ranking. The top EDs with highest reputation scores are selected to become edge miners (EMs) to perform mining, as indicated in Fig. 3. Also, similar to the traditional DPoS framework [38], in our PoR mechanism, each of the active EMs takes turn to act as a block manager during its time slot

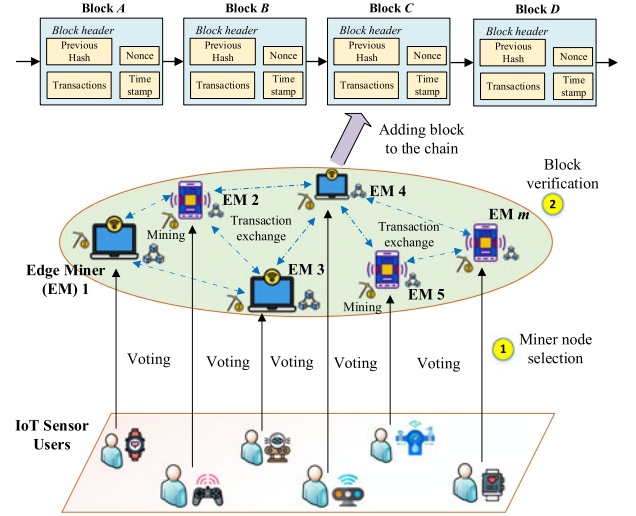


Fig. 3. The proposed PoR consensus in our blockchain-based MEC system.

to coordinate the consensus process. In other words, there is one manager in each consensus process. In the next time slot, another active EM will undertake this manager role.

### 3.2 Lightweight Block Verification

In this phase, the block manager first produces an unverified block  $B$  that contains several offloading transactions collected in a given amount of time. Then, the manager broadcasts this created block to all EMs within the miner network for verification. Different from the traditional DPoS scheme [38] which relies on a repeated verification process among miners, here we implement a lightweight verification solution that allows each miner to only verify once with another node during the consensus process. Algorithm 1 presents how our proposed block verification procedure is performed. In lines 4 to 9, the block manager divides the block  $B$  consisting of transactions into  $N$  transaction parts  $Tr_n$ ,  $n \in \mathcal{N}$  that will be assigned to each mining member  $EM$  within the miner group. Each miner  $EM$  will also be assigned a unique random number  $R_n$ . In lines 11 to 21, an  $EM$  selects any miner  $s$  ( $s \in \mathcal{N} \setminus n$ ) within the miner group to implement the verification for its assigned transaction part  $Tr_n$ . If 51% of the EMs respond positive verification, and the sum of random numbers  $Sum$  calculated by all EMs is equal to the initial number set  $Rnd$ , the block manager accepts the verified block  $B'$  and adds it to the blockchain with a signature. For instance, in Fig. 3, the EM 4 works as a manager to create the block  $C$  and append it into the blockchain. Otherwise, the manager discards it from the network (in lines 22 to 27).

### 3.3 Latency of Block Verification

In this sub-section, we calculate the verification latency incurred by the mining. For simplicity, we assume that the transaction part  $Tr_n$  (which also expresses the size) is the same for all EMs. Each EM is willing to allocate a certain CPU resource  $\phi_n$  (in CPU cycles) for the verification of transaction part  $n$ . Then, the CPU resource allocation policy for block verification of EDs can be defined as  $G = \{\phi_n | 0 < \phi_n \leq \Phi_n, n \in \mathcal{N}\}$ , where  $\Phi_n$  is the CPU resource budget of the ED  $n$ . Further, the size of verified transaction result for

the  $Tr_n$  is denoted by  $Tr_n^{re}$ . Hence, the transaction verification task can be expressed as a tuple  $(Tr_n, \phi_n, Tr_n^{re})$ .

---

**Algorithm 1.** Procedure of the Proposed PoR Consensus
 

---

```

1: Input: the unverified block  $B$ , a set of EMs  $\mathcal{N}$ 
2: Output: the verified block  $B'$ 
3: Initialization: Select an unverified block  $B$ , group the
   selected EMs  $EM$  in the list  $Array[n]$ ,  $n \in \mathcal{N}$ , initiate public
   key  $Array[n].PK$  and block manager  $BM$ 
4: Divide the block  $B$  into  $N$  parts  $Tr_n$ ,  $Sum \leftarrow 0$ 
5: for  $n = 1, \dots, N$  do
6:   Set a part of block  $Tr_n \rightarrow Array[n].content$ 
7:   Assign a random number  $R_n \leftarrow Random()$ 
8:   Calculate a signature as  $Sig_n \leftarrow Hash(Array[n].content,$ 
      $Array[n].PK, timestamp)$ 
9: end for
10: Specify the total random number  $Rnd = N(1 + \frac{N-1}{2})$ 
11: for  $n = 1, \dots, N$  do
12:   Run a random function  $s = Random.randrange(1, N, 1)$ 
13:   if  $s \neq n$  then
14:     Select a random different EM within the list
15:     Send the  $Tr_n$  to  $EM_s$ :  $EM \rightarrow EM_s : (Tr_n, Array[n].PK,$ 
        $Sig_n, timestamp)$ 
16:     Verify the transaction  $Tr_n$ 
17:     if  $(Array[n].PK_{EM_s} == EM_s^{EM.PK}) \cap (Verify(Sig_n) \leftarrow$ 
        $true)$  then
18:        $Sum \leftarrow Sum + R_n$ 
19:     end if
20:   end if
21: end for
22: if  $Sum == Rnd$  then
23:   Accept the block  $B$  as a verified one ( $B'$ ) and send it back
   to the block manager  $BM$ 
24:   The manager  $BM$  appends the verified block  $B'$  into the
   blockchain network:  $BM \rightarrow * : (BM_{PK}, B', Sig_n,$ 
      $timestamp)$ 
25: else
26:   Discard the block  $B$  from the blockchain
27: end if
    
```

---

Conceptually, the block verification process in our proposed PoR mechanism at an EM experiences four steps: (1) unverified block transmission from the block manager to the EMs, (2) local block verification at the EM, (3) broadcasting of the verification result among two EMs, and (4) transmission of verification result feedback from the EMs to the manager. For a miner EM  $n$ , the time required to complete these steps is expressed as

$$T_n^{PoR} = \frac{Tr_n}{r_n^d} + \frac{\phi_n}{F_n} + \xi Tr_n |L^2| + \frac{Tr_n^{re}}{r_n^u}, n \in \mathcal{N}, \quad (11)$$

where  $r_n^u$  and  $r_n^d$  are uplink and downlink transmission rates between the miner  $n$  and the block manager. Here, the transmission time of an unverified transaction part  $Tr_n$  from the block manager to the miner is  $\frac{Tr_n}{r_n^d}$ , while the local verification time of this transaction is  $\frac{\phi_n}{F_n}$ . Moreover, similar to [4], the time for transaction broadcasting among two miners is a function of transaction size  $Tr_n$  and network scale  $Tr_n |L^2|$  (which means two miners for transaction verification), which is defined as  $\xi Tr_n |M^2|$ . Here,  $\xi$  is a pre-defined parameter of broadcasting verification result and comparison among two

miners, which can be acquired from the previous verification records [4]. Besides,  $\frac{Tr_n^{re}}{r_n^u}$  is the verification feedback time

Meanwhile, in the traditional DPoS scheme [38], each miner has to implement a repeated verification process among all miners for the block  $B$ , instead of dividing into separate transaction parts like our proposed PoR model. Therefore, the verification latency of the DPoS consensus at an EM  $n$  is expressed as [4]

$$T_n^{DPoS} = \frac{B}{r_n^d} + \frac{\phi_n^B}{c_n^B} + \xi B |L^N| + \frac{B^{re}}{r_n^u}, \quad (12)$$

where  $\phi_n^B$  is the CPU resource occupied to verify the block  $B$  under the computation budget  $c_n^B$ .  $B^{re}$  denotes the size of verified result of the block  $B$ .  $|L^N|$  expresses the whole miner network which means all miners  $n$  join the repeated block verification in each consensus process, instead of two-miner verification in our PoR scheme. By comparison of Equations (11) and (12), it can be seen that the proposed PoR scheme needs less time for block verification in comparison with the traditional DPoS scheme, for the same block size and number of miners. Moreover, our mining scheme can save much network bandwidth due to less message exchange during the consensus process. The benefits of our proposed PoR mechanism are verified in the following sections.

## 4 SYSTEM UTILITY FORMULATION AND PROPOSED MA-DRL ALGORITHM

In this section, we present the system utility formulation for our proposed TOBM scheme based on the joint consideration of offloading utility and mining utility as presented in the previous sections. Then, we derive the system utility optimization problem as a cooperative game and propose a new MA-DRL algorithm to solve it.

### 4.1 System Utility Formulation

In this paper, we formulate the system utility for the TOBM scheme by taking both offloading utility and mining utility into account.

#### 4.1.1 Offloading Utility

Here, we focus on formulating the QoE-aware offloading utility function. In an MEC system, the offloading's QoE is mainly characterized by their task computation time, i.e.,  $T_n$  and energy consumption, i.e.,  $E_n$ . Specifically,  $T_n$  and  $E_n$  can be specified as

$$T_n = T_n^{off} x_n + T_n^l (1 - x_n), \forall n \in \mathcal{N}, \quad (13)$$

$$E_n = E_n^{off} x_n + E_n^l (1 - x_n), \forall n \in \mathcal{N}. \quad (14)$$

Accordingly, we define a QoE-aware utility function to measure the offloading utility that is specified as a trade-off between the time and energy consumption of the task compared with local execution

$$J_n^{off} = \lambda_n^T \left( \frac{T_n^l - T_n}{T_n^l} \right) + \lambda_n^E \left( \frac{E_n^l - E_n}{E_n^l} \right), \quad (15)$$

where  $\lambda_n^T, \lambda_n^E \in [0, 1]$  (with  $\lambda_n^T + \lambda_n^E = 1, \forall n \in \mathcal{N}$ ) are set by ED  $n$  to show the preference on time and energy cost when computing the task  $Y_n$ . If the task is emergency, the ED can increase the weighting factor of time consumption. Meanwhile, if the ED is operating with low battery, the factor of energy consumption should be preferred.

It is noting that here, the offloading utility function  $J_n^{off}$  reflects the improvement in QoE over local execution, that is measured by  $\left(\frac{T_n^l - T_n}{T_n}\right)$  and  $\left(\frac{E_n^l - E_n}{E_n}\right)$ , respectively. Specifically, when the ED  $n$  executes the task locally, the offloading utility equals 0 (i.e.,  $J_n^{off} = 0$ ). If the computation cost (i.e., latency and energy consumption) of the offloading mode is lower than that of the local execution mode, the offloading utility ( $J_n^{off}$ ) can be positive, which indicates the offloading's QoE improvement. However, if offloading too many tasks, the EDs can suffer from higher latency due to the traffic congestion, which would reduce the QoE. As a result, the offloading utility ( $J_n^{off}$ ) can be negative.

To this end, we formulate the QoE-aware offloading utility for the MEC system. Given the offloading decision policy  $X$ , the transmission power policy  $P$ , and the computational resource allocation policy  $F$ , we define the offloading utility as the weighted sum of all MDs' offloading utilities  $J_n^{off}$ , denoted as

$$J^{off} = \sum_{n \in \mathcal{N}} J_n^{off}(X, P, F). \quad (16)$$

#### 4.1.2 Mining Utility

We adopt the mining utility built in Equation (10) to analyse the efficiency of the mining in the blockchain-enabled task offloading. Given the CPU resource allocation policy  $G$ , each MEC server yields an utility  $J_n^{mine}$  when performing the mining process. Then, we can specify the total mining utility of EDs in the MEC system as

$$J^{mine} = \sum_{n \in \mathcal{N}} J_n^{mine}(G). \quad (17)$$

Accordingly, the system utility of each MD  $J_n$  can be expressed as

$$J_n = J_n^{off} + J_n^{mine}. \quad (18)$$

#### 4.1.3 System Utility Formulation

In this paper, our objective is maximize the total system utility as the sum of the offloading utility and the mining utility for the proposed TOBM scheme

$$\underset{X, P, F, G}{\text{maximize}} \quad J^{off} + J^{mine} \quad (19a)$$

$$\text{subject to} \quad x_n^k \in \{0, 1\}, \forall n \in \mathcal{N}, k \in \mathcal{K}, \quad (19b)$$

$$\sum_{k \in \mathcal{K}} x_n^k \leq 1, n \in \mathcal{N}, \quad (19c)$$

$$0 < p_n^k \leq P_n^k, \forall n \in \mathcal{N}_n, k \in \mathcal{K}, \quad (19d)$$

$$0 < f_n^l \leq F_n, \forall n \in \mathcal{N}, \quad (19e)$$

$$0 < \phi_n \leq \Phi_n, \forall n \in \mathcal{N}_n \quad (19f)$$

$$T_n \leq \tau_n, \forall n \in \mathcal{N}. \quad (19g)$$

Here, the constraints (19b) and (19c) imply that each task can be either executed locally or offloaded to the MEC server via a sub-channel. (19d) shows the transmission power constraint of each ED. The constraint (19e) states that each ED  $n$  must allocate a positive computational resource to execute the computing task, but not exceed the total computation budget  $F_n$ . Each ED  $n$  also must allocate a positive CPU resource for the block verification under a maximum CPU capability  $\Phi_n$ , as indicated in (19f). Constraint (19g) ensures that each data task needs to be completed under a delay threshold.

The key intuition behind this integrated calculation is that in the blockchain-based MEC system, each ED needs to simultaneously perform task offloading and block mining. In this context, the evaluation of the system quality, e.g., overall utility performance, must consider both offloading utility and mining utility. Indeed, an ED needs to minimize its offloading latency and energy consumption to maintain the quality of task offloading service, while also minimizing its mining latency to maintain the quality of block mining service. Therefore, we come up with a final solution to satisfy both services, aiming to optimize the overall system utility performance.

The problem (19) is non-convex and centralized. As a dynamic TOBM problem (due to varying channel conditions and task sizes) and high-dimensional system state space (due to the increase of EDs), the use of traditional optimization approaches results in high computational complexity which would hinder the applicability of the proposed model in practical blockchain-based MEC scenarios. Moreover, most of current solutions [13], [14], [32] rely on single-agent learning which suffers from some critical shortcomings:

- *Dimensionality*: The cardinalities of DNN input and output are generally proportional to the number of EDs, and thus the use of centralized learning to obtain the optimal policy for all EDs is challenging. Moreover, exploration in high-dimensional state space is inefficient especially when the number of EDs increases exponentially, which makes the learning in the blockchain-based MEC system impractical.
- *Information transmission*: Since the centralized learning always requires full information of all EDs (e.g., data task state, resource state, block size state) for decision making, the information transmission becomes challenging with the increase of EDs.

Due to the powerlessness of centralized learning algorithms in the multi-agent environment like our considered blockchain-based MEC system, we propose to use a distributed MA-DRL scheme to solve our cooperative TOBM problem, as presented in the following sub-sections.

## 4.2 Cooperative Learning Formulation

First, we convert the objective function (19a) from a system utility maximization problem to a reward maximization problem. To do this, we formulate the task offloading problem using a multi-agent version of MDP, also known as a *Markov game* which is denoted by a tuple  $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}, \mathcal{O} \rangle$ . Here, each ED  $n$  is considered as an intelligent agent to



learn its optimal policy by observing the TOBM environment and collaborating with other agents, aiming to achieve the optimal system utility. Then, we have  $\mathcal{N} = \{1, 2, \dots, N\}$  as the set of EDs (or agents). Moreover,  $\mathcal{S} = \{s_1, s_2, \dots, s_N\}$  is defined as the set of states,  $\mathcal{A} = \{a_1, a_2, \dots, a_N\}$  is a set of agent actions, and  $\mathcal{O} = \{o_1, o_2, \dots, o_N\}$  denotes a set of observations for agents. We assume that the considered cooperative TOBM scheme operates on discrete time horizon with each time slot  $t$  equal and non-overlapping, and the communication parameters keep unchanged during each time slot. Now we define each item in the tuple at each time slot  $t$  as follows.

#### 4.2.1 State

The environment states in the cooperative TOBM network include five components: task state  $S_{task}(t)$ , channel state  $S_{channel}(t)$ , power state  $S_{power}(t)$ , resource state  $S_{res}(t)$ , and transaction state  $S_{trans}(t)$ . Therefore, the system state is defined as a matrix

$$\mathcal{S}(t) = \{S_{task}(t), S_{channel}(t), S_{power}(t), S_{res}(t), S_{trans}(t)\}, \quad (20)$$

where each state vector is explained as follows.  $S_{task}(t)$  is defined as  $S_{task}(t) = [D_n(t), C_n(t)]$ ,  $n \in \mathcal{N}$  where  $D_n(t)$  represents the computation task size of the ED  $n$  and  $C_n(t)$  is the required input CPU cycles number to complete the task data  $D_n(t)$ .  $S_{channel}(t)$  is defined as

$$S_{channel}(t) = c_n^k(t) = \begin{bmatrix} c_{1,1} & \cdots & c_{1,K} \\ \vdots & \ddots & \vdots \\ c_{N,1} & \cdots & c_{N,K} \end{bmatrix}, \quad (21)$$

where  $c_n^k(t)$  indicates whether the sub-channel  $k$  is used by ED  $n$  at time slot  $t$ . If yes,  $c_n^k(t) = 1$ , otherwise  $c_n^k(t) = 0$ . Also,  $S_{power}(t)$  is defined as

$$S_{power}(t) = p_n^k(t) = \begin{bmatrix} p_{1,1} & \cdots & p_{1,K} \\ \vdots & \ddots & \vdots \\ p_{N,1} & \cdots & p_{N,K} \end{bmatrix}, \quad (22)$$

where  $p_n^k(t)$  represents the ED  $n$ 's transmit power level in the  $k$ th sub-channel, which is a continuous variable and satisfies  $0 < p_n^k(t) \leq P_n^k$ . Moreover, the resource state  $S_{res}(t)$  is expressed as

$$S_{res}(t) = \{v_1(t), v_2(t), \dots, v_N(t)\}, \quad (23)$$

where  $v_n(t)$  contains the states of current available computational resource  $f_n^l(t)$  and CPU resource  $\phi_n(t)$  of the ED  $n$ . Lastly, the transaction state  $S_{trans}(t)$  is defined as

$$S_{trans}(t) = \{Tr_1(t), Tr_2(t), \dots, Tr_N(t)\}, \quad (24)$$

where  $Tr_n(t)$  is the transaction state of ED  $n$ .

#### 4.2.2 Action

By observing the system states, each ED needs to make actions in each time step to deal with the task execution and block mining, including offloading decision, channel selection, transmit power selection, computational resource

allocation, and CPU resource allocation. Accordingly, the action space can be expressed as

$$\mathcal{A}(t) = \{x_n^k(t), k(t), p_n^k(t), f_n^l(t), \phi_n(t)\}, \quad (25)$$

where each action compoment is explained as follows:

- Offloading decision  $x_n^k(t)$ :  $x_n^k(t) \in \{0, 1\}$ , ( $n \in \mathcal{N}$ ,  $k \in \mathcal{K}$ ). Each ED  $n$  makes decision to execute the task locally  $x_n^k(t) = 0$  or offload it to the MEC server  $x_n^k(t) = 1$  via the channel  $k$ , based on the current task state  $S_{task}(t)$ .
- Channel selection  $k(t)$ :  $k(t) = [1, 2, \dots, K]$ . Each ED  $n$  selects one of the available channels to offload the task to the MEC server, based on the current channel state  $S_{channel}(t)$ .
- Transmit power selection  $p_n^k(t)$ :  $p_n^k(t) \in (0, P_n^k]$ , ( $n \in \mathcal{N}$ ,  $k \in \mathcal{K}$ ). Each ED  $n$  chooses a transmit power value to transmit the data task to the MEC server with respect to the current task state  $S_{task}(t)$  and channel state  $S_{channel}(t)$ .
- Computational resource allocation  $f_n^l(t)$ :  $f_n^l(t) = [f_1^l(t), f_2^l(t), \dots, f_N^l(t)]$ . Each ED  $n$  allocates part of its computational resource to execute the task with respect to the current resource state  $S_{res}(t)$  and task state  $S_{task}(t)$ .
- CPU resource allocation  $\phi_n(t)$ :  $\phi_n(t) = [\phi_1(t), \phi_2(t), \dots, \phi_N(t)]$ . Each ED  $n$  allocates part of its CPU resource to verify the blockchain transaction, based on the current resource state  $S_{res}(t)$  and transaction state  $S_{trans}(t)$ .

#### 4.2.3 System Reward Function

The system reward at one time slot  $t$  is the sum of the rewards of all EDs. Each ED  $n$  will get a reward  $r(s_n(t), a_n(t))$  in a certain state  $s_n(t)$  after executing each possible action  $a_n(t)$ . In our paper, the system reward function should be positively correlated to the objective function in the optimization problem (19), aiming to maximize the system utility of all EDs. Then, we can specify the system reward function of our offloading network at each time slot  $t$  as

$$r(s(t), a(t)) = \sum_{n \in \mathcal{N}} r(s_n(t), a_n(t)) = J(t), \quad (26)$$

where  $J(t) = J^{off}(t) + J^{mine}(t)$  is the total system utility of the blockchain-based MEC system.

### 4.3 Proposed MA-DRL Algorithm for Cooperative TOBM

In the cooperative TOBM problem in our blockchain-based MEC system, conventional single-agent [13], [14], [32] or independent multi-agent [17], [18], [19] solutions are unable to obtain the cooperative policies of EDs due to the nonstationary and partially observable environment. Indeed, when policies of other agents change (i.e., due to computation mode preference), the ED (agent)  $n$  observation  $O_n$  can be changed (nonstationary), which makes the obtained reward  $r_n(t)$  different from the accumulated reward from its actual state-action pair. Moreover, in independent multi-agent learning schemes, the agent  $n$  only has the local information and cannot know the updates from other agents due



resource. Therefore, we mainly focus on the computational complexity of the execution process at EDs. Here, a DRL agent with a DNN is established for each ED and all DNNs run in parallel across the MEC network. As a result, the overall complexity of the multi-agent system can be determined by the complexity of a single DNN at an ED. We assume that are  $K$  neurons at the input layer of the DNN for each ED, and  $Z$  as the number of neurons at the output layer. Also, the hidden layer is  $L$ , and the number of neurons at hidden layers is  $H$ . Accordingly, the computation cost at a DNN is  $(KH + (L - 1)HH + HZ) = \mathcal{O}(H(K + (L - 1)H + Z))$ . Also, the complexity of using activation function is  $\mathcal{O}(HL)$ . Hence, the total complexity is  $\mathcal{O}(H(K + HL - H + Z + L))$  which can be simplified as  $\mathcal{O}(H(K + HL + Z))$ .

#### 4.4 Cooperative Game-Theoretic Solution

We next develop a game-theoretic approach for the proposed TOMB problem, where each ED can act as a game player to react to other players' decisions for maximizing its utility [41], [42]. After a number of steps, all the EDs self-organize into a mutual equilibrium state, i.e., the Nash equilibrium, at which no ED can further increase its utility by unilaterally altering its strategy.

---

#### Algorithm 2. The MA-DDPG Training Procedure in the Blockchain-Based MEC System

---

- 1: **Input:** Replay memory  $\mathcal{D}$ , time budget  $T$ , exploration probability  $\epsilon$ , discount factor  $\gamma$ , update step  $\zeta$
  - 2: **Output:** The optimal policy  $\pi_{\theta_n}^*$  and maximum reward  $r^*(s, a)$
  - 3: **Initialization:** Initialize the deep Q network  $Q(s, a)$  with random weight  $\theta$  and  $\theta'$ , initialize the exploration probability  $\epsilon \in (0, 1)$
  - 4: **for** episode = 1,...,  $M$  **do**
  - 5:   Initialize the state  $s_0 \leftarrow \{S_{task}(t), S_{channel}(t), S_{power}(t)\}_{t=0}$
  - 6:   **for**  $t = 1, 2, \dots, T$  **do**
  - 7:     For each agent ED  $j \in \mathcal{N}$ , select a random action  $a_j(t)$  with probability  $\epsilon$ , otherwise  $a_j(t) = \pi_{\theta_j}(s_j(t))$
  - 8:     Execute actions  $a(t) = (a_1(t), a_2(t), \dots, a_N(t))$  by performing offloading decision  $x_n^k(t)$ , channel selection  $k(t)$ , transmit power selection  $p_n^k(t)$ , computational resource allocation  $f_n^l(t)$ , and CPU resource allocation  $\phi_n(t)$
  - 9:     Observe the system reward  $r(t)$  via (26) and the new state  $s'$
  - 10:    Store  $(s(t), a(t), r(t), s'(t))$  into the memory  $\mathcal{D}$
  - 11:    **for** agent  $j = 1$  to  $N$  **do**
  - 12:     Sample random mini-batch of transitions  $(s_j, a_j, r_j, s'_j)$  from  $\mathcal{D}$
  - 13:     Set  $y_j = r_j + \gamma Q_j^\pi(s'_j, a'_1, a'_2, \dots, a'_N) |_{a'_j = \pi'_j(a_j)}$
  - 14:     Update behavior critic by minimizing the loss:  $L(\theta_j) = \frac{1}{S} \sum_j [y_j - Q_j^\pi(s_j, a_1, a_2, \dots, a_N)]^2$
  - 15:     Update actor by using the sampled policy gradient:  $\nabla_{\theta_j} J(\pi_j) = \frac{1}{S} [\nabla_{\theta_j} Q_j^\pi(s'_j, a_1, a_2, \dots, a_N) \cdot \nabla_{\theta_j} \pi_j(a_j | s_j)]$
  - 16:    **end for**
  - 17:    Update the target network parameters for each agent via (30)
  - 18:   **end for**
  - 19: **end for**
- 

We first define the game as  $G = \{\mathcal{N}, \{\mathcal{A}_n\}_{n \in \mathcal{N}}, \{J_n\}_{n \in \mathcal{N}}\}$ , where  $\mathcal{N}$  is the set of rational game players,  $\mathcal{A}_n$  is the strategy set for player  $n$ , and  $J_n$  is the utility function of ED  $n$ . Let denote  $a_n$  as the offloading decision profile of the player  $n$  over the wireless sub-bands  $\mathcal{K}$ , from (18) we can rewrite as

$$J_n(a_n) = J_n^{off} I_{\{a_n=1\}} + J_n^{mine}, \quad (31)$$

where  $I_z$  is an indicator function. If  $z$  is true,  $I_z = 1$ ; otherwise,  $I_z = 0$ . Based on (18) and (31), the utility of a player  $n$  in the cooperative game can be expressed as

$$J(a_n, a_{-n}) = \begin{cases} J_n^{mine}, & a_n = 0 \\ J_n^{off} + J_n^{mine}, & a_n = 1 \end{cases}, \quad (32)$$

where  $a_{-n}$  is the offloading decisions of all players except  $n$ . Accordingly, by considering the influence of other players on the utility optimization of a player  $n$  in the cooperative game, we can express the game-theoretic utility function as

$$J(a_n, a_{-n}) = \begin{cases} J_n^{mine}, & a_n = 0 \\ \sum_{m \neq n} (J_n^{off}(a_n) + J_m^{off}(a_m)) I_{\{a_m=a_n\}} + J_n^{mine}(a_n), & a_n = 1 \end{cases} \quad (33)$$

where  $m \in \mathcal{N} \setminus \{n\}$ .

**Theorem 1:** The collaborative game  $G = \{\mathcal{N}, \{\mathcal{A}_n\}_{n \in \mathcal{N}}, \{J_n\}_{n \in \mathcal{N}}\}$  has a pure Nash equilibrium (NE) and guarantees the finite improvement property.

*Proof:* We first prove that the proposed game  $G$  is an exact (cardinal) potential game with potential function

$$\begin{aligned} \Psi(a_n, a_{-n}) = & \frac{1}{2} \sum_{n \in \mathcal{N}} \sum_{m \neq n} (J_n^{off} + J_m^{off}) I_{\{a_m=a_n\}} I_{\{a_n=1\}} \\ & + \sum_{n \in \mathcal{N}} J_n^{mine} I_{\{a_n=1\}} + \sum_{n \in \mathcal{N}} J_n^{mine} I_{\{a_n=0\}}, \end{aligned} \quad (34)$$

such that

$$\begin{aligned} \Psi(a'_n, a_{-n}) - \Psi(a_n, a_{-n}) = & J(a'_n, a_{-n}) - J(a_n, a_{-n}), \\ & \forall a_n, a_{-n} \in \mathcal{A}_n. \end{aligned} \quad (35)$$

We now consider three cases as follows:

Case 1:  $a_n > 0, a'_n > 0$

$$\begin{aligned} \Psi(a'_n, a_{-n}) - \Psi(a_n, a_{-n}) = & \frac{1}{2} \sum_{m \neq n} (J_n^{off}(a'_n) - J_n^{off}(a_n)) I_{\{a_m=a'_n\}} \\ & + \frac{1}{2} \sum_{m \neq n} (J_n^{off}(a_m) - J_n^{off}(a'_n)) I_{\{a'_n=a_m\}} + J_n^{mine}(a'_n) \\ & - \frac{1}{2} \sum_{m \neq n} (J_n^{off}(a_n) - J_n^{off}(a_m)) I_{\{a_m=a_n\}} \\ & - \frac{1}{2} \sum_{m \neq n} (J_n^{off}(a_m) - J_n^{off}(a_n)) I_{\{a_n=a_m\}} + J_n^{mine}(a_n) - J_n^{mine}(a_n) \\ = & \sum_{m \neq n} (J_n^{off}(a'_n) + J_m^{off}(a_m)) I_{\{a_m=a'_n\}} + J_n^{mine}(a'_n) \\ & - \sum_{m \neq n} (J_n^{off}(a_n) + J_m^{off}(a_m)) I_{\{a_m=a_n\}} - J_n^{mine}(a_n) \\ = & J(a'_n, a_{-n}) - J(a_n, a_{-n}). \end{aligned} \quad (36)$$

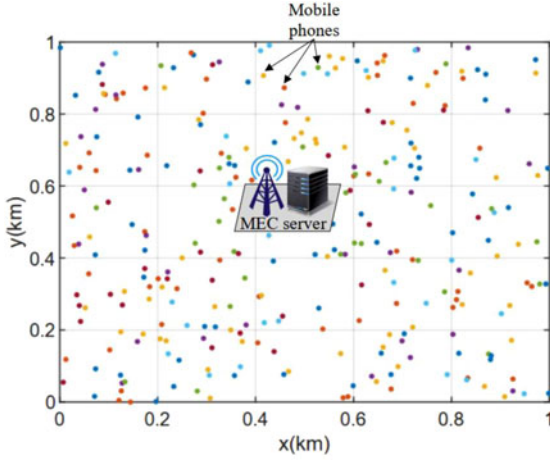


Fig. 5. Illustration of the MEC network with an MEC server and distributed mobile phones.

Case 2:  $a_n = 0, a'_n > 0$

$$\begin{aligned}
 & \Psi(a'_n, a_{-n}) - \Psi(a_n, a_{-n}) \\
 &= \frac{1}{2} \sum_{m \neq n} (J_n^{off}(a'_n) - J_n^{off}(a_m)) I_{\{a_m=a'_n\}} \\
 & \quad + \frac{1}{2} \sum_{m \neq n} (J_n^{off}(a_m) - J_n^{off}(a'_n)) I_{\{a'_n=a_m\}} + J_n^{mine}(a'_n) - J_n^{mine}(a_n) \\
 &= \sum_{m \neq n} (J_n^{off}(a'_n) + J_m^{off}(a_m)) I_{\{a_m=a'_n\}} + J_n^{mine}(a'_n) - J_n^{mine}(a_n) \\
 &= J(a'_n, a_{-n}) - J(a_n, a_{-n}).
 \end{aligned} \tag{37}$$

Case 3:  $a_n > 0, a'_n = 0$

Similar to Case 2, it is also straightforward to prove that  $\Psi(a'_n, a_{-n}) - \Psi(a_n, a_{-n}) = J(a'_n, a_{-n}) - J(a_n, a_{-n})$ .

Therefore, the game  $G$  is an exact potential game with the potential function given in (34). Finally, according to the potential game theory [41], our proposed collaborative game  $G$  has an NE and possesses the finite improvement property.

## 5 SIMULATIONS AND PERFORMANCE ANALYSIS

In this section, we perform extensive simulations to verify the performance of the proposed TOBM scheme.

### 5.1 Simulation Setting

We here leverage the widely used mobile wireless dataset provided by Shanghai Telecom<sup>1</sup> for numerical simulations. We select maximum 500 mobile phones as EDs and an MEC server in a sub-area of Shanghai city with the geographical distribution as illustrated in Fig. 5. Moreover, IoT sensor data traces from location services [43] collected during 6 months in 2014 are selected as data tasks for task offloading simulations. Inspired by [11], [13], [14], [32], our simulation parameters are configured as in Table 2.

Moreover, the channel gains  $h_n^k$  are generated using distance-dependent path-loss model  $L[dB] = 140.7 + 36.7 \log_{10} d_{[km]}$ . For the proposed multi-agent DRL algorithm, the discount factor  $\gamma$  equals 0.85 and the replay memory

TABLE 2  
Simulation Parameters

Parameter	Value
Number of EDs $N$	[5-500]
Number of sub-bands at the BS $K$	30
Size of task $D_n$	[0.1-5] MB
Task CPU workload $C_n$	[0.8-1.5] Gcycles
Maximum task execution latency $\tau_n$	1000 ms
The transmit power $p_n^k$	[0-24] dBm
The background noise variance $\sigma^2$	-100 dBm
The system bandwidth $W$	20 MHz
The ED's and MEC server's computing capability $f_n^l, f_n^e$	[0.1-1] GHz, 5 GHz
The ED's energy coefficient $\kappa$	$5 * 10^{-27}$
The weights of time and energy costs $\lambda_n^T, \lambda_n^E$	0.6, 0.4
The CPU resource for block verification $\phi_n$	$[10^3-10^6]$ CPU cycles
Input/output block data sizes $B, B^e$	[50-500] KB, [10-50] KB
The uplink/downlink block transmission rate $r_n^u, r_n^d$	[100-250] Kbps, [100-250] Kbps
The defined block broadcasting parameter $\xi$	0.5

capacity and training batch size are set to  $10^5$  and 128, respectively. The update step  $\zeta$  in the critic-actor training is set to 0.8. The used DNN structure has three hidden layers (64, 32 and 32 neurons) with ReLU as the activation function and Adam as the optimizer. For blockchain mining, we set up 10 transactions per block and vary the numbers of mining nodes (i.e., EDs) from 2 to 100. The other mining parameters are included in Table 2. The results from simulation are averaged from 50 runs of numerical simulations.

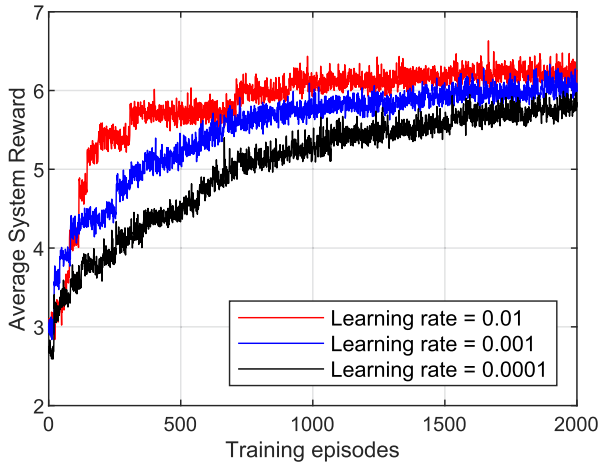
### 5.2 Evaluation of Training Performance

We first evaluate the training performance of our proposed algorithm. To prove the advantage of the proposed cooperative MA-DDPG algorithm, we compare its performance with the state-of-the-art non-cooperative schemes, including DDPG, actor-critic [26] and DQN [18]. Here, DDPG and actor-critic are policy-based algorithms where each ED agent only observes the local information and does not the information of other EDs during the training. Meanwhile, DQN is a value-based algorithm where each ED also has no information of other EDs.

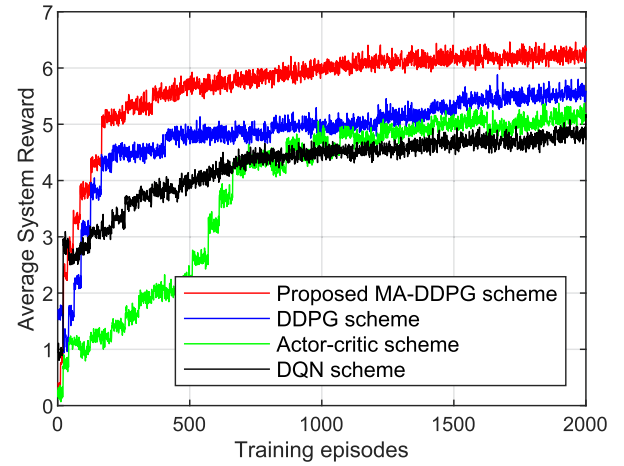
Fig. 6a shows the performance of average system reward with different learning rates  $\alpha$ . It can be seen that the learning rate affects the learning rewards over the training episodes. That is, when the learning rate decreases, the convergence performance of the proposed algorithm decreases due to slow learning speed. Based on our experimental results, the learning value  $\alpha = 0.01$  yields the best reward performance and has good convergence rate and thus we use it in the following system simulations and evaluations.

Fig. 6b shows the learning curves of the average system reward with the increase of episodes for an MEC system with 50 EDs. It is clear that our MA-DDPG scheme is more robust and yields the best performance in terms of average system reward, compared to the baseline schemes. This is because the proposed scheme allows EDs to learn mutually the cooperative policy which helps reduce the channel

1. <http://www.sguangwang.com/dataset/telecom.zip>



(a) Comparison of average system rewards with different learning rates.



(b) Comparison of average system rewards with different algorithms.

Fig. 6. Evaluation of training performance.

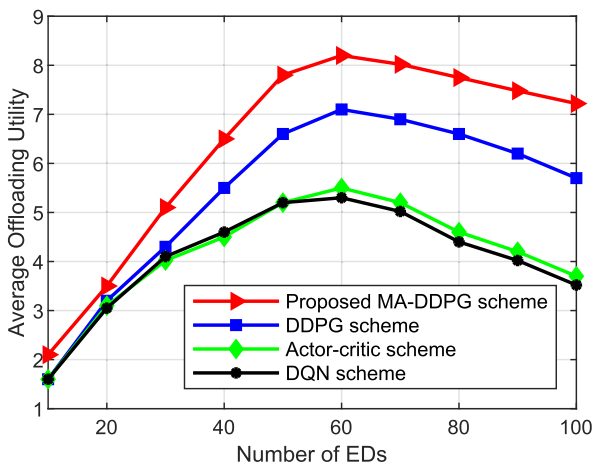
congestion and user interference, and enhance computational resource efficiency for improving the overall system reward. Meanwhile, in the DQN and actor-critic schemes, EDs greedily access the wireless channel spectrum to maximize their own utility as much as possible without collaborating with each other, which increases the possibility of channel collision and thus results in higher offloading latency. Consequently, the average system reward becomes worse. Although the DDPG scheme shows a better reward performance than these two schemes, it still remains a non-stationary learning issue and its average reward is lower than that of the MA-DDPG scheme.

### 5.3 Evaluation of Task Offloading Utility

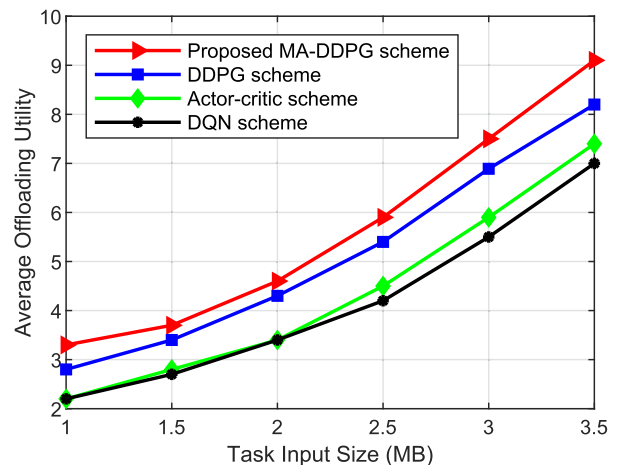
Next, Fig. 7a indicates the performance of average offloading utility versus the different numbers of offloaded EDs. It can be seen that when the number of EDs is small ( $< 60$ ), the average offloading utility increases with the number of EDs because in this case, the MEC system can support sufficient spectrum and computing resources to handle all tasks offloaded from EDs. However, after exceeding some thresholds (e.g.,  $N = 60$  EDs), the system utility decreases because

the higher the number of offloaded EDs, the higher the competition of resource usage (i.e., channel spectrum). Note that the configured number of sub-bands  $K = 30$  is relatively low with respect to the increase in the number of EDs, thus the channel bandwidth allocated for each ED decreases when there are more EDs in the system. In turn, this increases the offloading latency and thus, degrades the overall offloading utility. Nevertheless, our MA-DDPG scheme still achieves the best utility performance due to its cooperative offloading policies among EDs compared to the other schemes with selfish learning. For instance, in the case of 100 EDs, the average offloading utility of the MA-DDPG scheme is 22.5%, 37.5%, and 43.6% higher than those of the DDPG, actor-critic, and DQN schemes, respectively. These results also imply that as the EDs number increases, the cooperative policy learned by MA-DDPG becomes more important in the cooperative edge task offloading.

Moreover, we evaluate the effect of different task sizes on the average offloading utility as shown in Fig. 7b. We find that a higher task size results in a higher offloading utility. Specifically, when the task size is relatively high ( $> 2$  MB), the offloading utility increases significantly. The reason is



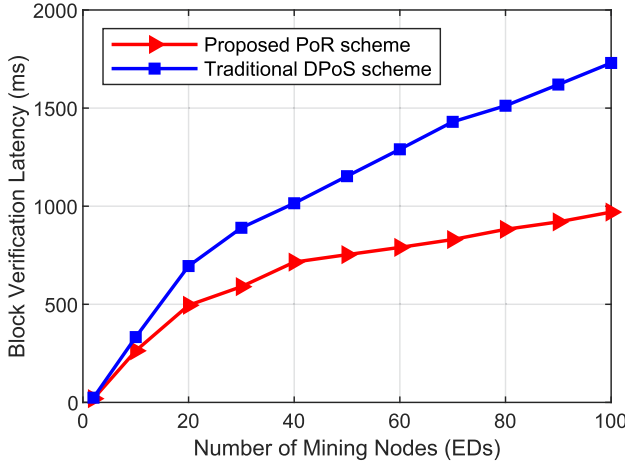
(a) Average offloading utility with different numbers of EDs.



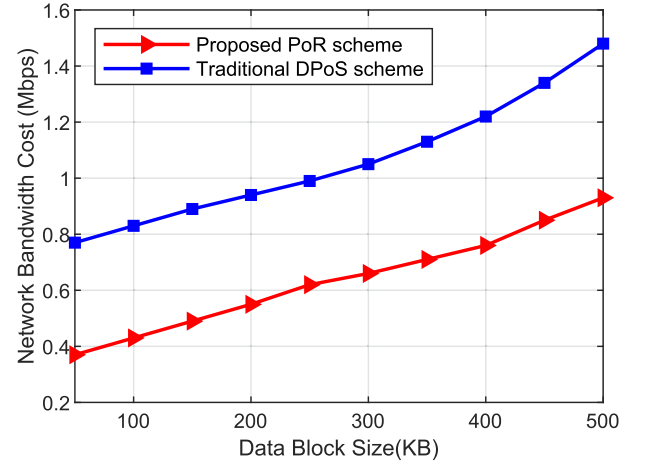
(b) Average offloading utility with different IoT data task sizes.

Fig. 7. Evaluation of task offloading performance.





(a) Comparison of block verification latency.



(b) Comparison of network bandwidth cost.

Fig. 8. Evaluation of blockchain performance.

that the edge computation mode becomes more efficient than the local execution mode in terms of lower computing cost in handling the larger-size tasks due to the high MEC capability. This leads to the increase of the user utility  $J_n$  which thus enhances the system-inter utility  $J$ . Particularly, when the task size increases, the advantage of the proposed MA-DDPG scheme over the baselines become more profound, with the larger performance gaps and better utilities. For example, as the task size is 3.5 MB, the proposed MA-DDPG scheme achieves 30%, 24.5% and 21.7% higher off-loading utilities compared with the DDPG, actor-critic, and DQN schemes, respectively.

#### 5.4 Evaluation of Blockchain Performance

Here, we evaluate the performance of our proposed PoR consensus scheme via numerical simulations using Python programming and compare it with the traditional DPoS scheme [38] via the verification block latency and bandwidth usage metrics.

We first show the block verification latency performance versus different numbers of mining nodes with the block size fixed at 50 KB and compare with the traditional DPoS scheme [38]. As illustrated in Fig. 8a, our proposed PoR scheme requires significantly less time for mining blocks, compared to the DPoS scheme thanks to the optimized

block verification procedure. Although the time required for block verification increases with the increasing number of miners, our solution still achieves a much better performance than that of the DPoS scheme. This result confirms our lightweight consensus design that is thus well suitable for large-scale blockchain-based MEC systems.

Next, Fig. 8b indicates the simulation result in terms of network bandwidth cost spent by the mining process for different data block sizes from 50 KB to 500 KB in the edge blockchain network. Due to the optimized block exchange procedure where each ES only needs to contact with one different miner for transaction verification, instead of using a repeated process, our PoR scheme can save much network bandwidth resources, compared to the DPoS scheme [38].

#### 5.5 Evaluation of Overall System Utility Performance

In this subsection, we evaluate the performance of the overall system utility of our TOBM scheme enabled by the joint consideration of offloading utility and mining utility. The performances of our cooperative scheme with our PoR mining design and other non-cooperative schemes with PoR and DPoS mining are illustrated in Fig. 9. Unsurprisingly, our TOBM scheme with a PoR mining design achieves the best overall system utility. The reasons for this observation

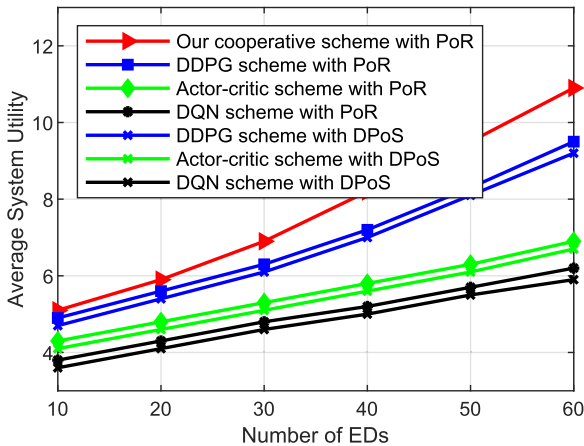


Fig. 9. Comparison of system utility with non-cooperative schemes.

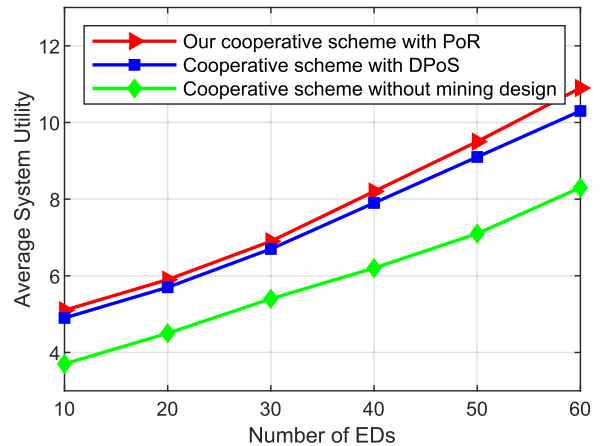
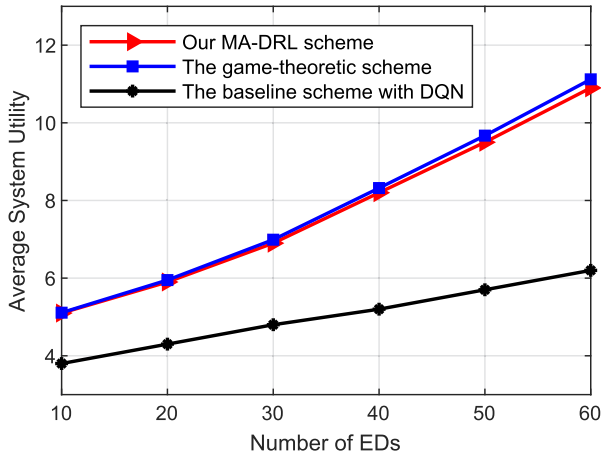
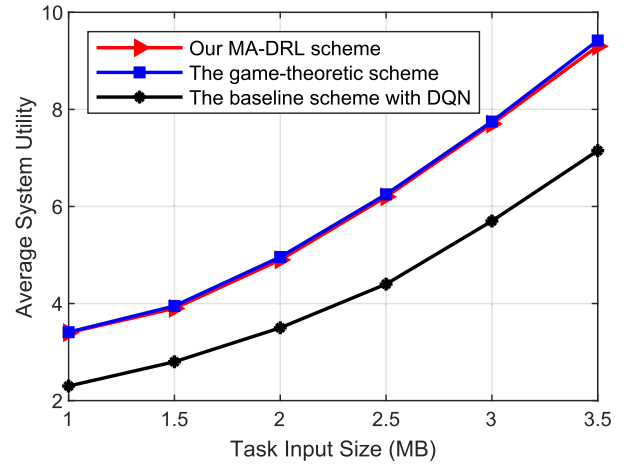


Fig. 10. Comparison of system utility with cooperative schemes.



(a) Average system utility with different numbers of EDs.



(b) Average system utility with different IoT data task sizes.

Fig. 11. Comparison of learning and game-theoretic approaches.

are two-fold. First, our offloading scheme with a cooperative MA-DDPG algorithm outperforms other non-cooperative offloading schemes in terms of a better offloading utility, as evidenced in Fig. 7. Second, our PoR design yields a lower mining latency which consequently increases the mining utility, as explained in Section 3.1.1. As a result, our scheme with a cooperative offloading design and a lightweight mining design achieves a much better overall system utility than the other non-cooperative offloading schemes with DPoS design. Moreover, due to its better mining utility, our PoR design contributes to better overall system utilities in each non-cooperative offloading scheme, compared to the use of DPoS design.

Furthermore, we compare the system utility performance of our cooperative TOBM scheme with other cooperative schemes, including a cooperative scheme with DPoS design [27] and a cooperative scheme without mining design [26]. As shown in Fig. 10, our TOBM scheme with PoR design achieves a better system utility than the cooperative scheme with DPoS design, thanks to the better mining utility of our PoR framework. Meanwhile, the cooperative scheme in [26] has lowest system utility due to the lack of consideration of mining design.

## 5.6 Comparison of Learning and Game-Theoretic Approaches

Next, we compare the utility performance between the MA-DRL scheme and the game-theoretic scheme, where the DQN scheme is used as the baseline, after averaging the results from 5 simulations. As shown in Fig. 11a, the game-theoretic approach can achieve the optimal system utility, compared to the MA-DRL scheme, when increasing the number of MDs. This is because in the game approach, each MD can obtain the full knowledge of other MDs' information such as the information of offloading decisions and mining status via the collaborative interactions. This allows each MD to determine the optimal computation offloading strategy to achieve the converged point of NE. Meanwhile, the MA-DRL scheme can also achieve reasonably close results, where the physical parameters of MDs are time-varying, and each MD can compute the approximately optimal computation offloading strategy without requiring any

prior information about other MDs. Similar performances can also be seen in Fig. 11b, when increasing the size of task inputs.

## 6 CONCLUSIONS AND FUTURE WORK

In this article, we have proposed a novel cooperative TOBM scheme to enable a joint design of task offloading and blockchain mining in blockchain-based MEC systems. First, we have proposed a new cooperative offloading framework that enables EDs to learn offloading policies in a collaborative manner. Then, we have designed a new PoR mining scheme enabled by a lightweight block verification strategy. To this end, we have formulated a joint offloading and mining optimization problem which is solved by an MA-DRL algorithm. We then have derived a game-theoretic solution to model the competition among EDs in offloading and mining as a potential game, and proved the existence of a pure Nash equilibrium. Simulation results have clearly shown the significant advantages of our proposed scheme over the existing schemes in terms of higher system rewards with better offloading utility and lower blockchain costs which thus enhance the overall system utility.

Our proposed approach has potential for future intelligent mobile networks, where EDs are able to build distributed intelligent solutions via our cooperative DRL model for enabling intelligent computation, communications and network control [44]. In future work, it is of interest to consider fair resource allocation strategies for simultaneously supporting the edge computation and blockchain services. The tradeoff between mining security and latency should also be studied to strike a beneficial balance between these two important design factors before integrating into MEC. Moreover, resource trading solutions should be developed to enable reliable energy purchase for resource-constrained edge nodes in blockchain-based MEC systems.

## REFERENCES

- [1] H. Guo, J. Liu, J. Ren, and Y. Zhang, "Intelligent task offloading in vehicular edge computing networks," *IEEE Wireless Commun.*, vol. 27, no. 4, pp. 126–132, Aug. 2020.

- [2] H. Guo and J. Liu, "UAV-enhanced intelligent offloading for Internet of Things at the edge," *IEEE Trans. Ind. Inform.*, vol. 16, no. 4, pp. 2737–2746, Apr. 2020.
- [3] P. A. Apostolopoulos, G. Fragkos, E. E. Tsiropoulou, and S. Papavassiliou, "Data offloading in UAV-assisted multi-access edge computing systems under resource uncertainty," *IEEE Trans. Mobile Comput.*, early access, Mar. 31, 2021, doi: [10.1109/TMC.2021.3069911](https://doi.org/10.1109/TMC.2021.3069911).
- [4] 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.
- [5] 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.
- [6] S. Guo, X. Hu, S. Guo, X. Qiu, and F. Qi, "Blockchain meets edge computing: A distributed and trusted authentication system," *IEEE Trans. Ind. Inform.*, vol. 16, no. 3, pp. 1972–1983, Mar. 2020.
- [7] J. Kang *et al.*, "Blockchain for secure and efficient data sharing in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4660–4670, Jun. 2019.
- [8] J. Guo, X. Ding, and W. Wu, "Reliable traffic monitoring mechanisms based on blockchain in vehicular networks," *IEEE Trans. Rel.*, early access, Jan. 13, 2021, doi: [10.1109/TR.2020.3046556](https://doi.org/10.1109/TR.2020.3046556).
- [9] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 4924–4938, Aug. 2017.
- [10] Z. Kuang, L. Li, J. Gao, L. Zhao, and A. Liu, "Partial offloading scheduling and power allocation for mobile edge computing systems," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6774–6785, Aug. 2019.
- [11] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7944–7956, Aug. 2019.
- [12] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Deep reinforcement learning for collaborative offloading in heterogeneous edge networks," in *Proc. IEEE/ACM 21st Int. Symp. Cluster, Cloud Internet Comput.*, May 2021, pp. 297–303.
- [13] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11 158–11 168, Nov. 2019.
- [14] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Privacy-preserved task offloading in mobile blockchain with deep reinforcement learning," *IEEE Trans. Netw. Serv. Manage.*, vol. 17, no. 4, pp. 2536–2549, Dec. 2020.
- [15] Y. Yu, S. C. Liew, and T. Wang, "Multi-agent deep reinforcement learning multiple access for heterogeneous wireless networks with imperfect channels," *IEEE Trans. Mobile Comput.*, early access, Feb. 9, 2021, doi: [10.1109/TMC.2021.3057826](https://doi.org/10.1109/TMC.2021.3057826).
- [16] X. Xu, X. Zhang, H. Gao, Y. Xue, L. Qi, and W. Dou, "BeCome: Blockchain-enabled computation offloading for IoT in mobile edge computing," *IEEE Trans. Ind. Inform.*, vol. 16, no. 6, pp. 4187–4195, Jun. 2020.
- [17] J. Heydari, V. Ganapathy, and M. Shah, "Dynamic task offloading in multi-agent mobile edge computing networks," in *Proc. IEEE Glob. Commun. Conf.*, Dec. 2019, pp. 1–6.
- [18] Y. Zhang, B. Di, Z. Zheng, J. Lin, and L. Song, "Joint data offloading and resource allocation for multi-cloud heterogeneous mobile edge computing using multi-agent reinforcement learning," in *Proc. IEEE Glob. Commun. Conf.*, Dec. 2019, pp. 1–6.
- [19] X. Liu, J. Yu, and Y. Gao, "Multi-agent reinforcement learning for resource allocation in IoT networks with edge computing," 2020, *arXiv:2004.02315*.
- [20] G. Xu, Y. Liu, and P. W. Khan, "Improvement of the DPoS consensus mechanism in blockchain based on vague sets," *IEEE Trans. Ind. Inform.*, vol. 16, no. 6, pp. 4252–4259, Jun. 2020.
- [21] L. Jiang, S. Xie, S. Maharjan, and Y. Zhang, "Joint transaction relaying and block verification optimization for blockchain empowered D2D communication," *IEEE Trans. Veh. Technol.*, vol. 69, no. 1, pp. 828–841, Jan. 2020.
- [22] X. Qiu, L. Liu, W. Chen, Z. Hong, and Z. Zheng, "Online deep reinforcement learning for computation offloading in blockchain-empowered mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 8050–8062, Aug. 2019.
- [23] S. Guo, Y. Dai, S. Guo, X. Qiu, and F. Qi, "Blockchain meets edge computing: Stackelberg game and double auction based task offloading for mobile blockchain," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 5549–5561, May 2020.
- [24] 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.
- [25] Z. Zhang, Z. Hong, W. Chen, Z. Zheng, and X. Chen, "Joint computation offloading and coin loaning for blockchain-empowered mobile-edge computing," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 9934–9950, Dec. 2019.
- [26] J. Feng, F. Richard Yu, Q. Pei, X. Chu, J. Du, and L. Zhu, "Cooperative computation offloading and resource allocation for blockchain-enabled mobile-edge computing: A deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6214–6228, Jul. 2020.
- [27] F. Guo, F. R. Yu, H. Zhang, H. Ji, M. Liu, and V. C. M. Leung, "Adaptive resource allocation in future wireless networks with blockchain and mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 1689–1703, Mar. 2020.
- [28] Z. Li, M. Xu, J. Nie, J. Kang, W. Chen, and S. Xie, "NOMA-enabled cooperative computation offloading for blockchain-empowered Internet of Things: A learning approach," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2364–2378, Feb. 2021.
- [29] M. Li, D. Hu, C. Lal, M. Conti, and Z. Zhang, "Blockchain-enabled secure energy trading with verifiable fairness in industrial Internet of Things," *IEEE Trans. Ind. Inform.*, vol. 16, no. 10, pp. 6564–6574, Oct. 2020.
- [30] Z. Yang, K. Liu, Y. Chen, W. Chen, and M. Tang, "Two-level stackelberg game for IoT computational resource trading mechanism: A smart contract approach," *IEEE Trans. Serv. Comput.*, early access, Sep. 18, 2020, doi: [10.1109/TSC.2020.3024729](https://doi.org/10.1109/TSC.2020.3024729).
- [31] 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.
- [32] J. Wang, J. Hu, G. Min, W. Zhan, Q. Ni, and N. Georgalas, "Computation offloading in multi-access edge computing using a deep sequential model based on reinforcement learning," *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 64–69, May 2019.
- [33] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst., Man, Cybern.*, vol. 38, no. 2, pp. 156–172, Mar. 2008.
- [34] Z. Cao, P. Zhou, R. Li, S. Huang, and D. Wu, "Multiagent deep reinforcement learning for joint multichannel access and task offloading of mobile-edge computing in industry 4.0," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6201–6213, Jul. 2020.
- [35] D. Kwon, J. Jeon, S. Park, J. Kim, and S. Cho, "Multi-agent DDPG-based deep learning for smart ocean federated learning IoT networks," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9895–9903, Apr. 2020.
- [36] S. Wang *et al.*, "A data-driven multi-agent autonomous voltage control framework using deep reinforcement learning," *IEEE Trans. Power Syst.*, vol. 35, no. 6, pp. 4644–4654, Apr. 2020.
- [37] D. Nguyen, P. Pathirana, M. Ding, and A. Seneviratne, "Secure computation offloading in blockchain based IoT networks with deep reinforcement learning," *IEEE Trans. Netw. Sci. Eng.*, early access, Aug. 24, 2021, doi: [10.1109/TNSE.2021.3106956](https://doi.org/10.1109/TNSE.2021.3106956).
- [38] W. Sun, J. Liu, Y. Yue, and P. Wang, "Joint resource allocation and incentive design for blockchain-based mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 9, pp. 6050–6064, Sep. 2020.
- [39] F. Yang, W. Zhou, Q. Wu, R. Long, N. N. Xiong, and M. Zhou, "Delegated proof of stake with downgrade: A secure and efficient blockchain consensus algorithm with downgrade mechanism," *IEEE Access*, vol. 7, pp. 118541–118555, 2019.
- [40] R. Lowe, Y. WU, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6379–6390.
- [41] H. Guo and J. Liu, "Collaborative computation offloading for multiaccess edge computing over fiber-wireless networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 5, pp. 4514–4526, May 2018.
- [42] Z. Ning *et al.*, "Partial computation offloading and adaptive task scheduling for 5G-enabled vehicular networks," *IEEE Trans. Mobile Comput.*, early access, Sep. 18, 2020, doi: [10.1109/TMC.2020.3025116](https://doi.org/10.1109/TMC.2020.3025116).



- [43] S. Yang, K. Xu, L. Cui, Z. Ming, Z. Chen, and Z. Ming, "EBI-PAI: Towards an efficient edge-based IoT platform for artificial intelligence," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9580–9593, Jun. 2020.
- [44] D. C. Nguyen *et al.*, "6G Internet of Things: A comprehensive survey," *IEEE Internet Things J.*, early access, Aug. 9, 2021, doi: [10.1109/JIOT.2021.3103320](https://doi.org/10.1109/JIOT.2021.3103320).



**Dinh C. Nguyen** (Member, IEEE) is currently working toward the PhD degree with the School of Engineering, Deakin University, Melbourne, VIC, Australia. He has published more than 20 papers as the first author in top-tier IEEE journals and conferences, such as the *IEEE Transactions on Mobile Computing*, *IEEE Wireless Communications Magazine*, *IEEE Communications Surveys and Tutorials*, *IEEE Internet of Things Journal*, *IEEE GLOBECOM*, *ICC*, and the *CCGrid* conferences. His research interests include wireless communications, federated learning, deep reinforcement learning, blockchain, and edge computing. He was the recipient of prestigious Data61 PhD scholarship, CSIRO, Australia. He was the TPC member of top-tier conferences including IEEE GLOBECOM 2021.



**Ming Ding** (Senior Member, IEEE) received the BS and MS degrees (with first-class Hons.) in electronics engineering, and the Doctor of Philosophy (PhD) degree in signal and information processing from Shanghai Jiao Tong University, Shanghai, China, in 2004, 2007, and 2011, respectively. From April 2007 to September 2014, he was a researcher or senior researcher or principal researcher with Sharp Laboratories of China, Shanghai, China. He is currently a senior research scientist with Data61, CSIRO, Sydney,

NSW, Australia. He has authored more than 140 papers in IEEE journals and conferences in recognized venues, and around 20 3GPP standardization contributions, and a Springer book *Multi-point Cooperative Communication Systems: Theory and Applications*. He also holds 21 U.S. patents and co-invented another more than 100 patents on 4G/5G technologies in CN, JP, KR, EU. His research interests include information technology, data privacy and security, and machine learning and AI. He is currently the editor of the *IEEE Transactions on Wireless Communications* and the *IEEE Wireless Communications Letters*. He was the guest editor/co-chair/co-tutor/TPC member for many IEEE top-tier journals/conferences. He was the recipient of several awards for his research work and professional services.



**Pubudu N. Pathirana** (Senior Member, IEEE) was born in 1970 in Matara, Sri Lanka, and was educated at Royal College Colombo. He received the BE degree (first class Hons.) in electrical engineering and the BSc degree in mathematics, and the PhD degree in electrical engineering from the University of Western Australia in 1996 and 2000, respectively, all sponsored by the government of Australia on EMSS and IPRS scholarships, respectively. He was a postdoctoral research fellow with Oxford University, Oxford, a research fellow with the

School of Electrical Engineering and Telecommunications, University of New South Wales, Sydney, NSW, Australia, and a consultant to the Defence Science and Technology Organization, Australia, in 2002. He was a visiting professor with Yale University in 2009. He is currently a professor and the head of the Discipline, Mechatronics, Electrical and Electronic Engineering and the director of the Network Sensing and Biomedical Engineering Research Group, the School of Engineering, Deakin University, Geelong, Australia. His research interests include bio-medical assistive device design, human motion capture, mobile/wireless and IoT networks, rehabilitation robotics, and signal processing.



**Aruna Seneviratne** (Senior Member, IEEE) is currently a foundation professor of telecommunications with the University of New South Wales, Sydney, NSW, Australia, where he holds the Mahanakorn chair of telecommunications. He was with a number of other Universities in Australia, U.K., and France, and industrial organizations, including Muirhead, Standard Telecommunication Labs, Avaya Labs, and Telecom Australia (Telstra). He held visiting appointments with INRIA, France. His research interests include physical analytics: technologies that enable applications to interact intelligently and securely with their environment in real time. Most recently, his team has been working on using these technologies in behavioral biometrics, optimizing the performance of wearables, and the IoT system verification. He was the recipient of number of fellowships, including one at British Telecom and one at Telecom Australia Research Labs.



**Jun Li** (Senior Member, IEEE) received PhD degree in electronic engineering from Shanghai Jiao Tong University, Shanghai, China, in 2009. From January 2009 to June 2009, he was a research scientist with the Department of Research and Innovation, Alcatel Lucent Shanghai Bell. From June 2009 to April 2012, he was a postdoctoral fellow with the School of Electrical Engineering and Telecommunications, University of New South Wales, Sydney, NSW, Australia. From April 2012 to June 2015, he was a research fellow with the

School of Electrical Engineering, University of Sydney, Sydney, NSW, Australia. Since 2015, he has been a professor with the School of Electronic and Optical Engineering, Nanjing University of Science and Technology, Nanjing, China. From 2018 to 2019, he was a visiting professor with Princeton University. He has coauthored more than 200 papers in IEEE journals and conferences and holds one U.S. patents and more than ten Chinese patents in his areas of research, which include network information theory, game theory, distributed intelligence, multiple agent reinforcement learning, and their applications in ultra-dense wireless networks, mobile edge computing, network privacy and security, and industrial Internet of things. He was the editor of the *IEEE Communication Letters* and a TPC member for several flagship IEEE conferences. He was the recipient of the Exemplary Reviewer of *IEEE Transactions on Communications* in 2018 and the Best Paper Award from the IEEE International Conference on 5G for Future Wireless Networks in 2017.



**H. Vincent Poor** (Fellow, IEEE) received the PhD degree in electrical engineering and computer sciences from Princeton University in 1977. From 1977 to 1990, he was on the faculty of the University of Illinois at Urbana-Champaign. Since 1990, he has been on the faculty at Princeton, where he is currently the Michael Henry Strater University professor. From 2006 to 2016, he served as the dean of the Princeton's School of Engineering and Applied Science. He has also held visiting appointments with several other universities,

including most recently at Berkeley and Cambridge. His research interests include information theory, machine learning and network science, and their applications in wireless networks, energy systems, and related fields. Among his publications in these areas is the forthcoming book *Machine Learning and Wireless Communications* (Cambridge University Press). Dr. Poor is a member of the National Academy of Engineering and the National Academy of Sciences and a foreign member of the Chinese Academy of Sciences, the Royal Society, and other national and international academies. He received the IEEE Alexander Graham Bell Medal in 2017.

► For more information on this or any other computing topic, please visit our Digital Library at [www.computer.org/csdl](http://www.computer.org/csdl).