Cooperative and Distributed Computation Offloading for Blockchain-Empowered Industrial Internet of Things

Wuhui Chen, *Member, IEEE*, Zhen Zhang, Zicong Hong, Chuan Chen, *Member, IEEE*, Jiajing Wu, *Member, IEEE*, Sabita Maharjan, *Member, IEEE*, Zibin Zheng, *Senior Member, IEEE*, and Yan Zhang, *Senior Member, IEEE*

***Abstract*—Offloading computation-intensive blockchain mining tasks to the edge servers (ESs) is a promising solution for blockchain-empowered Industrial Internet of Things (IIoT) because the computing capabilities in IIoT are usually limited, whereas the blockchain mining tasks are computationally intensive. However, the computation offloading solutions for data processing tasks and for blockchain mining tasks have been studied separately. Moreover, most of the existing solutions for offloading assume that all IIoT devices can directly connect to the ESs or cloud data centers. To address these issues, in this paper, we propose a multihop cooperative and distributed computation offloading algorithm that considers the data processing tasks and the mining tasks together for blockchain-empowered IIoT. First, we study the multihop computation offloading problem for both the data processing tasks and the mining tasks to minimize the economic cost of IIoT devices. Second, we formulate the offloading problem as a potential game in which the IIoT devices can make their decisions autonomously and prove the existence of Nash equilibrium (NE) for the game. Third, we design an efficient distributed algorithm based on exchanging messages between IIoT devices to achieve the NE with low computational complexity. Lastly, our experimental results demonstrate that our distributed algorithm scales well as the number of IIoT devices increases and has the minimum system cost compared with other approaches.**

***Index Terms*—Blockchain, computation offloading, edge computing, Industrial Internet of Things (IIoT).**

Manuscript received March 16, 2019; revised May 6, 2019; accepted May 15, 2019. Date of publication May 22, 2019; date of current version October 8, 2019. This work was supported in part by the National Key Research and Development Plan under Grant 2018YFB1003800, in part by the National Natural Science Foundation of China under Grant 61802450, in part by the Guangdong Province Universities and Colleges Pearl River Scholar Funded Scheme under Grant 2016, in part by the Natural Science Foundation of Guangdong under Grant 2018A030313005, and in part by the Program for Guangdong Introducing Innovative and Entrepreneurial Teams under Grant 2017ZT07X355. *(Corresponding authors: Chuan Chen; Jiajing Wu.)*

W. Chen, Z. Zhang, Z. Hong, C. Chen, J. Wu, and Z. Zheng were with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China (e-mail: chenwuh@mail.sysu.edu.cn; chenchuan@mail.sysu.edu.cn; wujiajing@mail.sysu.edu.cn; zhzibin@mail.sysu.edu.cn; zhangzh297@mail2.sysu.edu.cn; hongzc@mail2.sysu.edu.cn).

S. Maharjan was with the Simula Research Laboratory, University of Oslo, 0316 Oslo, Norway (e-mail: sabita@ieee.org).

Y. Zhang was with the Department of Informatics, University of Oslo, 0316 Oslo, Norway (e-mail: yanzhang@ieee.org).

Digital Object Identifier 10.1109/JIOT.2019.2918296

I. Introduction

THE Industrial Internet of Things (IIoT) connects a large scale of homogeneous and heterogeneous devices in industrial applications for economic benefits, efficiency improvements, and reduced human interference, in which IIoT devices may need to interact with each other (e.g., data exchanging, energy sharing, and computing resource trading) [1], [2]. However, because the IIoT devices (e.g., robots, electric vehicles, and smart grid) may be owned by different parties, their interactions inevitably involve different self-interested parties [3], [4], which can lead to conflicts among them. Thus, no single party can be really trusted, especially in a large number of processing interactions among several parties [5]–[7]. Therefore, it is important to build up trust among different parties involved in IIoT.

Blockchain technologies, which allow IIoT to maintain information transparency and build up trust among participants via blockchain's decentralized, tamper-proof, secure, and traceable characteristics, can provide feasible solutions for trust enhancement, cost reduction, and failure avoidance for IIoT [8]–[10]. However, using IIoT devices to support the sophisticated security approach is challenging because the mining tasks consume enormous computing power [11]–[13]. The limited computing and storage capabilities of IIoT devices limit their real-world applications. Interestingly, many studies have been devoted to solving this problem, among which the edge-computing model allowing IIoT devices to offload the computation-intensive tasks (e.g., solving proof-of-work puzzles) to the edge servers (ESs) is a promising solution for blockchain-empowered IIoT [14]–[16].

The existing work [17], [18] on the computation offloading problem has achieved some positive results. However, they have not incorporated the following vital issues. First, they do not consider the data processing tasks and the mining tasks together in computation offloading. Instead, they only study the computation offloading problem separately either for data processing tasks or for mining tasks. However, in practice, these two types of tasks need to be processed simultaneously in blockchain-empowered IIoT. Therefore, the computation offloading solution must consider the data processing tasks and the mining tasks together to achieve resource optimization globally. Second, most of these works assume that all IIoT

devices can connect to the ESs directly via wireless networks. However, in real-world scenarios, some IIoT devices may experience poor connectivity or may even fail to connect to any edge access point (AP) directly because they are far away from the APs. These devices then need to connect to well-connected neighboring devices to offload tasks to the ESs, i.e., a multihop computation offloading scenario, in which a task may need to be transmitted by more than one device from the original device to the ES. Therefore, some IIoT devices may need to transmit their own tasks and tasks from neighboring devices, making the computation offloading problem more complex.

In this paper, we propose a multihop cooperative and distributed computation offloading solution to consider the data processing tasks and the mining tasks together for blockchain-empowered IIoT. To address the first issue presented, we develop a game-theory-based distributed computation offloading strategy to allow the data processing tasks and the mining tasks to be offloaded to the ESs to achieve global resource optimization. To address the second issue presented, we formulate the offloading problem as a multihop computation offloading game (MCOG) and design a distributed algorithm by which the game can quickly converge to a stable state, i.e., a Nash equilibrium (NE). To the best of our knowledge, we are the first to study the multihop computation offloading problem while considering both the data processing tasks and the mining tasks for blockchain-empowered IIoT.

Our main contributions are summarized as follows.

1) We first investigate the multihop computation offloading solution that addresses the data processing tasks and the mining tasks together to minimize the economic cost of IIoT devices for blockchain-empowered IIoT. Then, we model the offloading problem as a potential game.
2) We show the existence of NE for this offloading game and develop a distributed algorithm to quickly reach the NE point for the game. We also propose an incentive model to motivate the IIoT devices to transmit data cooperatively and introduce a time-slot mechanism that enables the algorithm to be put into practice well.
3) Lastly, we evaluate our distributed algorithm through extensive experiments. The experimental results demonstrate that our algorithm has the minimum system cost under different environments, is more stable than other algorithms, and scales well as the number of IIoT devices increases.

The remainder of this paper is organized as follows. We discuss related work in Section II. We describe the system model in Section III. We formulate the computation offloading problem in Section IV and describe the game in Section V. In Section VI, we design a distributed algorithm to reach the NE for the game. Lastly, we evaluate the performance of our algorithm in Section VII and conclude this paper in Section VIII.

II. Related Work

The computation offloading problem for IoT has been studied extensively [19], [20]. For example, Sun *et al.* [21] proposed an efficient task scheduling scheme in the vehicular cloud by jointly considering the instability of resources, the heterogeneity of vehicular computing capabilities, and the interdependency of computing tasks. Shah-Mansouri and Wong [22] used a computation offloading game to model the competition between IoT users who aim to maximize their own quality of experience in a hierarchical fog-cloud computing paradigm. Zheng *et al.* [23] investigated the problem of multiuser computation offloading for mobile cloud computing under dynamic environments and proposed a stochastic game-theory approach for dynamic computation offloading. Guo *et al.* [24] proposed an iterative searching-based task offloading scheme that jointly optimizes task offloading, computational frequency scaling, and transmit-power allocation. Zhang *et al.* [25] developed a multiqueue model to explore the impact of offloading policies on the performance of IoT devices with the computing resources supported by an edge-computing server. Guo *et al.* [26] studied the computation offloading problem in a mobile edge-computing framework and proposed a two-tier game-theory greedy offloading scheme as the solution. Chen *et al.* [27] proposed a dynamic computation offloading algorithm to solve the problem in a distributed manner for IIoT.

In addition, only a few recent studies have started to study the multihop computation offloading problem. Compared with the single-hop computation offloading problem, the multihop computation offloading problem is more complex because a task may be transmitted by more than one rational intermediate node along the path toward the ES. Therefore, this problem involves the choice of the transmission path and incentives for the rational intermediate nodes. Al-Shatri *et al.* [28] proposed a distributed algorithm on computation offloading in multihop networks that aims to minimize the total network energy consumed. However, the transmission path of each node is given in advance and unique, and the entire routing topology is tree shaped. Hong *et al.* [29] addressed the multihop computation offloading problem by proposing a game-theory-based solution where the transmission path can dynamically change. However, the trust and security issues among different participants remain challenging.

More recently, some works, such as [17], [18], and [30] have studied the computation offloading problem for blockchain-based IoT to solve the computation-intensive proof-of-work puzzle. Chatzopoulos *et al.* [17] proposed a multilayer computation offloading framework, FlopCoin, to integrate a distributed incentive scheme and a distributed reputation mechanism for mobile devices. Liu *et al.* [18] proposed a novel mobile edge-computing-enabled wireless blockchain framework where the computation-intensive mining tasks can be offloaded to the nearby edge-computing nodes and the cryptographic hashes of blocks can be cached in the ESs. Luong *et al.* [14] developed an optimal auction based on deep learning for edge-resource allocation for mobile blockchain networks. Jiao *et al.* [31] proposed an auction-based edge-computing resource allocation mechanism for the edge-computing service provider to support the mobile blockchain mining task. Xiong *et al.* [12] studied the resource pricing between the cloud/fog providers and the miners in a

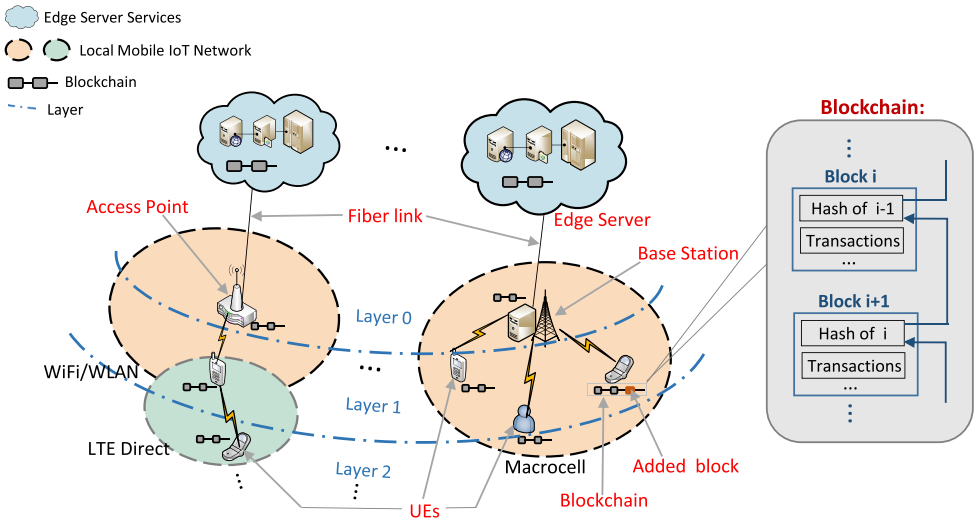


Fig. 1. Illustration of the multihop computation offloading framework for blockchain-empowered IIoT.

proof-of-work-based blockchain network using a game-theory approach. Li *et al.* [32] proposed a double auction-based computing resource trading approach for edge-cloud-assisted IoT. Moura and Hutchison [33] discussed some Stackelberg game-based blockchain mining offloading approaches in which the leader is the IoT device and the followers are the ESs. However, most studies have not considered the data processing tasks and the mining tasks together in computation offloading. In this paper, we study the multihop computation offloading problem while considering the data processing tasks and the mining tasks together for blockchain-empowered IIoT.

III. System Model

Fig. 1 shows the multihop computation offloading framework for blockchain-empowered IIoT. The framework consists of three main components, the IIoT devices, the ESs, and the blockchain. The IIoT devices can communicate with the neighbor peers constituting the local communities via LTE Direct Technology [34]. Some of the peers are also directly connected with the APs that provide access services to the edge layer. We denote by $\mathcal{B}$ the set of APs. The ESs provide sufficient computing and storage resources for mining and consensus processing and data processing tasks. Lastly, the blockchain is used to ensure high credibility and security. In what follows we explain our model in detail.

First, we consider the IIoT network of n user equipments (UEs). Let $\mathcal{N}$ denote the set of the UEs in the network. Each UE $i \in \mathcal{N}$ performs a data processing task $\mathcal{K}_i^1(s_i^1, T_i^1, l_i^1)$, where s_i^1 is the input data size, T_i^1 is the deadline, and l_i^1 is the number of CPU cycles needed to complete the task. For convenience, we call this kind of task *normal task*. Besides, we assume that some UEs can also execute a mining task $\mathcal{K}_i^0(s_i^0)$ if they decide to mine, where s_i^0 is a constant denoting the size of the block being mined. In summary, there are two types of task in the IIoT network: 1) the normal task and 2) the mining task. We assume that the normal task can be completed either locally or in the ES, and the mining task can only be completed in the ES if its corresponding UE decides to participate in the mining process. We use O_i to denote the set of UE i's tasks that are offloaded to the ES. For UE i that performs normal tasks and mining tasks, $O_i = \{\mathcal{K}_i^0, \mathcal{K}_i^1\}$ if it offloads its normal task and mining task to the ES and $O_i = \{\mathcal{K}_i^0\}$ if it only offloads its mining task to the ES. For UE i that only performs normal tasks, $O_i = \{\mathcal{K}_i^1\}$ if it offloads its normal task to the ES and $O_i = \emptyset$ if it performs its normal task locally.

Second, in the multihop computation offloading framework, an incentive model is required for incentivizing the intermediate nodes to forward the tasks of other nodes and incentivizing the ESs to provide computing resources. First, the UEs that offload tasks via UE i should give certain rewards to motivate UE i to help them transmit data. Therefore, we assume that each UE i sets its forwarding price p_i based on its transmission rate. The UEs equally share the expense p_i if they all offload tasks via UE i. Besides, the UEs that offload tasks to the ESs should pay the ESs for providing computing resources. We suppose that p^s is the unit price of the computing resources in the ESs, and the ESs are elastic servers that can provide sufficient computing resources for UEs. Let the ESs allocate resources f^1 to support the normal tasks and f^0 to support the mining tasks, where f^1 and f^0 are both constants. According to the description, some transactions may occur between UEs and UEs or ESs. Hence, all UEs, APs, and ESs maintain a blockchain to record these transactions and ensure them to be tamper resistant and traceable.

Lastly, we divide the APs and UEs into many different layers based on their communication distance in the network. A UE that cannot communicate with any other UEs and APs is not included in our model. We assume that the layer where the APs locate in is layer 0. The UEs that can communicate with the APs directly are located in layer 1, and so forth. The maximal layer is denoted as $L_{\max}$. Each UE i located in the layer j can choose one and only one UE that it can communicate with directly from the layer $j-1$ as its parent, denoted by $p(i)$. Sometimes, there will be more than one UEs to offload tasks via UE i; thus, each task set O_j ($j \in \mathcal{N}$ is the UE which

TABLE I
NOTATIONS USED IN THIS PAPER

Notations	Meanings
$\mathcal{N}$	The set of UEs in the network
n	The number of UEs in $\mathcal{N}$
$\mathcal{K}_i^0$	The mining task
s_i^0	The data size of the mining tasks
$\mathcal{K}_i^1$	The normal task
s_i^1	The data size of the normal tasks
T_i^1	The deadline of the normal tasks
l_i^1	The number of CPU cycles of the normal tasks
O_i	The set of UE i's tasks that are offloaded to the ES
s_i	The total data size of the tasks in O_i
f^1	The resources used to support normal tasks by the ESs
f^0	The resources used to support mining tasks by the ESs
p^e	The unit price of energy
p^s	The unit price of the computing resources in the ESs
p_i	The forwarding price of UE i
N_i	The relay set of UE i
$\mathbf{D}$	All possible decision profiles
$\mathbf{D}_i$	The possible decisions of UE i
$\mathbf{d}$	A decision profile that includes the decisions of all UEs
$\mathbf{d}_{-i}$	The decisions of all UEs except UE i
d_i	The decision chosen by UE i
$\mathcal{R}_i(\mathbf{d})$	The expected mining reward of the UE i in $\mathbf{d}$
$V_{\langle i,j\rangle}$	The total transmission rate of the link between i and j
$v_{\langle i,j\rangle}(\mathbf{d})$	The transmission rate that the UEs receive when they offload via UE i in $\mathbf{d}$
$C_i^1(\mathbf{d})$	The cost of the normal task for UE i in $\mathbf{d}$
$C_i^0(\mathbf{d})$	The cost of the mining task for UE i in $\mathbf{d}$
$C_i^{rel}(\mathbf{d})$	The cost of helping others to transmit data in $\mathbf{d}$
$C_i(\mathbf{d})$	The total cost of UE i in $\mathbf{d}$

offloads its tasks to the ESs via UE i) constitutes the relay set N_i of UE i. $|N_i|$ is the size of the set N_i, denoting the number of UEs that offload tasks via UE i.

To describe this paper more clearly, we list the important notations used in this paper in Table I.

IV. PROBLEM FORMULATION

A. Decision Description

We attempt to solve our problem using game theory because it can leverage the intelligence and computing capabilities of IIoT devices and ease the heavy burden of computation and management at the center. To apply game theory, we assume that the players of the game in our model are all UEs, and each UE is bounded rational and can make choice autonomously. We use decisions to describe the choices made by each UE. The possible decisions of UE i are defined as $\mathbf{D}_i$, and the decision chosen by UE i is expressed as $d_i = (a_i, b_i, \mathbf{r}_i)$, where $d_i \in \mathbf{D}_i$

$$a_i = \begin{cases} 0, & \text{if UE } i \text{ does not perform the mining task} \\ 1, & \text{if UE } i \text{ offloads the mining task to the ES} \end{cases}$$

$$b_i = \begin{cases} 0, & \text{if UE } i \text{ performs the normal task locally} \\ 1, & \text{if UE } i \text{ offloads the normal task to the ES} \end{cases}$$

and $\mathbf{r}_i = \{p(i), p(p(i)), \ldots, b\}$, where $b \in \mathcal{B}$ and $p(b) = \text{ES}$, is a node sequence denoting the transmission path from UE i to AP b. Then, AP b transmits the data belonging to the task of UE i to the ES.

Then, the set of all possible decision profiles is $\mathbf{D} = \mathbf{D}_1 \times \mathbf{D}_2 \times \cdots \times \mathbf{D}_n$ and a decision profile that includes the decisions of all UEs is $\mathbf{d} = \{d_1, d_2, \ldots, d_n\}$, where $\mathbf{d} \in \mathbf{D}$. Furthermore, the decisions of all UEs except UE i are denoted by $\mathbf{d}_{-i}$.

B. Cost of the Normal Task

According to the description in Section IV-A, the normal tasks can be performed locally or in the ESs. The cost of these two situations are discussed as follows.

1) Local Computing Cost: In the case of local computation, the normal task of each UE is completed using its own computing resources. For uniform units, we can convert energy into tokens. Thus, we can denote the unit price of energy by p^e. With the above notations, the cost of the normal task computed locally by UE i can be expressed as

$$C_i^{\text{local}} = \sigma l_i^1 p^e \tag{1}$$

where σ is the consumed energy per CPU cycle when the onboard computing resources of UE i are used to compute the normal task and can be measured by the approach in [35].

2) Computation Offloading Cost: In a multihop computation offloading framework, some UEs need to offload their normal tasks to the ESs with assistance from other UEs. Thus, the transmission rate of the link between the sender i and the receiver j is defined as $V_{\langle i,j\rangle}$. We denote by $v_{\langle i,j\rangle}(\mathbf{d})$ the transmission rate that the UEs can get when they offload tasks via UE i in the decision profile $\mathbf{d}$. As in [36], $v_{\langle i,j\rangle}(\mathbf{d})$ depends on the total transmission rate $V_{\langle i,j\rangle}$ and $|N_i(\mathbf{d})|$ that denotes the number of UEs that offload tasks via UE i, and $v_{\langle i,j\rangle}(\mathbf{d})$ can thus be defined as

$$v_{\langle i,j\rangle}(\mathbf{d}) = \frac{V_{\langle i,j\rangle}}{|N_i(\mathbf{d})|}. \tag{2}$$

Based on the incentive model, UE i that decides to offload needs to pay the intermediate UEs in $\mathbf{r}_i$ for helping it transmit data and the ES for providing computing resources. Thus, in the case of computation offloading, the total cost for computing the normal task in the ES includes the reward to the intermediate nodes, the payment to the ES, and the cost for sending data. Thus, each UE i that offloads its normal task to the ES should pay the ES an amount of $A_i^1 = l_i^1 p^s$. Besides, for sending data to the AP or to the intermediate UE, UE i itself also consumes some energy. The transmission power of UE i is denoted by w_i. Therefore, the cost that the normal task $\mathcal{K}_i^1$ of UE i takes to reach and be processed in the ES is

$$C_i^1(\mathbf{d}) = \sum_{j \in \mathbf{r}_i} \frac{p_j}{|N_j(\mathbf{d})|} + A_i^1 + W_i\left(\mathbf{d}, \mathcal{K}_i^1\right) \tag{3}$$

where $W_i(\mathbf{d}, \mathcal{K}_i^1) = (s_i^1 / [v_{\langle i,p(i)\rangle}(\mathbf{d})]) w_i p^e$ is the cost for sending the data belonging to the normal task. Similar to prior work [22], [36], we do not consider the cost for returning the results of normal tasks to original UEs because the size of the result is much smaller than s_i^1 for many applications such as face recognition.

C. Cost of the Mining Task

As mentioned earlier, the computing resources supported by the ESs for the mining tasks of the UEs that decide to mine are equal. Thus, the hash power of miner i is

$$\alpha_i(\mathbf{d}) = \frac{1}{m(\mathbf{d})} \tag{4}$$

where $m(\mathbf{d})$ is the number of miners (i.e., the number of UEs that decide to mine) [31].

In a blockchain network, all miners compete with each other to first complete the mining task and reach consensus and then receive the reward accordingly [37]. The probability of becoming an orphaned block that is discarded because of not reaching a consensus approximately follows the following distribution: $\mathcal{P}(x) = 1 - e^{-\lambda\beta(x)}$, where $-\lambda$ is a constant rate, x is the block size, and $\beta(x)$ is a function of x. We consider that the first miner that mines a block and has its block reach consensus receives the reward of k tokens. Hence, the expected mining reward of miner i can be expressed as

$$\mathcal{R}_i(\mathbf{d}) = k\alpha_i(\mathbf{d})\left(1 - \mathcal{P}\left(s_i^0\right)\right) = \frac{R}{m(\mathbf{d})} \tag{5}$$

where $R = ke^{-\lambda\beta(s_i^0)}$ is a constant.

Similar to the case of a normal task, the UE i that decides to mine with the computing resources of the ES needs to pay the server for consuming the computing resources. When the UE i only offloads its mining task to the ES (i.e., when $a_i = 1$ and $b_i = 0$), the cost of rewarding the intermediate nodes in $\mathbf{r}_i$ for relaying the data of its mining task should be included in the cost function of the mining task. To express the cost of the mining task for UE i in decision profile $\mathbf{d}$, we define an indicator function [36] for UE i

$$I(x, y) = \begin{cases} 1, & \text{if } x = y \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

Each UE that decides to mine with the computing resources of the ES should pay the ES an amount of $A^0 = f^0 p^s$. Thus, the cost that the mining task $\mathcal{K}_i^0$ of UE i takes to reach and be processed in the ES is

$$C_i^0(\mathbf{d}) = I(b_i, 0)\sum_{j\in\mathbf{r}_i}\frac{p_j}{\left|N_j(\mathbf{d})\right|} + A^0 + W_i\left(\mathbf{d}, \mathcal{K}_i^0\right) - \mathcal{R}_i(\mathbf{d}) \tag{7}$$

where $W_i(\mathbf{d}, \mathcal{K}_i^0) = (s_i^0/[v_{\langle i,p(i)\rangle}(\mathbf{d})])w_i p^e$ is the cost for sending the data belonging to the mining task. Here, we neglect the cost for returning the results of mining tasks because the ES just needs to transfer the mining reward to the account of the according miner if it mines a block.

D. Cost of Relaying

We know that O_j is the set of UE j's tasks the are offloaded to the ES, and the total data size of the tasks in O_j is $s_j = s_j^0 I(a_j, 1) + s_j^1 I(b_j, 1)$. $v_{(i,p(i))}(\mathbf{d})$ denotes the transmission rate that the UEs receive when they offload via UE i. Therefore, the time of relaying the task set O_j from UE i to its parent $p(i)$ is calculated as

$$T^{\text{rel}}_{\langle i,p(i)\rangle}\left(\mathbf{d}, O_j\right) = \frac{s_j}{v_{\langle i,p(i)\rangle}(\mathbf{d})}. \tag{8}$$

Then, the cost of relaying the task set O_j from UE i to its parent $p(i)$ is

$$C^{\text{rel}}_{\langle i,p(i)\rangle}\left(\mathbf{d}, O_j\right) = \frac{s_j}{v_{\langle i,p(i)\rangle}(\mathbf{d})} w_i p^e. \tag{9}$$

Except for its own tasks, UE i may help other UEs to transmit data in the multihop computation offloading framework. When helping others transmit data, UE i consumes some energy and receives the forwarding reward p_i as its remuneration at the same time. Thus, the relay cost of helping others to transmit data is

$$C_i^{\text{rel}}(\mathbf{d}) = \sum_{O_j\in N_i(\mathbf{d})\setminus O_i} C^{\text{rel}}_{\langle i,p(i)\rangle}\left(\mathbf{d}, O_j\right) - p_i\eta \tag{10}$$

where $\eta = 1$ if $N_i(\mathbf{d}) \setminus O_i \neq \emptyset$. Otherwise, $\eta = 0$.

E. Total Cost

The total cost of UE i in the decision profile $\mathbf{d}$ can be defined as

$$\begin{aligned} C_i(\mathbf{d}) = {} & C_i^0(\mathbf{d})I(a_i, 1) + C_i^1(\mathbf{d})I(b_i, 1) \\ & + C_i^{\text{local}}I(b_i, 0) + C_i^{\text{rel}}(\mathbf{d}). \end{aligned} \tag{11}$$

Then, the cost of the whole system can be defined as $C(\mathbf{d}) = \sum_{i\in\mathcal{N}} C_i(\mathbf{d})$.

F. Time Restriction

Based on our model, if the UEs offload more tasks via the same path, they share lower cost, but this can result in inefficiency and congestion. To address this issue, we set a deadline for each normal task. To some extent, this setting of deadlines can appropriately distribute the transmission load on the nodes close to the APs, thus reducing the transmission load on these nodes and keeping the load within an acceptable range. The relay time of offloading the normal task $\mathcal{K}_i^1$ of UE i to the ES by the intermediate UEs in $\mathbf{r}_i$ is calculated as

$$T^{\text{rel}}_{\langle i,\text{ES}\rangle}\left(\mathbf{d}, \mathcal{K}_i^1\right) = s_i^1 \sum_{j\in\mathbf{r}_i}\frac{\left|N_j(\mathbf{d})\right|}{V_{\langle j,p(j)\rangle}}. \tag{12}$$

The completion time of computing the normal task $\mathcal{K}_i^1$ in the ESs is

$$T_i^{\text{com}}\left(\mathbf{d}, \mathcal{K}_i^1\right) = \frac{l_i^1}{f^1}. \tag{13}$$

The total time for computing the normal task $\mathcal{K}_i^1$ in the ES is the sum of relay time and completion time. Therefore, if UE i chooses to offload its normal task to the ES, the following condition must be satisfied: $T^{\text{rel}}_{\langle i,\text{ES}\rangle}(\mathbf{d}, \mathcal{K}_i^1) + T_i^{\text{com}}(\mathbf{d}, \mathcal{K}_i^1) \leq T_i^1$. Otherwise, its normal task must be executed locally.

V. Game-Theory Approach

In the multihop computation offloading framework, the UEs are assumed to be selfish, so they will choose an optimal decision based on their own interests after observing the decisions of other UEs. Each UE aims to minimize its own total cost, and

competitively seeks for an optimal decision d_i^* that satisfies

$$\begin{aligned} &\min \ C_i(d_i, \mathbf{d}_{-i}) \\ &\text{s.t. } d_i \in \mathbf{D_i}, \ T^{\text{rel}}_{\langle i,\text{ES}\rangle}\Big(\mathbf{d}, \mathcal{K}_i^1\Big) + T_i^{\text{com}}\Big(\mathbf{d}, \mathcal{K}_i^1\Big) \le T_i^1. \end{aligned} \tag{14}$$

Thus, we consider that the UEs play a noncooperative game $G : \{\mathcal{N}, (\mathbf{D}_i)_{i\in\mathcal{N}}, (C_i)_{i\in\mathcal{N}}\}$, in which the set of UEs $\mathcal{N}$ denotes the set of players, $\mathbf{D}_i$ are the possible decisions of player i, and C_i is the cost function of player i.

Definition 1: An NE of the MCOG is a decision profile $\mathbf{d}^*$ that satisfies

$$C_i\big(d_i^*, \mathbf{d}_{-i}^*\big) \le C_i\big(d_i, \mathbf{d}_{-i}^*\big) \quad \forall d_i \in \mathbf{D}_i. \tag{15}$$

Based on (15), we call d_i' a better decision for UE i if $C_i(d_i', \mathbf{d}_{-i}) \le C_i(d_i, \mathbf{d}_{-i})$, and d_i^* the best response decision to $\mathbf{d}_{-i}$ if it satisfies (14). Thus, we can deduce that each UE i will play its best response decision to $\mathbf{d}_{-i}$ when the NE is reached. In this case, no UE can decrease its total cost by finding a better decision and has incentives to deviate unilaterally. In addition, in order to enable the MCOG to converge to a stable state, we add a constraint: only the UEs that do not help other UEs to transmit data (i.e., $N_i(\mathbf{d}) \setminus O_i = \varnothing$) can choose a better decision. These UEs are called *free* players. Otherwise, they are called *locked* players.

Before proving the existence of NE in the MCOG, we first need to prove that the MCOG in our model is an exact potential game. Because the decision of each player is a three tuple, each player may have multiple states in the game. Because of space limitations, we divide the state transitions into three different types and give a case in each type to complete the proof. The three types are described as follows.

1) A UE i that changes the value of a_i or b_i but keeps the transmission path $\mathbf{r}_i$ unchanged (e.g., from $d_i = (0, 1, \mathbf{r}_i)$ to $d_i' = (1, 1, \mathbf{r}_i)$).
2) A UE i that keeps the value of a_i and b_i unchanged but changes the transmission path $\mathbf{r}_i$ (e.g., from $d_i = (0, 1, \mathbf{r}_i)$ to $d_i' = (0, 1, \mathbf{r}_i')$).
3) A UE i that changes the value of a_i or b_i and the transmission path $\mathbf{r}_i$ at the same time (e.g., from $d_i = (0, 1, \mathbf{r}_i)$ to $d_i' = (1, 1, \mathbf{r}_i')$).

Definition 2: A strategic game $\{\mathcal{N}, (\mathbf{D}_i)_{i\in\mathcal{N}}, (C_i)_{i\in\mathcal{N}}\}$ is an exact potential game if there exists an exact potential function $\Phi : \mathbf{D} \to \mathbb{R}$ such that $\forall i \in \mathbf{N}$

$$\Phi\big(d_i', \mathbf{d}_{-i}\big) - \Phi(d_i, \mathbf{d}_{-i}) = C_i\big(d_i', \mathbf{d}_{-i}\big) - C_i(d_i, \mathbf{d}_{-i}). \tag{16}$$

Theorem 1: The MCOG admits an exact potential function for free UEs

$$\begin{aligned} \Phi(\mathbf{d}) = &\sum_{i\in\mathcal{N}} \sum_{j=1}^{|N_i(\mathbf{d})|} \frac{p_i}{j} - \sum_{i=1}^{m(\mathbf{d})} \left(\frac{R}{i} - A^0\right) + \sum_{i=1}^{n} A_i^1 I(b_i, 1) \\ &+ \sum_{i=1}^{n} C_i^{\text{local}} \ I(b_i, 0) + \sum_{i=1}^{n} Q_i\big(s_i, V_{\langle i,p(i)\rangle}\big) \end{aligned} \tag{17}$$

where $Q_i = (s_i/[V_{\langle i,p(i)\rangle}])w_i p^e$ is a function of s_i and $V_{\langle i,p(i)\rangle}$.

Proof (Case 1): The UE i changes from its previous decision $d_i = (0, 1, \mathbf{r}_i)$ to a better decision $d_i' = (1, 1, \mathbf{r}_i)$. In this case, the change of the total cost of UE i is

$$\begin{aligned} \Delta C_i &= C_i\big(d_i', \mathbf{d}_{-i}\big) - C_i(d_i, \mathbf{d}_{-i}) \\ &= A^0 - \frac{R}{m(\mathbf{d}) + 1} + Q_i\Big(s_i^0, V_{\langle i,p(i)\rangle}\Big) \end{aligned}$$

and the change of the potential function is

$$\begin{aligned} \Delta\Phi &= \Phi\big(d_i', \mathbf{d}_{-i}\big) - \Phi(d_i, \mathbf{d}_{-i}) \\ &= -\sum_{i=1}^{m(\mathbf{d}')} \left(\frac{R}{i} - A^0\right) + \sum_{i=1}^{m(\mathbf{d})} \left(\frac{R}{i} - A^0\right) \\ &\quad + \sum_{i=1}^{n} Q_i\big(s_i', V_{\langle i,p(i)\rangle}\big) - Q_i\big(s_i, V_{\langle i,p(i)\rangle}\big) \\ &= A^0 - \frac{R}{m(\mathbf{d}) + 1} + Q_i\Big(s_i^0, V_{\langle i,p(i)\rangle}\Big). \end{aligned}$$

Thus, $\Phi(d_i', \mathbf{d}_{-i}) - \Phi(d_i, \mathbf{d}_{-i}) = C_i(d_i', \mathbf{d}_{-i}) - C_i(d_i, \mathbf{d}_{-i})$ holds in this case.

Case 2: The UE i changes from its previous decision $d_i = (0, 1, \mathbf{r}_i)$ to a better decision $d_i' = (0, 1, \mathbf{r}_i')$. In this case, the change of the total cost of UE i is

$$\begin{aligned} \Delta C_i &= C_i\big(d_i', \mathbf{d}_{-i}\big) - C_i(d_i, \mathbf{d}_{-i}) \\ &= \sum_{j\in\mathbf{r}_i'} \frac{p_j}{\big|N_j(\mathbf{d}')\big|} - \sum_{j\in\mathbf{r}_i} \frac{p_j}{\big|N_j(\mathbf{d})\big|} \\ &\quad + Q_i\Big(s_i^1, V_{\langle i,p'(i)\rangle}\Big) - Q_i\Big(s_i^1, V_{\langle i,p(i)\rangle}\Big) \end{aligned}$$

and the change of the potential function is

$$\begin{aligned} \Delta\Phi &= \Phi\big(d_i', \mathbf{d}_{-i}\big) - \Phi(d_i, \mathbf{d}_{-i}) \\ &= \sum_{i\in\mathcal{N}} \sum_{j=1}^{|N_i(\mathbf{d}')|} \frac{p_i}{j} - \sum_{i\in\mathcal{N}} \sum_{j=1}^{|N_i(\mathbf{d})|} \frac{p_i}{j} \\ &\quad + \sum_{i=1}^{n} Q_i\big(s_i, V_{\langle i,p'(i)\rangle}\big) - \sum_{i=1}^{n} Q_i\big(s_i, V_{\langle i,p(i)\rangle}\big) \\ &= \sum_{j\in\mathbf{r}_i'} \frac{p_j}{\big|N_j(\mathbf{d}')\big|} - \sum_{j\in\mathbf{r}_i} \frac{p_j}{\big|N_j(\mathbf{d})\big|} \\ &\quad + Q_i\Big(s_i^1, V_{\langle i,p'(i)\rangle}\Big) - Q_i\Big(s_i^1, V_{\langle i,p(i)\rangle}\Big). \end{aligned}$$

Thus, $\Phi(d_i', \mathbf{d}_{-i}) - \Phi(d_i, \mathbf{d}_{-i}) = C_i(d_i', \mathbf{d}_{-i}) - C_i(d_i, \mathbf{d}_{-i})$ also holds in this case.

Case 3: The UE i changes from its previous decision $d_i = (0, 1, \mathbf{r}_i)$ to a better decision $d_i' = (1, 1, \mathbf{r}_i')$. In this case, the change of the total cost of UE i is

$$\begin{aligned} \Delta C_i &= C_i(d_i', \mathbf{d}_{-i}) - C_i(d_i, \mathbf{d}_{-i}) \\ &= \sum_{j\in\mathbf{r}_i'} \frac{p_j}{\big|N_j(\mathbf{d}')\big|} - \sum_{j\in\mathbf{r}_i} \frac{p_j}{\big|N_j(\mathbf{d})\big|} + Q_i\big(s_i', V_{\langle i,p'(i)\rangle}\big) \\ &\quad - Q_i\Big(s_i^1, V_{\langle i,p(i)\rangle}\Big) + A^0 - \frac{R}{m(\mathbf{d}) + 1} \end{aligned}$$

and the change of the potential function is

$$\begin{aligned}\Delta\Phi &= \Phi\left(d_i', \mathbf{d}_{-i}\right) - \Phi(d_i, \mathbf{d}_{-i}) \\ &= \sum_{i\in\mathcal{N}}\sum_{j=1}^{|N_i(\mathbf{d}')|}\frac{p_i}{j} - \sum_{i\in\mathcal{N}}\sum_{j=1}^{|N_i(\mathbf{d})|}\frac{p_i}{j} - \sum_{i=1}^{m(\mathbf{d}')}\left(\frac{R}{i} - A^0\right) \\ &\quad + \sum_{i=1}^{m(\mathbf{d})}\left(\frac{R}{i} - A^0\right) + \sum_{i=1}^{n} Q_i\left(s_i, V_{\langle i,p'(i)\rangle}\right) \\ &\quad - \sum_{i=1}^{n} Q_i\left(s_i, V_{\langle i,p(i)\rangle}\right) \\ &= \sum_{j\in\mathbf{r}_i'}\frac{p_j}{\left|N_j(\mathbf{d}')\right|} - \sum_{j\in\mathbf{r}_i}\frac{p_j}{|N_j(\mathbf{d})|} + Q_i\left(s_i', V_{\langle i,p'(i)\rangle}\right) \\ &\quad - Q_i\left(s_i^1, V_{\langle i,p(i)\rangle}\right) + A^0 - \frac{R}{m(\mathbf{d})+1}.\end{aligned}$$

Thus, $\Phi(d_i', \mathbf{d}_{-i}) - \Phi(d_i, \mathbf{d}_{-i}) = C_i(d_i', \mathbf{d}_{-i}) - C_i(d_i, \mathbf{d}_{-i})$ also holds in this case. The proof for the other cases is similar and straightforward, demonstrating that the MCOG is an exact potential game. ■

Theorem 2: The MCOG possesses a pure-strategy NE.

Proof: We have proven that the MCOG is an exact potential game. Because each exact potential game admits a pure-strategy NE [38], the MCOG possesses a pure-strategy NE, which completes the proof of Theorem 2. ■

VI. Distributed Algorithm

In this section, we propose a distributed computation offloading algorithm shown in Algorithm 1, namely, the best update (BU) algorithm, with which the MCOG can reach an NE quickly. Besides, despite the existence of a controller that coordinates the signals, the major computing tasks are distributed.

Algorithm 1: Distributed Algorithm for the MCOG

1 **Initialization:**
2 each UE **sets** its decision $d_i = (0, 0, \mathbf{r}_i)$ and state to be **free**
3 **for** *each* $i \in \mathcal{N}$ **do**
4 **set** forwarding price p_i based on its transmission rate
5 **Iteration:**
6 **iterate** for each time slot t:
7 APs **broadcast** SM and CMs to the UEs of the next layer
8 **for** *each* $i \in \mathcal{N}$ **do**
9 **if** *i* ***receives*** *SM* **then**
10 **transmit** SM and CMs to the next layer
11 **if** *UE i is* ***free*** **then**
12 **compute** its best response set $\Upsilon_i(t)$
13 **if** $\Upsilon_i(t) \neq \varnothing$ **then**
14 **send** DM to contend the chance of update
15 **if** *i* ***receives*** *PM from the controller* **then**
16 **choose** a decision $d_i(t) \in \Upsilon_i(t)$ for slot t
17 **continue**
18 **keep** $d_i(t) = d_i(t-1)$ for slot t
19 **if** *the controller does not* ***receive*** *DM* **then**
20 **broadcast** EM to all UEs
21 **record** all transactions taking place in the network
22 **for** *each* $i \in \mathcal{N}$ **do**
23 **if** $N_i(\mathbf{d}(t)) \setminus O_i = \emptyset$ **then**
24 **set** UE i to be **free**
25 **else**
26 **set** UE i to be **locked**
27 **until** EM is got from the controller

A. Algorithm Design

Based on Theorem 1, because the MCOG is a potential game, it must have finite improvement paths [38]. We can use this property to design our distributed algorithm. Before the algorithm starts, each UE i needs to initialize its decision and state and set the forwarding price p_i based on its transmission rate (Lines 2–4). Next, for the synchronization of all UEs, we introduce a *time-slot mechanism* that enables the algorithm to be put into practice smoothly. Before each time slot begins, the controller broadcasts the starting message (SM) to all UEs (Line 7). In each time slot t, the following three steps are performed.

1) Message Exchanging Among UEs: To apply the BU algorithm to the real-world scenarios, each UE needs to send information about its transmission rate, forwarding price, and the number $|N_i(\mathbf{d})|$ to the UEs in the next layer after getting SM (Lines 9 and 10). We call these information congestion messages (CMs). After receiving enough CMs, each free UE computes the best decisions of the next slot based on the information, and each free UE needs to send its decision message (DM) to the controller for unified control, rather than directly updating their own decisions. The key point of the algorithm is that there is one and only one UE to change its decision in each time slot. Therefore, the controller needs to select one UE to update its decision and send a permit message (PM) to it. When no DM is received within a specified period of time, the controller broadcasts an ending message (EM) to all UEs, indicating that the NE is reached in the BU algorithm (Lines 19 and 20). For the sake of clarity, the sequence of the messages sent in each time slot is described in Fig. 2.

2) Decision Update (Lines 11–18): After receiving sufficient CMs, each free UE i computes its best response decision set with the formula as

$$\begin{aligned}\Upsilon_i(t) \triangleq \big\{d_i^* : d_i^* \text{ satisfies (14) and } \\ C_i\left(d_i^*, \mathbf{d}_{-i}(t-1)\right) < C_i(d_i(t-1), \mathbf{d}_{-i}(t-1))\big\}.\end{aligned}$$

Next, if UE i has at least one better decision to update (i.e., $\Upsilon_i(t) \neq \varnothing$), it sends a DM to the controller to contend for the update opportunity. Otherwise, UE i keeps its decision unchanged, i.e., $d_i(t) = d_i(t-1)$. Then, the controller randomly chooses a UE j that has sent a DM and sent a PM to it to update its decision at time slot t. Thus, UE j selects a decision from its best response decision set as its decision at the next time

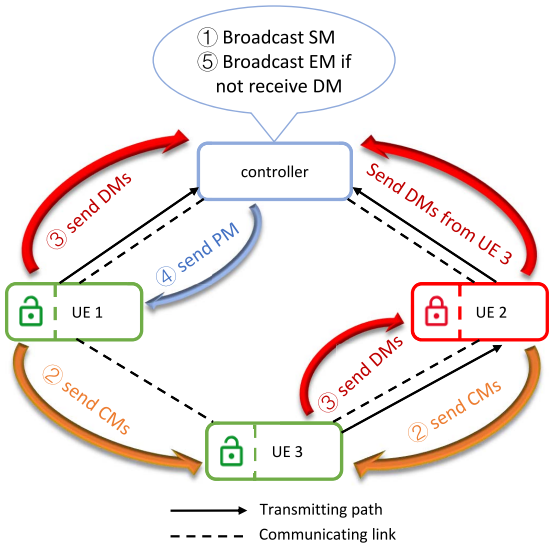


Fig. 2. Sequence of the messages sent in each time slot.

slot, i.e., $d_j(t) \in \Upsilon_j(t)$. As for the UEs that have already sent CM to the controller but have not got the PM, they keep their previous decisions, i.e., $d_i(t) = d_i(t-1)$.

3) State Update (Lines 22–26): At the end of each time slot t, the state of each UE needs to be updated. If $N_i(\mathbf{d}(t)) \backslash O_i = \emptyset$, UE i is set to be free. Otherwise, UE i is set to be locked.

Finally, each UE i pays the intermediate UEs in $\mathbf{r}_i$ and the ES based on its own decision. Then, all transactions occurring in the network are recorded into blockchain blocks so the transactional records cannot be tampered with (Line 21).

B. Performance Analysis

In this section, we analyze the performance of the NE in the MCOG by giving the bound of price of anarchy (PoA), which is a concept that measures the inefficiency of the NE in game theory. It is defined as the ratio of the worst system cost resulting from the NE in our game to the best system cost of the optimal result. Our goal is to minimize the total cost of the system, so a smaller system cost is desirable (i.e., a smaller PoA is desirable). In this paper, we consider the sum of the cost of all UEs as the system cost, i.e., $C(\mathbf{d}) = \sum_{i\in\mathcal{N}} C_i(\mathbf{d})$. Thus, the PoA in our game can be calculated as

$$\text{PoA} = \frac{\max_{\mathbf{d}^*} \sum_{i\in\mathcal{N}} C_i(\mathbf{d}^*)}{\min_{\mathbf{d}\in\mathbf{D}} \sum_{i\in\mathcal{N}} C_i(\mathbf{d})}.$$

Theorem 3: The multihop cooperative computation offloading game has a PoA that satisfies

$$\text{PoA} \le \frac{\sum_{i\in\Delta} C_i^{\text{local}} + \sum_{i\in\mathcal{N}\backslash\Delta} K_i}{\sum_{i\in\mathcal{N}} M_i}$$

where $K_i = \max\{C_i^{\text{local}}, \hat{C}_{i,\max}\} + \max_{\mathbf{d}\in\mathbf{D}} C_i^{\text{rel}}(\mathbf{d})$ and $M_i = \min\{C_i^{\text{local}}, \hat{C}_{i,\min}\} + \min_{\mathbf{d}\in\mathbf{D}} C_i^{\text{rel}}(\mathbf{d})$.

Proof: We set $\mathbf{d}^*$ as an arbitrary NE of the MCOG. To compute the worst system cost, we first consider the set of the UEs located in the outermost layers as Δ. For the UE i in Δ, $C_i(\mathbf{d}_i^*, \mathbf{d}_{-i}^*) \le C_i^{\text{local}}$ must be satisfied, and the worst case for the UEs in Δ is that they all compute their normal tasks using local computing resources and undertake no mining task. Because the UEs in the outermost layers have no relay tasks, i.e., they are free, they can change their decisions based on their own interests. As such, if $C_i(\mathbf{d}_i^*, \mathbf{d}_{-i}^*) > C_i^{\text{local}}$, UE i will change its decision to compute normal tasks locally and undertake no mining task, which contradicts the assumption that the decision profile $\mathbf{d}^*$ is an NE of the game. For the UEs not in Δ, they may be locked and help other UEs to transmit tasks. To compute the worst system cost, we suppose that each UE i not in Δ is locked and chooses the worst decision that makes its total cost maximal. Let $\hat{C}_{i,\max} \triangleq \max_{\mathbf{d}\in\mathbf{D}}\{C_i^1(\mathbf{d}) + C_i^0(\mathbf{d})\}$. Thus, $\max_{\mathbf{d}^*} \sum_{i\in\mathcal{N}\backslash\Delta} C_i(\mathbf{d}^*) \le \sum_{i\in\mathcal{N}\backslash\Delta} \max\{C_i^{\text{local}}, \hat{C}_{i,\max}\} + \max_{\mathbf{d}\in\mathbf{D}} C_i^{\text{rel}}(\mathbf{d})$.

Next, we compute the lower bound of the optimal system cost in the game. We consider $\tilde{\mathbf{d}} \in \mathbf{D}$ as the optimal decision in the game. If $C_i^{\text{local}} \le C_i(\mathbf{d})$ for $\forall \mathbf{d} \in \mathbf{D}$ and $i \in \mathcal{N}$, then the best decision of UE i is to compute normal tasks locally and undertake no mining tasks, i.e., $C_i(\tilde{\mathbf{d}}) = C_i^{\text{local}}$. Otherwise, UE i chooses a transmitting path $\mathbf{r}_i$ for offloading the computation to the ESs for lower total cost. In the best case, the NE in our game produces the minimum system cost. Let $\hat{C}_{i,\min} \triangleq \min_{\mathbf{d}\in\mathbf{D}}\{C_i^1(\mathbf{d}) + C_i^0(\mathbf{d})\}$. Thus, $\min_{\mathbf{d}\in\mathbf{D}} \sum_{i\in\mathcal{N}} C_i(\mathbf{d}) \le \sum_{i\in\mathcal{N}} \min\{C_i^{\text{local}}, \hat{C}_{i,\min}\} + \min_{\mathbf{d}\in\mathbf{D}} C_i^{\text{rel}}(\mathbf{d})$.

Therefore, according to the upper bound of the system cost resulting from the worst NE and the lower bound of the system cost resulting from the optimal decision, we can conclude that

$$\begin{aligned}\text{PoA} &= \frac{\max_{\mathbf{d}^*} \sum_{i\in\mathcal{N}} C_i(\mathbf{d}^*)}{\min_{\mathbf{d}\in\mathbf{D}} \sum_{i\in\mathcal{N}} C_i(\mathbf{d})} \\ &\le \frac{\sum_{i\in\Delta} C_i^{\text{local}} + \max_{\mathbf{d}^*} \sum_{i\in\mathcal{N}\backslash\Delta} C_i(\mathbf{d}^*)}{\sum_{i\in\mathcal{N}} \min\left\{C_i^{\text{local}}, \hat{C}_{i,\min}\right\} + \min_{\mathbf{d}\in\mathbf{D}} C_i^{\text{rel}}(\mathbf{d})} \\ &\le \frac{\sum_{i\in\Delta} C_i^{\text{local}} + \sum_{i\in\mathcal{N}\backslash\Delta} K_i}{\sum_{i\in\mathcal{N}} M_i}. \end{aligned} \tag{18}$$

■

VII. Numerical Results

A. Evaluation Metrics and Baseline Algorithms

In this section, we evaluate the performance of the proposed BU algorithm via extensive experiments. First, for the sake of performance evaluation, we consider three metrics as follows.

1) *System Cost:* The system cost is calculated as $C(\mathbf{d}) = \sum_{i\in\mathcal{N}} C_i(\mathbf{d})$. It is used as a metric to evaluate the performance of an algorithm. The smaller the system cost, the better the performance of the algorithm.
2) *Number of Iterations (or Time Slots):* The number of iterations is used to indicate the convergence rate and computational complexity of an algorithm. A smaller number of iterations means that the algorithm converges faster.
3) *Message Exchanging Volume:* The message exchanging volume is used to evaluate the communication loads of an algorithm. The less the message exchanging volume, the lower the communication loads of the algorithm, which means a better performance of the algorithm.

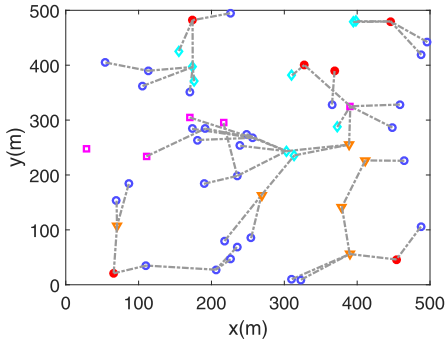


Fig. 3. Map showing the final decisions of the UEs.

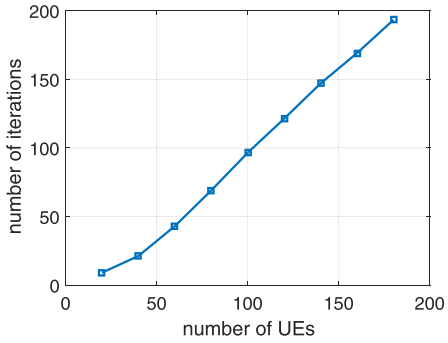


Fig. 4. Number of iterations versus the number of UEs.

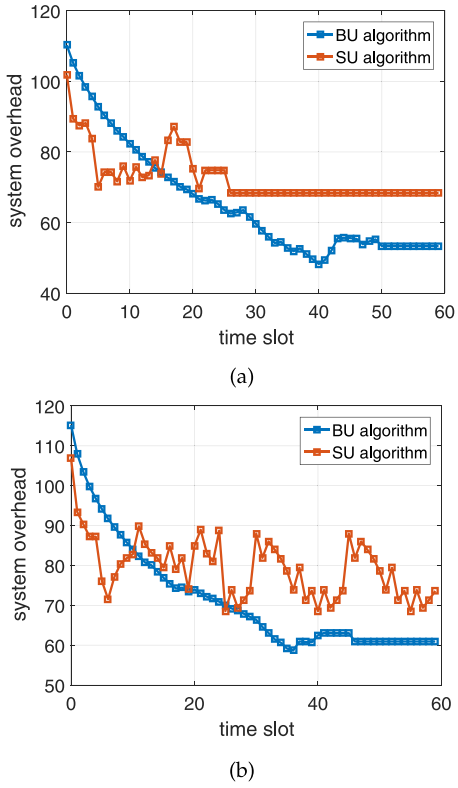


Fig. 5. (a) Dynamics of the system cost over time slot in the case where the SU algorithm converges. (b) Dynamics of the system cost over time slot in the case where the SU algorithm does not converge.

Second, we consider two other computation offloading algorithms for comparison.

1) *Synchronous Update (SU) Algorithm:* The SU algorithm optimizes each UE's cost considering both the normal tasks and the mining tasks, but this algorithm is unstable and may fail to converge in some cases. Differing from the SU algorithm adopted in [39], in this paper, we let the free UEs that have at least one better decision and locate in the same layer update their decisions simultaneously and perform the updates from layer 1 to layer $L_{\max}$. This process constitutes an iteration of the SU algorithm.
2) *Random Algorithm:* In the random algorithm, each UE randomly makes a decision that satisfies time restriction. Thus, the UEs in the random algorithm are irrational. In the experiments, we run the random algorithm 1000 times and take the average results for comparison.

B. Simulation Settings

Here, we consider a scenario where 6 APs and 50 UEs are randomly distributed in a region of 500 m $\times$ 500 m. Each UE and AP can communicate with each other within the range of 80 m. We suppose the channel bandwidth between UEs and APs or UEs is $B = 20$ MHz [18], while the data transmission power w_i is set to be 0.5 W. The Gaussian white noise $\varpi = -120$ dBm and the channel gain $G_n = d_{n,a}^{-\gamma}$, where $d_{n,a}$ is the distance between the UEs and APs or UEs, and $\gamma = 4$ is the path loss factor [40]. The data transmission rate from each AP to the ESs is set to be 1 Mb/s. For the computing resources supported by the ESs, we set $f^0 = 100$ GHz and $f^1 = 1000$ GHz. The computational capacity of UEs follows a uniform distribution on $[1.5, 2]$ GH.

For the mining tasks, the data size is set to be $s_i^0 = 5$ kB [18]. The first miner whose block achieves a consensus receives $k = 30$ tokens. Each miner should pay $p^s = 0.025$ tokens/Gcycle to the ESs. For the normal tasks, the data size s_i^1 and the number of CPU cycles l_i^1 are uniformly distributed in the range $[200, 400]$ kB and $[20, 40]$ Gcycles, respectively. Besides, T_i^1 is set to the completion time of computing the normal task locally. Each UE i sets its forwarding price $p_i = 0.002\overline{V}$ tokens, where $\overline{V}$ is the average transmission rate of UE i. Lastly, we consider $p^e = 0.1$ tokens/J [18].

C. Decision Simulation and Computational Complexity

With above simulation settings, the MCOG can achieve an NE by performing the BU algorithm. Fig. 3 simulates the final decision of each UE when the NE is achieved in the BU algorithm. The dashed lines in Fig. 3 denote the transmission paths chosen by each UE. The red points denote the APs in the region. The pink squares denote the UEs whose decisions are $d_i = (0, 0, \mathbf{r}_i)$, the blue circles denote the UEs whose decisions are $d_i = (0, 1, \mathbf{r}_i)$, the orange triangles denote the UEs whose decisions are $d_i = (1, 0, \mathbf{r}_i)$, and the blue-green diamonds denote the UEs whose decisions are $d_i = (1, 1, \mathbf{r}_i)$. Eventually, each UE plays the best response decision based on its own interests.

Then, we can use the number of iterations to show the computational complexity of the BU algorithm. Thus, in Fig. 4,

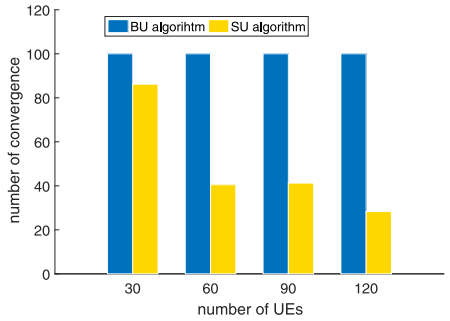


Fig. 6. Average number of successful convergences of the BU algorithm and the SU algorithm per 100 experiments.

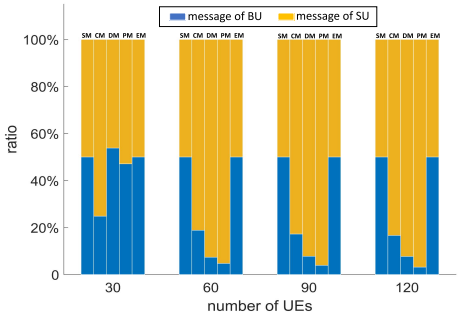


Fig. 7. Comparison of the message exchanging volume in the BU algorithm and that in the SU algorithm.

we plot the number of iterations versus the number of UEs. We run this experiment with $n = 20, 40, 60, \ldots, 180$ UEs and 6 APs. The results in Fig. 4 indicate that the number of iterations increases almost linearly with the number of UEs, demonstrating that the BU algorithm has relatively high efficiency.

D. Convergence Analysis

To demonstrate the convergence of the BU algorithm and the SU algorithm, we plot the dynamics of the system overhead over time slot in Fig. 5.

Fig. 5 shows that the system cost of the BU algorithm finally becomes constant because the system reaches to a stable point where no UEs can reduce their cost by changing their decisions unilaterally, i.e., the NE of the game, indicating that the BU algorithm has good convergence. In the SU algorithm, the free UEs in the same layer update their decisions simultaneously. In this manner, the cost functions of the free UEs cannot be mapped into the potential function, which is important for the property of the finite-time convergence of an iterated game toward an NE, leading to the fact that the SU algorithm may not converge as shown in Fig. 5(b).

Lastly, in order to further verify the convergence of the BU algorithm and the SU algorithm, the average successful convergence times of the BU algorithm and the SU algorithm per 100 experiments are recorded in Fig. 6. The BU algorithm always converges in the experiments, demonstrating that it has good convergence, while the average successful convergence times of the SU algorithm gradually decrease with an increased number of UEs because the mutual effects of decision updates become more and more serious.

E. Message Exchanging Volume

During the running of the BU algorithm and the SU algorithm, five kinds of messages are required to coordinate the decision updates of the UEs. We compare the exchanging volume of the SM, CM, DM, PM, and EM in the BU algorithm and the SU algorithm, respectively, and the results are recorded in Fig. 7. The height of each bar in Fig. 7 is 100%, the blue part in each bar is the proportion of the message exchanging volume of the BU algorithm, and the yellow part in each bar is the proportion of the message exchanging volume of the SU algorithm. Fig. 7 shows that the exchanging volume

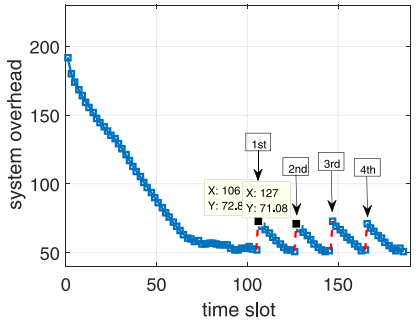


Fig. 8. BU algorithm adapts to the varying topology.

of the SM in the BU algorithm and that in the SU algorithm are equal, and so is the exchanging volume of the EM. In contrast, the exchanging volume of the other three kinds of messages in the BU algorithm is much less than that in the SU algorithm. Moreover, as the number of UEs increases, the SU algorithm requires more and more message exchanging than the BU algorithm.

In summary, the results in Fig. 7 demonstrate that the BU algorithm requires much less message exchanging compared with the SU algorithm and can keep a higher performance at the same time.

F. Topology-Varying Adaption

In a practical mobile scenario, the topology of the network changes over time with the movement of the UEs, so we need to run the proposed BU algorithm continuously. In order to evaluate the performance of our proposed algorithm when the topology changes, we conduct a simulation to imitate how our algorithm adapts to the varying topology.

We first execute the BU algorithm to achieve the NE of the game and then allow the UEs to move freely at a speed of 5 m/s for 2 s. The movement of the UEs may make the decisions of some UEs no longer optimal, so we run the BU algorithm again. We repeat this process for multiple times, and the system cost in this period is recorded in Fig. 8. Fig. 8 shows that the MCOG reaches to the NE point by running the BU algorithm after 106 time slots. Then, the system cost increases to 72.8 because of the first topology change, which is much smaller than the initial system cost of 190. By running the BU algorithm, the MCOG reaches to the NE

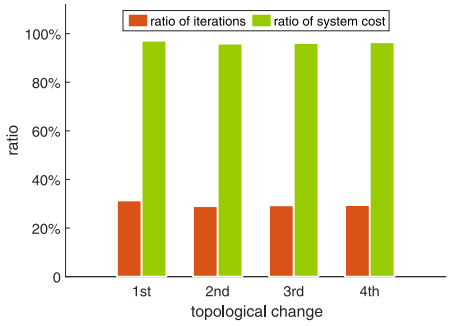


Fig. 9. Illustration of the universality of the phenomenon in Fig. 8.

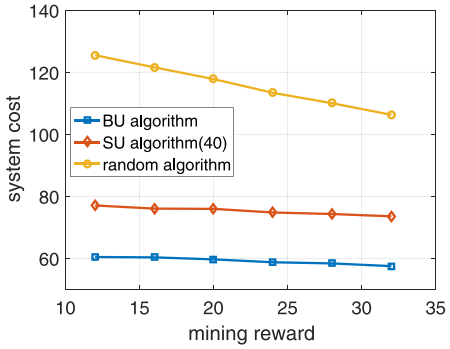


Fig. 10. System cost versus the mining reward.

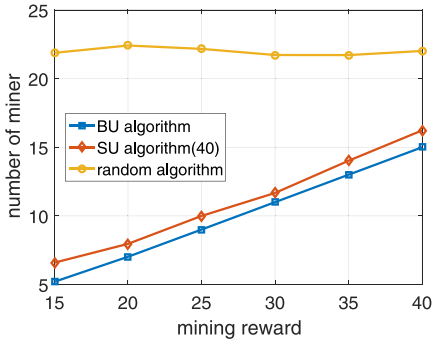


Fig. 11. Number of miners versus the mining reward.

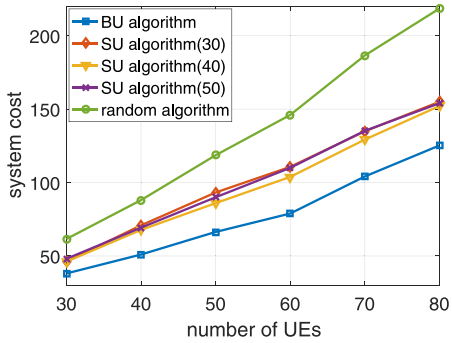


Fig. 12. System cost versus the number of UEs.

point again after 21 time slots, which is much faster than the first time. Moreover, Fig. 8 shows that the other three times topology change obtain similar results. In addition, the SU algorithm may not converge as we explained in Section VII-D, i.e., the existence of NE cannot be guaranteed. Therefore, the performance of the SU algorithm in the topology change scenario is incomparable to that of the BU algorithm, and we only study the topology-varying adaption for BU algorithm in this part.

In order to show the universality of the phenomenon in Fig. 8, we conducted the simulation in Fig. 8 repeatedly, and the average results are recorded in Fig. 9. In Fig. 9, the red bars denote the ratios of the number of iterations needed to achieve NE after the first, second, third, and fourth topology changes to that needed to achieve NE before changing the topology. The green bars denote the ratios of system cost at NE after the first, second, third, and fourth topology changes to that at NE before changing the topology. The figure shows that the heights of the red bars are about 30% and the heights of the green bars are about 100%. These results show that after changing the topology of the network, the BU algorithm can quickly reach to the NE again, and the change of the system cost at NE is very small.

In the conclusion, the BU algorithm can adapt to the varying topology quickly and maintain high performance in a practical mobile scenario.

G. Effects of Parameter Settings

Next, we compare the system cost of each algorithm with different mining rewards. Fig. 4 shows that when the number of UEs is set as 50, the number of iterations of the BU algorithm is around 40. Then, we also iterate the SU algorithm 40 times for comparison in this experiment. In Fig. 11, the number of miners in the BU and SU algorithms increases as the mining reward increases, and the tendencies of their curves are consistent. This relationship is because the UEs are rational in the BU algorithm and SU algorithm. In this manner, a greater mining reward allows more UEs to benefit from the mining tasks. Besides, as shown in Fig. 10, the system cost of the BU algorithm and the SU algorithm are much less than that of the random algorithm. The reason for this is that the UEs in the random algorithm are irrational, implying that the number of miners remains almost constant, as shown in Fig. 11. In Fig. 10, the BU algorithm averagely saves 36.34% of the system cost compared with the random algorithm, and 10.46% compared with the SU algorithm. It demonstrates that the BU algorithm has the lowest system cost because of the more rational improvement on the decision of each UE, and can effectively optimize the number of miners at the same time.

In some scenarios, the UEs in IIoT may execute different normal tasks. Therefore, we also investigate the effects of the number of UEs, the data size of the normal tasks, and the number of CPU cycles of the normal tasks on the system cost of each algorithm. Considering the SU algorithm may not converge, we iterate it 30, 40, and 50 times as comparative experiments. Here, the number of iterations of the BU algorithm is around 40. Then, we repeat each algorithm 1000 times and record the mean system cost. Fig. 12 shows that as the number of UEs increases, the system cost of each algorithm increases. While in the proposed BU algorithm, because of the game between different UEs, the increasing of the system

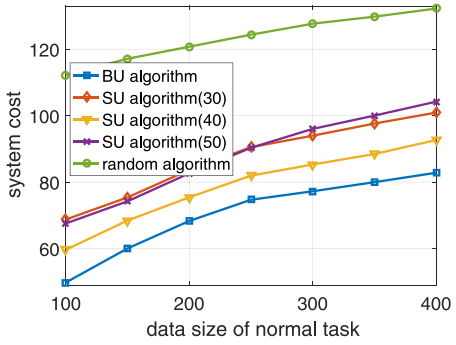


Fig. 13. System cost versus the data size of the normal tasks.

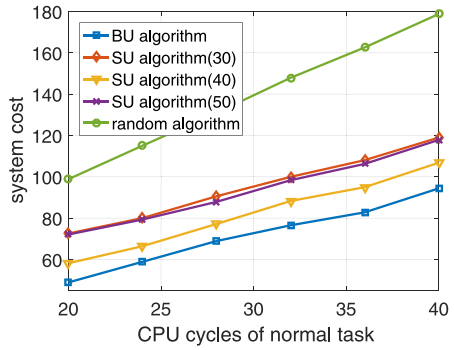


Fig. 14. System cost versus the number of CPU cycles of the normal tasks.

cost with the increasing of the number of UEs is slowest. Next, Fig. 13 shows that the system cost of each algorithm increases with the increasing of the data size of the normal tasks. This relationship is because transmitting larger normal tasks costs more energy. Lastly, Fig. 14 shows that the system cost increases when the number of CPU cycles of the normal tasks increases. This relationship is because of a higher cost of computing normal tasks with the limited on-board computing resources of UEs. In Figs. 12–14, the BU algorithm averagely saves 33.98%, 34.42%, and 36.79% of the system cost compared with the random algorithm, respectively, and 11.84%, 6.32%, and 6.65% of the system cost compared with the SU algorithm, respectively. It demonstrates that the proposed BU algorithm possesses the minimum system cost compared with that of other algorithms because the BU algorithm can ensure that the value of the function $\Phi(\mathbf{d})$ reduces continually in each time slot, while the SU algorithm and the random algorithm cannot.

H. Experiments in Real-World Scenarios

To demonstrate the practicality and applicability of the proposed framework and algorithm, we implement a prototype system for computation offloading that considers both the normal tasks and the mining tasks. The experiments use self-assembled hosts with Intel Core i7-6800K CPU as the ESs and a set of Raspberry Pi3 B as the devices in blockchain-empowered IIoT. The details are shown in Table II. Each device is equipped with a wireless network adapter, allowing them to communicate with each other. We also install the Geth client on each device to build the private blockchain.

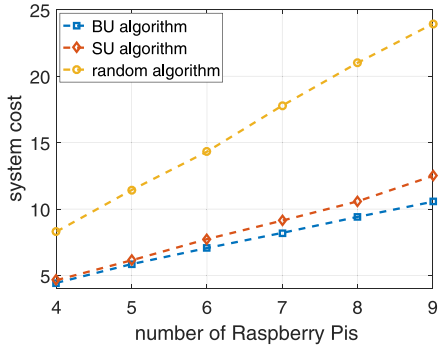


Fig. 15. System cost versus the number of Raspberry Pis.

TABLE II
SPECIFICATIONS OF THE DEVICES USED IN THE EXPERIMENTS

Devices Used	 Self-assembled host	 Raspberry Pi3 B
CPU	Intel(R) Core(TM) i7-6800K CPU	ARMv7 Processor rev 4 (v7l)
Cores	6 Physical Cores	4 Physical Cores
Clock Speed	4 GHz	1.2 GHz
RAM	112 GB	0.75 GB
Storage	1TB	16GB
Operating System	Linux version 4.15.0-36-generic	Linux version 4.14.71-v7+
Price	$ 2000	$ 36

Then, the transactions are recorded in the blockchain by using the Ethereum interface.

In our real experiments, each Pi performs a face recognition task with an input data size of approximately 120 kB and builds a private blockchain using Geth to record the transactions. We record the execution time of the face recognition tasks on each Pi and then estimate the energy consumption based on the approach in [35]. The ESs create an account for each miner and perform mining tasks using Geth. The experimental results are shown in Fig. 15, in which when the number of Raspberry Pis increases, the system cost of each algorithm increases. While in the proposed BU algorithm, because of the game between different Raspberry Pis, the increasing of the system cost with the increasing of the number of Raspberry Pis is slowest. Moreover, the system cost in the proposed BU algorithm under different numbers of Raspberry Pis remains smallest. In Fig. 15, the BU algorithm averagely saves 35.56% of the system cost compared with the random algorithm, and 5.73% compared with the SU algorithm. It demonstrates that our proposed BU algorithm is the most stable and can obtain the lowest system cost, indicating that our proposed algorithm can also work well in real-world scenarios.

VIII. CONCLUSION

In this paper, we have studied the multihop computation offloading problem that considers the normal tasks and the mining tasks together for blockchain-empowered IIoT. We formulate the offloading problem as an MCOG and prove the

existence of NE for the game. Then, we designed a high-efficiency distributed algorithm based on the finite improvement property of the MCOG. Lastly, our experimental results demonstrated that the proposed BU algorithm can converge to a stable state (i.e., an NE of the MCOG) quickly, and its number of iterations increases almost linearly with an increased number of UEs. The proposed BU algorithm require much less message exchanging comparing with other algorithms and can adapt to the varying topology quickly. Moreover, the performance of the proposed BU algorithm rises steadily with increasing mining rewards and can effectively optimize the number of miners at the same time. Furthermore, when the number of UEs, the data size of the normal tasks, and the number of CPU cycles of the normal tasks increase, the proposed BU algorithm remains the lowest system cost compared with other approaches. This result shows that our algorithm scales well as the number of UEs increases and possesses a relatively high efficiency under various parameter settings.

REFERENCES

[1] M. Aazam, S. Zeadally, and K. A. Harras, "Deploying fog computing in industrial Internet of Things and industry 4.0," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4674–4682, Oct. 2018.

[2] K.-K. R. Choo, S. Gritzalis, and J. H. Park, "Cryptographic solutions for industrial Internet-of-Things: Research challenges and opportunities," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3567–3569, Aug. 2018.

[3] L. Li *et al.*, "Creditcoin: A privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 7, pp. 2204–2220, Jul. 2018.

[4] O. Novo, "Blockchain meets IoT: An architecture for scalable access management in IoT," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1184–1195, Apr. 2018.

[5] C. Qiu, F. R. Yu, H. Yao, C. Jiang, F. Xu, and C. Zhao, "Blockchain-based software-defined industrial Internet of Things: A dueling deep Q-learning approach," *IEEE Internet Things J.*, to be published.

[6] Z. Su, Y. Wang, Q. Xu, M. Fei, Y.-C. Tian, and N. Zhang, "A secure charging scheme for electric vehicles with smart communities in energy blockchain," *IEEE Internet Things J.*, to be published.

[7] R. Li, T. Song, B. Mei, H. Li, X. Cheng, and L. Sun, "Blockchain for large-scale Internet of Things data storage and protection," *IEEE Trans. Services Comput.*, to be published.

[8] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[9] Y. Zhang, S. Kasahara, Y. Shen, X. Jiang, and J. Wan, "Smart contract-based access control for the Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1594–1605, Apr. 2019.

[10] Z. Li, J. Kang, R. Yu, D. Ye, Q. Deng, and Y. Zhang, "Consortium blockchain for secure energy trading in industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3690–3700, Aug. 2018.

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

[12] 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.

[13] Z. Yang, K. Yang, L. Lei, K. Zheng, and V. C. M. Leung, "Blockchain-based decentralized trust management in vehicular networks," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1495–1505, Apr. 2019.

[14] N. C. Luong, Z. Xiong, P. Wang, and D. Niyato, "Optimal auction for edge computing resource management in mobile blockchain networks: A deep learning approach," in *Proc. IEEE Int. Conf. Commun.*, Kansas City, MO, USA, May 2018, pp. 1–6.

[15] M. Li, L. Zhu, and X. Lin, "Efficient and privacy-preserving carpooling using blockchain-assisted vehicular fog computing," *IEEE Internet Things J.*, to be published.

[16] C. Xu, K. Wang, and M. Guo, "Intelligent resource management in blockchain-based cloud datacenters," *IEEE Cloud Comput.*, vol. 4, no. 6, pp. 50–59, Nov./Dec. 2017.

[17] D. Chatzopoulos, M. Ahmadi, S. Kosta, and P. Hui, "Flopcoin: A cryptocurrency for computation offloading," *IEEE Trans. Mobile Comput.*, vol. 17, no. 5, pp. 1062–1075, May 2018.

[18] M. Liu, R. 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.

[19] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint computation offloading and user association in multi-task mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12313–12325, Dec. 2018.

[20] G. Qiao, S. Leng, and Y. Zhang, "Online learning and optimization for computation offloading in D2D edge computing and networks," *Mobile Netw. Appl.*, pp. 1–12, Jan. 2019.

[21] F. Sun *et al.*, "Cooperative task scheduling for computation offloading in vehicular cloud," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11049–11061, Nov. 2018.

[22] H. Shah-Mansouri and V. W. S. Wong, "Hierarchical fog-cloud computing for IoT systems: A computation offloading game," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 3246–3257, Aug. 2018.

[23] J. Zheng, Y. Cai, Y. Wu, and X. S. Shen, "Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach," *IEEE Trans. Mobile Comput.*, vol. 18, no. 4, pp. 771–786, Apr. 2019.

[24] H. Guo, J. Zhang, J. Liu, and H. Zhang, "Energy-aware computation offloading and transmit power allocation in ultra-dense IoT networks," *IEEE Internet Things J.*, to be published.

[25] Y. Zhang, B. Feng, W. Quan, G. Li, H. Zhou, and H. Zhang, "Theoretical analysis on edge computation offloading policies for IoT devices," *IEEE Internet Things J.*, to be published.

[26] H. Guo, J. Liu, J. Zhang, W. Sun, and N. Kato, "Mobile-edge computation offloading for ultradense IoT networks," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4977–4988, Dec. 2018.

[27] Y. Chen, N. Zhang, Y. Zhang, and X. Chen, "Dynamic computation offloading in edge computing for Internet of Things," *IEEE Internet Things J.*, to be published.

[28] H. Al-Shatri, S. Müller, and A. Klein, "Distributed algorithm for energy efficient multi-hop computation offloading," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kuala Lumpur, Malaysia, May 2016, pp. 1–6.

[29] Z. Hong, H. Huang, S. Guo, W. Chen, and Z. Zheng, "QoS-aware cooperative computation offloading for robot swarms in cloud robotics," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 4027–4041, Apr. 2019.

[30] Y. Zhu, G. Zheng, L. Wang, K.-K. Wong, and L. Zhao, "Content placement in cache-enabled sub-6 GHz and millimeter-wave multi-antenna dense small cell networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 5, pp. 2843–2856, May 2018.

[31] Y. Jiao, P. Wang, D. Niyato, and Z. Xiong, "Social welfare maximization auction in edge computing resource allocation for mobile blockchain," in *Proc. IEEE Int. Conf. Commun.*, Kansas City, MO, USA, 2018, pp. 1–6.

[32] Z. Li, Z. Yang, and S. Xie, "Computing resource trading for edge-cloud-assisted Internet of Things," *IEEE Trans. Ind. Informat.*, to be published.

[33] J. Moura and D. Hutchison, "Game theory for multi-access edge computing: Survey, use cases, and future trends," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 260–288, 1st Quart., 2019.

[34] U. N. Kar and D. K. Sanyal, "An overview of device-to-device communication in cellular networks," *ICT Exp.*, vol. 4, no. 4, pp. 203–208, 2017.

[35] Y. Wen, W. Zhang, and H. Luo, "Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones," in *Proc. IEEE INFOCOM*, Orlando, FL, USA, 2012, pp. 2716–2720.

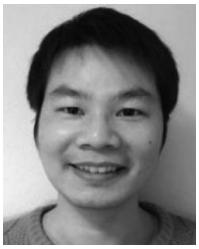
[36] S. Jošilo and G. Dán, "A game theoretic analysis of selfish mobile computation offloading," in *Proc. IEEE INFOCOM*, Atlanta, GA, USA, 2017, pp. 1–9.

[37] Z. Xiong, S. Feng, D. Niyato, P. Wang, and Z. Han, "Optimal pricing-based edge computing resource management in mobile blockchain," in *Proc. IEEE Int. Conf. Commun.*, Kansas City, MO, USA, 2018, pp. 1–6.

[38] D. Monderer and L. S. Shapley, "Potential games," *Games Econ. Behav.*, vol. 14, no. 1, pp. 124–143, 1996.

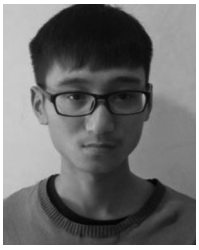
[39] W. Saad, Q. Zhu, T. Basar, Z. Han, and A. Hjorungnes, "Hierarchical network formation games in the uplink of multi-hop wireless networks," in *Proc. IEEE GLOBECOM*, Honolulu, HI, USA, 2009, pp. 1–6.

[40] T. S. Rappaport *et al.*, *Wireless Communications: Principles and Practice*. Upper Saddle River, NJ, USA: Prentice-Hall, 1996.



Wuhui Chen (M'15) received the bachelor's degree from Northeast University, Shenyang, China, in 2008, and the master's and Ph.D. degrees from the University of Aizu, Aizuwakamatsu, Japan, in 2011 and 2014, respectively.

From 2014 to 2016, he was a Research Fellow with the Japan Society for the Promotion of Science, Tokyo, Japan. From 2016 to 2017, he was a Researcher with the University of Aizu. He is currently an Associate Professor with Sun Yat-Sen University, Guangzhou, China. His current research interests include edge/cloud computing, cloud robotics, and blockchain.



Zhen Zhang received the B.S. degree from Tianjin Polytechnic University, Tianjin, China, in 2018. He is currently pursuing the M.S. degree at the School of Data and Computer Science, Sun Yat-Sen University, Guangzhou, China.

His current research interests include blockchain, Internet of Things, edge computing, and game theory.

Zicong Hong is currently pursuing the B.Eng. degree in software engineering at the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China.

His current research interests include game theory, Internet of Things, blockchain, and edge/cloud computing.

Chuan Chen (M'18) received the B.S. degree from Sun Yat-sen University, Guangzhou, China, in 2012 and the Ph.D. degree from Hong Kong Baptist University, Hong Kong, in 2016.

He is currently an Associate Research Fellow with the School of Data and Computer Science, Sun Yat-Sen University. His current research interests include machine learning, numerical linear algebra, and numerical optimization.

Jiajing Wu (M'15) received the B.Eng. degree in communication engineering from Beijing Jiaotong University, Beijing, China, in 2010, and the Ph.D. degree from Hong Kong Polytechnic University, Hong Kong, in 2014.

In 2015, she joined the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China, where she is currently an Assistant Professor. Her current research interests include network science and its applications in engineering networked systems, such as communication networks, power grids, and cyber-physical systems.

Dr. Wu was a recipient of the Hong Kong Ph.D. Fellowship Scheme during her Ph.D. study in Hong Kong from 2010 to 2014. She is serving as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS–II: EXPRESS BRIEFS.

Sabita Maharjan (M'09) received the Ph.D. degree in networks and distributed systems from the Simula Research Laboratory, University of Oslo, Oslo, Norway, in 2013.

She is currently a Senior Research Scientist with the Simula Metropolitan Center for Digital Engineering, Oslo, and an Associate Professor with the University of Oslo. Her current research interests include wireless networks, network security and resilience, smart grid communications, Internet of Things, machine-to-machine communications, software-defined wireless networking, and Internet of Vehicles.

Zibin Zheng (SM'17) is currently a Professor with the School of Data and Computer Science, Sun Yat-Sen University, Guangzhou, China. His current research interests include service computing and cloud computing.

Mr. Zheng was a recipient of the Outstanding Ph.D. Thesis Award of the Chinese University of Hong Kong in 2012, the Association for Computing Machinery's Special Interest Group on Software Engineering Distinguished Paper Award of the International Conference on Science and Engineering in 2010, the Best Student Paper Award of the International Conference on Web Services in 2010, and the IBM Ph.D. Fellowship Award in 2010. He served as a Program Committee member of the IEEE International Conference on Cloud Computing, the International Conference on Web Services, the International Conference on Service Computing, the International Conference on Service-Oriented Computing, and the International Symposium on Service-Oriented System Engineering.

Yan Zhang (M'05–SM'10) received the Ph.D. degree from the School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore.

He is a Full Professor with the Department of Informatics, University of Oslo, Oslo, Norway. His current research interests include next-generation wireless networks leading to 5G, green and secure cyber-physical systems (e.g., smart grid, healthcare, and transport).

Dr. Zhang was a recipient of the Highly Cited Researcher Award (top 1% by citations) by Clarivate Analytics in 2018. He is an Associate Technical Editor of *IEEE Communications Magazine*, an Editor of the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING, the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, and the IEEE INTERNET OF THINGS JOURNAL, and an Associate Editor of IEEE ACCESS. He has served as the Chair for a number of conferences, including IEEE GLOBECOM 2017, IEEE VTC-Spring 2017, IEEE PIMRC 2016, IEEE CloudCom 2016, IEEE ICCC 2016, IEEE CCNC 2016, IEEE SmartGridComm 2015, and IEEE CloudCom 2015. He has served as a TPC member for numerous international conference, including IEEE INFOCOM, IEEE ICC, IEEE GLOBECOM, and IEEE WCNC. He is an IEEE Vehicular Technology Society (VTS) Distinguished Lecturer. He is also a Senior Member of IEEE ComSoc, IEEE CS, IEEE PES, and IEEE VTS. He is a Fellow of the IET.