Cooperative Computation Offloading and Resource Allocation for Blockchain-Enabled Mobile Edge Computing: A Deep Reinforcement Learning Approach

Jie Feng, F. Richard Yu, *Fellow, IEEE,* Qingqi Pei, Xiaoli Chu, Jianbo Du, and Li Zhu

***Abstract*—Mobile edge computing (MEC) is a promising paradigm to improve the quality of computation experience of mobile devices because it allows mobile devices to offload computing tasks to MEC servers, benefiting from the powerful computing resources of MEC servers. However, the existing computation-offloading works have also some open issues: 1) security and privacy issues, 2) cooperative computation offloading, and 3) dynamic optimization. To address the security and privacy issues, we employ blockchain technology that ensures the reliability and irreversibility of data in MEC systems. Meanwhile, we jointly design and optimize the performance of blockchain and MEC. In this paper, we develop a cooperative computation offloading and resource allocation framework for blockchain-enabled MEC systems. In the framework, we design a multi-objective function to maximize the computation rate of MEC systems and the transaction throughput of blockchain systems by jointly optimizing offloading decision, power allocation, block size and block interval. Due to the dynamic characteristics of the wireless fading channel and the processing queues at MEC servers, the joint optimization is formulated as a Markov decision process (MDP). To tackle the dynamics and complexity of the blockchain-enabled MEC system, we develop an A3C-based cooperation computation offloading and resource allocation algorithm to solve the MDP problem. In the algorithm, deep neural networks are optimized by utilizing asynchronous gradient descent and eliminating the correlation of data. Simulation results show that the proposed algorithm converges fast and achieves significant performance improvements over existing schemes in terms of total reward.**

***Index Terms*—Mobile edge computing, blockchain, computation offloading, transaction throughput, A3C.**

This work is supported by the National Key Research and Development Program of China under Grant 2018YFE0126000, the Key Program of NSFC-Tongyong Union Foundation under Grant U1636209, the National Natural Science Foundation of China under Grant 61902292, and the Key Research and Development Programs of Shaanxi under Grant 2019ZDLGY13-07 and 2019ZDLGY13-04, the Natural Science Foundation of China under Grant 61901367, and the Doctoral Student's Short-Term Study Abroad Scholarship Fund of Xidian University. (*Corresponding authors: Qingqi Pei and F. Richard Yu*)

Jie Feng and Qingqi Pei are with State Key Laboratory of Integrated Services Networks (Xidian University), School of Telecomm. Engineering, Xidian University, Xi'an, Shaanxi, China (email: jiefeng1228@gmail.com; qqpei@mail.xidian.edu.cn).

F. Richard Yu is with the Dept. of Systems and Computer Eng., Carleton University, Ottawa, ON, Canada (email: Richard.Yu@carleton.ca).

Xiaoli Chu is with University of Sheffield, S1 3JD, UK (email: x.chu@sheffield.ac.uk).

Jianbo Du is with Shaanxi Key Laboratory of Information Communications, Xi'an University of Posts and Telecommunications, Xi'an, Shaanxi, China (email: dujianboo@163.com).

L. Zhu is Beijing Jiaotong University Beijing, P.R. China (email: zhulibjtu@gmail.com).

I. Introduction

Mobile edge computing (MEC) is a promising technology that can promote the computation capability of mobile devices by offloading the computing tasks from mobile devices to MEC servers [1]. Compared with the centralized cloud computing system, the distributed structure of MEC systems has many advantages, including reduced energy consumption and decreased latency. Many efforts have been made on computation offloading and resource allocation of MEC systems [2]–[5]. However, the above existing methods are not suitable for some specific environments because of the following challenges.

1) *Security and Privacy Issues*: Most of the existing studies [6], [7] pay little attention to the security and privacy of MEC. The interaction between heterogeneous edge nodes and the migration of service across edge nodes are likely to challenge its security and privacy. To address these issues, *blockchain* has been envisioned as a promising approach [8]. Different from traditional digital ledger approaches, which depend on a trusted central authority, blockchain employs community verification to synchronize the decentralized ledgers that are replicated across multiple nodes [9]. Blockchain can facilitate the establishment of a trusted, secure, and decentralized MEC systems. In blockchain-enabled MEC systems, MEC servers not only handle their tasks but also deal with the task (e.g. generate blocks and perform consensus process) from blockchain systems, which makes the design of the system more complex. Therefore, the design and optimization of blockchain and MEC should be implemented simultaneously.

2) *Cooperative Computation Offloading*: This approach has been only considered by a few researchers in the previous works. Most existing computation offloading schemes [6], [7] assume that computing tasks can be directly offloaded to MEC servers via wireless communications. However, a mobile device may be experiencing weak or intermittent connectivity and thus cannot directly offload computing tasks to MEC servers. If computing tasks are forced to offload to MEC servers directly, the computation offloading performance of mobile devices may be affected due to signal loss. A mobile device must offload computing tasks to MEC servers with the help of neighbouring nodes. Therefore, it is necessary to study cooperative computation offloading. Furthermore, if there exist malicious nearby nodes, the data security and privacy of

mobile devices will be susceptible to attacks. Therefore, the trust model needs to be considered on cooperative computation offloading.

3) *Dynamic Optimization*: Moreover, most of the existing works [6], [10], [11] in the computation offloading decision and resource allocation strategies are optimized based on a one-time slot, and the long-term computation offloading performance cannot be characterized [12], [13]. The design and optimization of blockchain-enabled MEC systems should account for the environmental dynamics, such as the time-vary channel conditions and the task arrival.

To deal with the first two challenges, in this paper, we propose to maximize the weighted sum of the computation rate and the transaction throughput for blockchain-enabled MEC systems by jointly optimizing the cooperative computation offloading decision and resource allocation. Specifically, the computation tasks are offloaded from mobile devices to MEC servers through cooperative communications, wherein blockchain technology is applied to guarantee data security. For the dynamic optimization issue, we formulate the joint optimization as a Markov decision process (MDP) problem, and develop an efficient deep reinforcement learning (DRL) based offloading decision and resource allocation algorithm to solve the problem.

The contributions of this paper are summarized as follows.

- In most existing works [10], [11], [14], [15], the design and optimization of blockchain and MEC are done separately, which will result in sub-optimal performance. We propose a cooperative computation offloading framework for blockchain-enabled MEC systems to enable the joint analysis of the MEC computation rate and the blockchain transaction throughput while considering the trust model.
- The study jointly considers the offloading decision, power allocation, block size, and block interval to maximize the weighted sum of computation rate of MEC systems and transaction throughput of blockchain systems. Considering the dynamic characteristics of wireless channels and the available resources, the optimization problem is formulated as an MDP. Since the action space of the MDP problem has both continuous actions and discrete actions, traditional learning algorithms, such as Q-learning [16] and SARSA [17] and so on, are powerless. An asynchronous advantage actor-critic (A3C) reinforcement learning algorithm is introduced to solve the MDP problem, in which deep neural networks are optimized by using asynchronous gradient descent and eliminating the correlation of data.
- The proposed algorithm and other baseline functions are implemented by using Tensorflow on a Python-based simulator. Extensive simulation results show that the proposed algorithm has good convergence, and has significant performance improvements over existing algorithms. Furthermore, we observe that the proposed scheme can achieve the optimal trade-off between the performance of the MEC system and the blockchain system.

The rest of this paper is organized as follows. In Section II, we discuss related research. We introduce the system model in Section III. In Section IV, the trust calculation in blockchain-enabled MEC systems is described. In Section V, the joint problem of offloading decision and resource allocation is formulated. We introduce the offloading decision and resource allocation in A3C framework in Section VI. The performance of the proposed algorithm is evaluated and analyzed by simulations in Section VII. Finally, in Section VIII, we conclude this paper and look forward to future work.

II. Related Work

The cooperative computation offloading problem has been widely studied for MEC and cloud computing systems [18]. Hong *et al.* [19] proposed a quality of service (QoS) cooperative computation offloading problem for robots swarms in clouding systems aimed at minimizing latency. Cao *et al.* [20] studied a novel cooperative computation offloading based on both computation and communication of MEC systems to improve the energy efficiency for latency-constrained computation. Guo *et al.* [21] presented an efficient dynamic offloading and resource scheduling strategy to decrease energy consumption and latency. However, these approaches do not take into account the security and privacy of data in cooperative computation offloading.

The application of the blockchain in the MEC systems can significantly improve the network security, data integrity and computation validity of the system [22]. The computation offloading problem has also been studied for the blockchain-enabled MEC system [11], [14]. Liu *et al.* [10] proposed a novel blockchain-based framework with an adaptive block size in MEC systems, which considered two offloading models, i.e., offloading to MEC servers or nearby device-to-device users. Kang *et al.* [23] proposed a secure and distributed vehicular blockchain for data management in vehicular edge computing and networks. Based on the common decentralization characteristic of MEC and blockchain technology, Xu *et al.* [24] proposed a trustless crowd-intelligence ecosystem to improve network congestion. However, these works only consider blockchain as an overlay system above the MEC system, which will give rise to sub-optimal performance. Furthermore, these approaches utilize static optimization techniques, which cannot characterize the long-term computation offloading performance. Therefore, their methods cannot be applied in practical dynamic systems.

Deep reinforcement learning (DRL) is emerging as one of the efficient methods to obtain the optimal decision-making policy and maximize long-term rewards. Therefore, the use of DRL to solve computation offloading problems for blockchain-enabled MEC systems has attracted considerable interest from academia. Qiu *et al.* [25] proposed a model-free DRL-based computation offloading scheme for blockchain-enabled MEC systems while considering mining tasks and data processing tasks. A computation offloading problem based on an advanced deep Q-learning network (DQN) was presented to minimize energy consumption and delay [26]. However, all of the above-proposed algorithms can only handle discrete action space and do not apply to continuous action cases. Therefore, we develop an A3C-based cooperative computation offloading and resource allocation algorithm to achieve the optimal trade-off between the performance of the MEC system and the

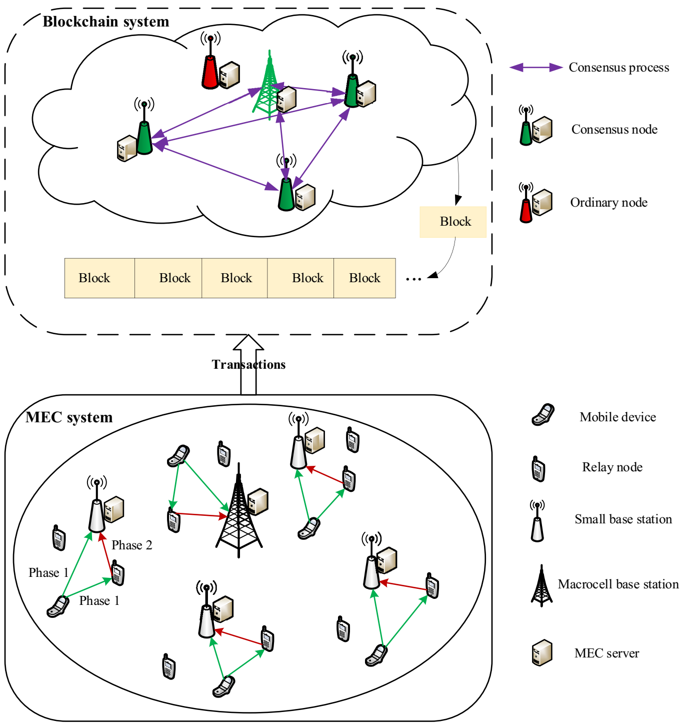


Fig. 1: The system model.

performance of the blockchain system while considering the trust model.

III. System model

In this section, we first present the network model, then depict the MEC model and the blockchain model in detail, respectively.

A. Network Model

We consider a blockchain-enabled MEC system, as shown in Fig. 1, which is composed of an MEC system and a blockchain system. In the MEC system, a single macrocell base station (MBS) is located in the center of the coverage area. Several small base stations (SBS) are distributed around the MBS, and all BSs are connected by wire links, each of which is integrated with an MEC server. We consider an interference-free system in the paper, which users utilize orthogonal spectrums for data transmission [27]. Let $\mathcal{N} = \{1, 2, ..., N\}$ denote the set of BSs. We assume that each BS only serves one mobile device [28]. However, the scenario in which each BS servers multiple mobile devices will be discussed in our future work. Therefore, the set of mobile devices is denoted by $\mathcal{N}^* = \mathcal{N} = \{1, 2, ..., N\}$, where mobile device n is served by its corresponding BS n. We assume that each mobile device is running independent and fine-grained tasks [2], [29]. Since mobile devices have relatively weak computing capability, the computing tasks of mobile devices need to be completed with the help of MEC servers to improve the quality of the user's computation experience.

We adopt cooperative communications to offload tasks in the MEC system, i.e., offloading computation tasks with the help of relay nodes, to meet the computation requirements of mobile devices that are far from MEC servers. Assume that there are R relay nodes around each mobile device, and each mobile device can only select one relay node to offload computation tasks collaboratively. Let $\mathcal{R} = \{1, 2, ...R\}$ denote the set of relay nodes within the coverage of each BS. When offloading computation tasks by relaying, there may be selfish and malicious nodes. Therefore, security plays an important role in realizing cooperative communications [30]. In this paper, we consider a trust-based secure computation offloading scheme. Let $D_{n\to r}^{trust}$ denote the trust value of mobile device n to relay node r.

In the blockchain system, the blockchain nodes consist of all BSs. These nodes have two types of roles: ordinary nodes and consensus nodes. The blockchain system mainly deals with transactions, i.e., offloading data records, from the MEC network. To handle the transactions, the blockchain system needs to complete two steps. One is the block generation, and the other is the consensus process. Ordinary nodes only transfer and accept ledger data, while consensus nodes produce blocks and perform the consensus process. However, there may be a security issue during the block generation process, i.e., malicious consensus nodes may tamper with transaction data. Therefore, the trust value of each candidate should be considered when voting for consensus nodes. Consensus nodes with high trust value are likely to ensure a secure and reliable block generation process and consensus process [14]. Meanwhile, the blockchain system can also store some parameters for calculating the trust value of the interactive nodes (i.e., relay nodes and consensus nodes), such as network status, resources availability, and trustworthiness of interactive nodes [31]. Note that there are K consensus nodes selected out of $\mathcal{N}$ according to a certain rule (specified in Subsection II-C). Let $\mathcal{K} = \{1, 2, ...K\}$ denote the set of consensus nodes. Similar to many previous works [2], [15], time is slotted in this paper and the length of a time slot is Δt. All the notions used are listed in Table I.

B. MEC System

1) *Local Computing Mode*: Let L_n denote the number of CPU cycles required for mobile device n to process 1-bit computing task, which is determined by the types of applications and can be procured by off-line measurements [32]. We let f_n denote the CPU-cycle frequency of mobile device n, which must meet the constraint $f_n \leq f_n^{max}$, by using dynamic voltage and frequency scaling (DVFS). Let t_{loc} $(0 \leq t_{loc} \leq \triangle t)$ denote the computing time of mobile device. The computation rate for local computation (in bits per second), denoted by r_{locl}, can be given by

$$r_{locl} = \frac{f_n}{L_n}. \tag{1}$$

2) *Cooperative Offloading Mode*: Cooperative offloading is composed of two phases, as shown in Fig. 1. In the first phase, the mobile device transmits wireless signals that contain the offloaded data size to the BS, which is simultaneously overheard by a relay node. The selected relay node forwards the detected signals to BS by employing the regenerate and

TABLE I: Notation Definitions

Symbol	Definition
$\mathcal{N}^*/\mathcal{N}$	The set of all mobile devices or BSs
$\mathcal{R}$	The set of all relay nodes
$D_{n\to r}^{trust}$	The trust value of mobile device n to relay node r
$\mathcal{K}$	The set of the block producers
$P_n(t)$	The transmit power of mobile device n in slot t
$P_{r,n}(t)$	The transmit power of relay node r to BS n in slot t
$g_n(t)$	The channel gain of mobile device n to BS n in slot t
$g_{n,r}(t)$	The channel gain of mobile device n to relay node r in slot t
$g_{r,n}(t)$	The channel gain of relay node r to BS n in slot t
ψ_n	The delay of mobile device n in offloading data
C_n	The processing density
s_n	The CPU-cycle frequency
F'	The total computation capability of the MEC server
$\sigma_n(t)$	The noise variances of mobile device n to BS n in slot t
$\sigma_{n,r}(t)$	The noise variances of mobile device n to relay node r in slot t
$\sigma_{r,n}(t)$	The noise variances of relay node n to BS n in slot t
τ_n	The tolerable maximum delay
T_{min}	The minimum computing capacity required by the blockchain system
f_n^{max}	The maximum CPU-cycle frequency of mobile device n
$S_b(t)$	The block size in slot t
$T_b(t)$	The block interval in slot t

forward scheme. The offloaded date received in both two phases is combined at the BS using maximal ratio combining (MRC) [33]. The transmit rate of mobile device n when the relay node r is selected in time slot t is expressed as

$$R_n(t) = \frac{1}{2}\min\left\{\log_2(1+\frac{P_n(t)g_n(t)}{\sigma_n^2(t)}), \log_2(1+\frac{P_n(t)g_{n,r}(t)}{\sigma_{n,r}^2(t)}+\frac{P_{r,n}(t)g_{r,n}(t)}{\sigma_{r,n}^2(t)})\right\}, \quad (2)$$

where $P_n(t)$ and $P_{r,n}(t)$ are the transmit power of mobile device n and relay node r to BS n in time slot t, respectively. $g_n(t)$, $g_{n,r}(t)$, and $g_{r,n}(t)$ are the channel gain between mobile device n and BS n, mobile device n and relay node r and relay node r and BS n in slot t, respectively.

The total power of the mobile device and relay node in BS n is given by

$$P_{tot,n}(t) = P_n(t) + P_{r,n}(t). \quad (3)$$

When the direct channel conditions are less than the relay channel conditions, i.e., $g_n(t)/\sigma_n^2(t) < g_{n,r}(t)/\sigma_{n,r}^2(t)$, the cooperative computation offloading is adopted. In the cooperative computation offloading, any increase of power has to shared between the mobile device and relay node. Therefore, the transmit rate can reach the maximum when the following equation is satisfied.

$$\frac{P_n(t)g_n(t)}{\sigma_n^2(t)}+\frac{P_{r,n}(t)g_{r,n}(t)}{\sigma_{r,n}^2(t)} = \frac{P_n(t)g_{n,r}(t)}{\sigma_{n,r}^2(t)}. \quad (4)$$

According to (3), the transmit power of mobile device n and relay node r to BS n is respectively given by

$$P_n(t) = \frac{g_{r,n}(t)P_{tot,n}(t)/\sigma_{r,n}^2(t)}{g_{n,r}(t)/\sigma_{n,r}^2(t)+g_{r,n}(t)/\sigma_{r,n}^2(t)-g_n(t)/\sigma_n^2(t)}, \quad (5)$$

$$P_{r,n}(t) = \frac{(g_{n,r}(t)/\sigma_{n,r}^2(t)-g_n(t)/\sigma_n^2)P_{tot,n}(t)}{g_{n,r}(t)/\sigma_{n,r}^2(t)+g_{r,n}(t)/\sigma_{r,n}^2(t)-g_n(t)/\sigma_n^2(t)}. \quad (6)$$

Substituting (5) and (6) to (2), we can have

$$\frac{P_n(t)g_n(t)}{\sigma_n^2(t)} = \frac{g_{r,n}(t)g_n(t)P_{tot,n}(t)/(\sigma_{r,n}^2(t)\sigma_n^2(t))}{g_{n,r}(t)/\sigma_{n,r}^2(t)+g_{r,n}(t)/\sigma_{r,n}^2(t)-g_n(t)/\sigma_n^2(t)}. \quad (7)$$

$$\frac{P_n(t)g_{n,r}(t)}{\sigma_{n,r}^2(t)}+\frac{P_{r,n}(t)g_{r,n}(t)}{\sigma_{r,n}^2(t)} = \frac{(g_{r,n}(t)/\sigma_{r,n}^2(t))(2g_{n,r}(t)/\sigma_{n,r}^2(t)-g_n(t)/\sigma_n^2)P_{tot,n}(t)}{g_{n,r}(t)/\sigma_{n,r}^2(t)+g_{r,n}(t)/\sigma_{r,n}^2(t)-g_n(t)/\sigma_n^2(t)}. \quad (8)$$

Since $g_n(t)/\sigma_n^2(t) < g_{n,r}(t)/\sigma_{n,r}^2(t)$, then we have $g_n(t)/\sigma_n^2(t) < 2g_{n,r}(t)/\sigma_{n,r}^2(t) - g_n(t)/\sigma_n^2(t)$. Then, the transmit rate which mobile device n offloads the computation tasks through relay node r is given by

$$R_{n,r}(t) = \frac{BD_{n\to r}^{trust}(t)}{2} \log_2(1+\frac{g_{r,n}(t)g_n(t)P_{tot,n}(t)/(\sigma_{r,n}^2(t)\sigma_n^2(t))}{g_{n,r}(t)/\sigma_{n,r}^2(t)+g_{r,n}(t)/\sigma_{r,n}^2(t)-g_n(t)/\sigma_n^2(t)}), \quad (9)$$

where B is bandwidth.

For secure communications, the trust value of relay node should be considered when offloading data through the relay node. Then, the selection of relay node is based on the transmit rate $R_{n,r}(t)$. When $R_{n,r*}(t) > R_{n,r}(t), r \in \mathcal{R}$, mobile device n offloads computation tasks through relay node $r*$. Then the rate of mobile device n is given by

$$r'_n(t) = \max\{R_{n,r}(t),\ \forall r \in \mathcal{R}\}. \quad (10)$$

The delay for mobile device n to offload data is ψ_n. Then the offloaded data size for mobile device n in slot t is given by

$$D_n(t) = b_n(t)\upsilon(t) = r'_n(t)\psi_n, \quad (11)$$

where $b_n(t)$ and $\upsilon(t)$ are the amount of raw data and the communication overhead in computation offloading, respectively. Then, the computation rate of cooperation offloading, denoted by $r_{off}(t)$, is given by

$$r_{n,off}(t) = \frac{r'_n(t)\psi_n}{\triangle t\upsilon(t)}. \quad (12)$$

After decoding the signals from mobile devices and relay nodes, MEC servers can perform the offloaded tasks. The clock speed of CPU consumed by the computing tasks of mobile device n is represented by s_n (in CPU cycles/s), which is a constant. Then the time when the MEC server performs mobile device n is given by

$$\tau_n(t) = \frac{D_n(t)L_n}{s_n} = \frac{r'_n(t)\psi_n L_n}{s_n}. \tag{13}$$

Since the computation results are very small, we ignore the return time of the computing results in this paper. The computation rate of MEC server n is given by $\frac{s_n}{L_n}$. The time that the computation tasks of mobile device n are successfully executed is given by

$$t'_{n,off}(t) = \psi_n + \tau_n(t). \tag{14}$$

Accordingly, the total computation rate and the total time of mobile device n are respectively given by

$$r_n(t) = a_n(t)r_{loc}(t) + (1 - a_n(t))(r_{n,off}(t) + \frac{s_n}{L_n}), \tag{15}$$

$$t_{tot,n}(t) = a_n(t)t_{loc} + (1 - a_n(t))t'_{n,off}(t), \tag{16}$$

where $a_n(t) \in \{0, 1\}$ is the offloading decision of mobile device n. When $a_n(t) = 1$, the computation tasks of mobile device n are executed locally. Otherwise, the computation tasks are offloaded to the MEC server.

C. Blockchain System

Any node in the blockchain can collect the transactions from the MEC system. To improve the system performance, some blockchain nodes with a high number of votes are selected as consensus nodes to participate in generating blocks and verifying blocks. The number of votes for a consensus node candidate is determined by the number of stakes it holds, its available resources and its trust value. We assume that the stake and available computing resource of blockchain node n in slot t are represented by $\Phi_n(t)$ and $T_n(t)$, respectively. The available computing resource of the node is the remaining resource after processing the offloaded tasks. Denote the sets of the stake and available computing resource of nodes by $\mathbf{\Phi}_s(t) = \{\Phi_1(t), \Phi_2(t), ..., \Phi_n(t)\}$ and $\boldsymbol{T}(t) = \{T_1(t), T_2(t), ..., T_n(t)\}$, respectively. We assume that the MEC server has a first in first out (FIFO) data buffer to store the arrived but not yet executed offloaded tasks. Hence, the dynamics of the processing queue at the beginning of the $t+1$ time slot can be given by as follows.

$$F_n(t+1) = \max\{F_n(t) - s_n + \rho_n r_n(t), 0\}, \tag{17}$$

where ρ_n is the processing density (in CPU cycles/bit). Then, the computing resource available to the blockchain system by MEC server n in the slot t is given by

$$T_n(t) = \max\{F' - F_n(t), T_{min}\}, \tag{18}$$

where F' and T_{min} are the total computing capacity of MEC servers and the minimum computing capacity required by the blockchain system, respectively. Let D_n^{trust} denote the trust value of blockchain nodes. In the paper, we assume that these K block producers, in turn, generate blocks [15]. Let $S_b(t)$ and $T_b(t)$ denote the block size (in *MB*) and block interval (in *seconds*) in slot t, respectively.

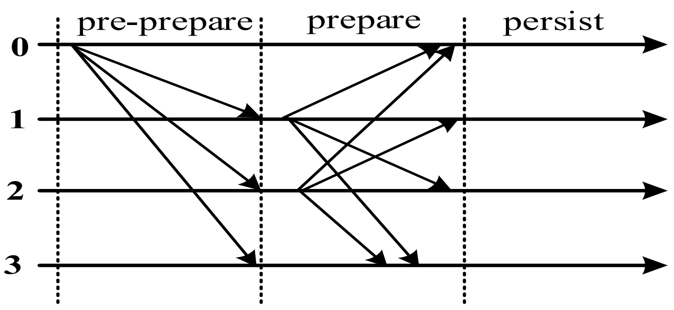


Fig. 2: The process of the consensus algorithm.

After generating a block, the block needed to be verified. In the consensus process, we utilize the delegated Byzantine fault tolerance (dBFT) consensus mechanism [34]. When there are K consensus nodes in the consensus process, we assume that $K \geq 3f + 1$, where f is the maximum number of fault-tolerant nodes. In the consensus process, the leader of the node is called the speaker, and the others are called members. The speaker is responsible for broadcasting new block proposals to other nodes. The members are responsible for voting on the new block proposal. When the number of votes is not less than $K - f$, the proposal is passed. The speaker p of the consensus process is determined by

$$p = (h - v) \bmod N, \tag{19}$$

where h is the block height of the current consensus, and v is the view number. Then, the process of the consensus algorithm is shown in Fig. 2.

The algorithm can be divided into three phases: *pre-prepare*, *prepare*, and *persist*. During the pre-prepare phase, the speaker for this round is responsible for broadcasting a message to other members. Meanwhile, the speaker launches a proposal. In the prepare phase, the members broadcast the message and vote. When a consensus node receives no less than $K - f$ signatures of the block, it enters the third phase, and a block is successfully generated in the phase. Meanwhile, the block is broadcasted the whole blockchain system, and then enter the next round of the consensus process.

Let $T_c(t)$ denote the time cost in the consensus process. For simplicity, the consensus process is divided into two parts, i.e., message propagating and message verification, including signatures verification, message authentication codes (MAC) generation, and MAC verification [35]. Then, the latency of the consensus process in slot t is give by

$$T_c(t) = T_p(t) + T_v(t), \tag{20}$$

where $T_p(t)$ and $T_v(t)$ are the time of message propagation and the validation time in slot t, respectively.

Similar to [15], we utilize latency time to finality (LTF) as the latency of the blockchain system. The LTF is given by

$$T_{total}(t) = T_c(t) + T_b(t). \tag{21}$$

Then, the transaction throughput [15] can be expressed as

$$\Psi(t) = \frac{\lfloor S_b(t)/\chi \rfloor}{T_b(t)}, \tag{22}$$

where χ is the average size of transactions.

IV. Trust Calculation in Blockchain-enabled MEC Systems

For secure communications, only relay nodes with high trust values should be chosen to relay the offloaded data to the MEC server. If computing tasks are relayed by a relay node with low trust value, the relay node may take malicious actions, e.g., dropping relaying data packets. Therefore, each mobile device should interact with relay nodes with high trust value to avoid potential security threats. Similarly, malicious block producers may generate a fake block. Therefore, the consensus nodes selected should have a higher trust value. To compute the trust values of nodes (relay nodes and consensus nodes), we jointly utilize two common ways to evaluate, i.e., direct trust and indirect trust (recommendation) [36]. Direct trust values of nodes are calculated based on subjective logic, while indirect trust values are computed based on the third party's recommendations. In this work, we evaluate the trust value of a node by a real number ranging from 0 to 1. Like most literature, such as [14], [37], the trust threshold is set 0.5. In other words, the node is trustworthy when its trust value is higher than 0.5; otherwise, it is not trustworthy. Next, we first calculate the trust value of relay nodes.

A. Calculation of Direct Trust

Similar to [36], we utilize node honesty and node capacity to calculate direct trust. Since mobile communication channels between mobile devices and relay nodes are unstable and noisy, the communication behaviors in computation offloading involves considerable uncertainty. We tackle the uncertainty by using a Subjective Logic framework [38]. The trust value of mobile device n to relay node r in the Subjective Logic framework can be described as a triplet $\omega_{n\to r} = \{b_{n\to r}, d_{n\to r}, \nu_{n\to r}\}$, where $b_{n\to r}$, $d_{n\to r}$, and $\nu_{n\to r}$ represent belief, disbelief and uncertainty, respectively. Peculiarly, the relationships among them are determined by

$$\begin{aligned} &b_{n\to r}, d_{n\to r}, \nu_{n\to r} \in [0,1], \\ &b_{n\to r} + d_{n\to r} + \nu_{n\to r} = 1. \end{aligned} \tag{23}$$

Based on the trust model of [39], the node honesty (NH) can be given by

$$NH_{n\to r} = b_{n\to r} + \xi \nu_{n\to r}, \tag{24}$$

where $0 \leq \xi \leq 1$ is a constant indicating the degree of influence of trust uncertainty [40] and

$$\begin{aligned} b_{n\to r} &= (1-\nu_{n\to r})\frac{\alpha_{n\to r}}{\alpha_{n\to r}+\beta_{n\to r}}, \\ d_{n\to r} &= (1-\nu_{n\to r})\frac{\beta_{n\to r}}{\alpha_{n\to r}+\beta_{n\to r}}, \\ \nu_{n\to r} &= 1 - s_{n\to r}, \end{aligned} \tag{25}$$

where $\alpha_{n\to r}$ and $\beta_{n\to r}$ are the number of successful and unsuccessful communication, respectively. $s_{n\to r}$ represents the quality of communication link, which refers to the packet success probability. The packet loss is not only caused by mobile communication channels, but also induced by malicious nodes [36]. Therefore, the value of $\alpha_{n\to r}$ and $\beta_{n\to r}$ can be respectively recast as

$$\alpha_{n\to r}^{new} = \alpha_{n\to r} + P_{n\to r}^{plr} \times (\alpha_{n\to r} + \beta_{n\to r}), \tag{26}$$

$$\beta_{n\to r}^{new} = \beta_{n\to r} - P_{n\to r}^{plr} \times (\alpha_{n\to r} + \beta_{n\to r}), \tag{27}$$

where $P_{n\to r}^{plr}$ is the packet loss rate. Similar to [36], the packet loss rate is estimated by the following equation.

$$P_{n\to r}^{plr} = 1 - \frac{\sum_b^c \omega(b) \times \omega(b)}{\sum_b^c \omega(b)}, \tag{28}$$

where $\omega(b)$ is the weight value of a historical link state and let *link* $= (\omega(1), \omega(2), ..., \omega(b))$ be a historical link state record. The wight value is given by $\omega(b) = \frac{2b}{c(c+1)}$, where b and c are the serial number of $\omega(b)$ in *link* and the number of the state record, respectively.

On the other hand, we assume that all relay nodes have the same initial energy consumption rate and energy level. When malicious nodes launch malicious attacks, they can always consume anomalous energy. Therefore, we utilize energy as a quality of service (QoS) trust metric to measure whether a relay node is malicious or not. Let $P_{n\to r}^{pen}$ be the energy consumption rate, which is achieved by using the Ray Projection method [41] ($P_{n\to r}^{pen} \in [0,1]$). Then the node competence (NC) is given by

$$NC_{n\to r} = \begin{cases} 1 - P_{n\to r}^{pen}, & \text{if } E_{n\to r}^{res} \geq \theta, \\ 0, & \text{otherwise,} \end{cases} \tag{29}$$

where $E_{n\to r}^{res}$ and θ are the residual energy of one relay node and the energy threshold, respectively.

As mentioned above, the node trust relies on the node honesty and node competence. Then, the direct trust of a relay node is defined as

$$D_{n\to r}^{direct} = \begin{cases} 0.5 + (NH_{n\to r} - 0.5) \times NC_{n\to m}, & \text{if } NH_{n\to r} \geq 0.5, \\ NH_{n\to r} \times NC_{n\to r}, & \text{otherwise.} \end{cases} \tag{30}$$

B. Calculation of Recommendation

For calculation of trust value, we also consider the recommendation from the third party, i.e., blockchain systems. We assume that some relay nodes are willing to contribute their resources to help mobile devices offload computing tasks. These relay nodes are called candidates. When a mobile device needs to offload tasks via relay model, the candidates around it send a request to the blockchain system and recommend themselves to assist it in completing the tasks offloading. Upon receiving the request, the blockchain system will select a suitable candidate based on the recommended value of each candidate stored in the system. We assume that the blockchain system periodically updates and stores the candidate's recommended value. However, not every updated recommendation is reliable. If only a single updated recommended value of a candidate is considered, it is likely that an unreliable candidate's recommendation is adopted, resulting in an unreliable trust evaluation. Therefore, we need to detect whether the recommendation is reliable before calculating the trust value.

For this purpose, we present a simple way to detect the trust value by defining the recommended reliability $R_{n\to r}^{rel}$. To begin with, we compute the average value of all updated recommendations for candidate r, denoted by R_r^{ave}. Then, we obtain the difference between the recommendation value and the average value. The greater the difference, the lower the reliability of the recommendation. Therefore, the recommended reliability $R_{n\to r}^{rel}$ is given by

$$R_{n\to r}^{rel} = 1- \mid R_{n\to r}^{rec,i} - R_r^{ave} \mid, \tag{31}$$

where $R_{n\to r}^{rec,i}$ represents the recommended value for the ith update in the blockchain system.

If the recommended reliability of a recommender is less than 0.5, even if it has a high recommended value, the recommended value cannot be used to compute the recommended trust. Therefore, we obtain the recommended trust based on the recommended reliability and the recommended value as follows.

$$R_{n\to r}^{recom} = \frac{\sum_{i=1}^{I} R_{n\to r}^{rel} \times R_{n\to r}^{rec,i}}{I}, \tag{32}$$

where I is the number of updates. Therefore, the relay node trust is given by

$$D_{n\to r}^{trust} = \begin{cases} D_{n\to r}^{direct}, & \text{if } \alpha_{n\to r}^{new} \geq Th_{num}, \\ \omega_{direct} D_{n\to r}^{direct} + \omega_{recom} R_{n\to r}^{recom}, & \text{otherwise,} \end{cases} \tag{33}$$

where ω_{direct}, ω_{recom}, and Th_{num} are the weight values of the direct value and the recommendation, and the number of interaction between recommenders and the blockchain system, respectively. $\omega_{direct} \in [0, 1]$, $\omega_{recom} \in [0, 1]$, and $\omega_{direct} + \omega_{recom} = 1$. Similarly, the trust value of the nodes in the blockchain system is evaluated using the same method.

V. Problem Formulation

It is well known that wireless channels have the Markovian property [42], [43]. Therefore, the blockchain-enabled MEC system is formulated as a discrete MDP to maximize the system reward. Since it is impossible to predict the state transition probability and reward in advanced in mobile environment, we propose a model-free approach based on deep reinforcement learning to solve the above the MDP problem. The MDP is defined by a tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, r >$, where $\mathcal{S}$ is the state set of the system, $\mathcal{A}$ is the action set of the system, $\mathcal{P}$ is the state transition probabilities, and r is the reward function.

A. State Space and State Transition Probability

We define the state space at the current decision epoch t ($t = 1, 2, ...$) as a union of the wireless channels conditions $\boldsymbol{G}(t) = (g_n(t), g_{n,r}(t), g_{r,n}(t))$, the available computing resource of the MEC server $\boldsymbol{T}(t) = \{T_1(t), T_2(t), ..., T_n(t)\}$, the number of the stakes $\boldsymbol{\Phi}_s(t) = \{\Phi_1(t), \Phi_2(t), ..., \Phi_n(t)\}$, and the trust value of relay nodes and blockchain nodes $\boldsymbol{D}^{trust}(t) = \{D_{n\to r}^{trust}(t), D_n^{trust}(t)\}$, which is denoted as

$$\mathcal{S}(t) \triangleq \left\{\boldsymbol{G}(t), \boldsymbol{T}(t), \boldsymbol{\Phi}_s(t), \boldsymbol{D}^{trust}(t)\right\}. \tag{34}$$

Since the state space is continuous, the probability of being in a particular state is zero. The probability that the process will leave the state $s(t)$ to transition to the next state $s(t+1)$ after taking an action $a(t) \in \mathcal{A}$ can be expressed as

$$Pr(s(t+1) \mid s(t), a(t)) = \int_{\mathcal{S}^{t+1}} f(s(t), a(t), s')ds', \tag{35}$$

where f is the state transition probability density function.

B. Action Space

The action space includes offloading decision $\boldsymbol{a}(t)$, power allocation $\boldsymbol{P}(t)$, block size $S_b(t)$, and block interval $T_b(t)$. We utilize $\mathcal{A}(t) = [\boldsymbol{a}(t), \boldsymbol{P}(t), a_{S_k}(t), a_{T_k}(t)]$ to define the action set.

Offloading Decision: The offloading decision is denoted by

$$\boldsymbol{a}(t) \triangleq [a_1(t), a_2(t), ..., a_N(t)]. \tag{36}$$

Power Allocation Decision: The power allocation decision will be obtained based on achieving a maximum reward. We denote the power allocation decision by $\boldsymbol{P}(t)$, as shown below.

$$\boldsymbol{P}(t) \triangleq [P_{total,1}(t), P_{total,2}(t), ..., P_{total,N}(t)]. \tag{37}$$

Block Size and Block Interval: The delegators are elected by voting based on the number of stakes held by the blockchain nodes, the trust value of blockchain nodes, and available computing resource. After determining the delegators, they take turns to produce blocks. By using the limits fractional method, the block size and block interval decisions are respectively given by

$$a_{S_k}(t) \in [0.2, \dot{S}_b], \tag{38}$$

$$a_{T_k}(t) \in [0.1, \dot{T}_b], \tag{39}$$

where $\dot{T}_b$ are the block size limit and the maximum block interval, respectively.

C. Reward Function

In this paper, we formulate an optimization problem to maximize the weighted sum of the computation rate of the MEC system and the transaction throughput of the blockchain system, which jointly optimizes offloading decision, power allocation, block size, and block interval. Then, the joint optimization problem is formulated as

$$\begin{aligned} \max_{\mathcal{A}^t} \mathbb{E} & \left[\sum_{t=0}^{T-1} \omega_1 \omega_2 \sum_{n=1}^{N} r_n(t) + (1-\omega_1)\Psi(t)\right] \\ \text{s.t.} \quad & (\text{C1}) : t_{tot,n}(t) \leq \varepsilon, \\ & (\text{C2}) : T_{total}(t) \leq \omega \times T_b(t), \\ & (\text{C3}) : 0 \leq P_{tot,n}(t) \leq P_T, \\ & (\text{C4}) : a_n(t) \in \{0, 1\}, \end{aligned} \tag{40}$$

where $\omega > 1$, and $\omega_1 (0 < \omega_1 < 1)$ is a weight factor to combine the objective function to a single one, and ω_2 is a mapping factor that ensures that the objective function is at the same level. $\varepsilon (\varepsilon \leq \Delta t)$ is the maximum tolerable average delay in offloading tasks. P_T is the sum of the power available for all mobile devices and relay nodes in the network. Then, we define the reward function as

$$r_t = \begin{cases} O(t), & \text{if } C1 - C4 \text{ are satisfied,} \\ 0, & \text{otherwise,} \end{cases} \tag{41}$$

where $O(t) = \omega_1 \omega_2 \sum_{n=1}^{N} r_n(t) + (1 - \omega_1)\Psi(t)$.

VI. Offloading Decision and Resource Allocation in the A3C Framework

Compared with other DRL algorithms, such as actor-critic learning (AC), advantage actor-critic learning (A2C), and policy-based learning, the A3C is a faster, simpler, and more robust parallel reinforcement learning algorithm proposed by Google DeepMind in 2016 [44]. It can reliably train deep neural network policies. Different from the underlying reinforcement learning algorithms, such as actor-critic that is an on-policy search algorithm, and Q-learning that is an off-policy value-based search algorithm, A3C combines the benefits of the value-based method and the policy-based method [44]. More importantly, it could work in discrete as well as continuous action spaces. A3C utilizes asynchronous actor-learners, i.e., employing multiple CPU threads on a single machine, to learn more efficiently. Multiple actor-learners running in parallel can interact with their environment and obtain different exploration policies. Moreover, the exploration policy of each actor-learner is independent of those of the others. Hence, the overall exploration policy available for training becomes more diverse.

In an A3C algorithm, we need to maintain a policy $\pi(a_t|s_t;\theta)$ (a set of action probability outputs) with the parameter θ and an estimate of the value function $V(s_t;\theta_v)$ (how good a certain state is to be) with the parameter θ_v. Compared with traditional policy gradient methods, A3C is more intelligent because the agent utilizes the estimated value function (the critic) to update the policy (the actor). The policy and the value function are updated in the terminal state or after maximum step t_{max} actions. In the policy-based methods, the rule is updated by using the discounted returns, which is given by

$$R_t(\theta_v) = \gamma(r) = \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V(s_{t+k};\theta_v), \tag{42}$$

where k is vary from state to state and is upper-bounded by t_{max}, r_{t+i} is immediate reward, and $\gamma \in (0, 1]$ is the discount factor.

However, the estimate can cause the variance. To reduce the variance of the estimate, the advantage estimates is adopted, which is given by

$$A(a_t, s_t) = Q(a_t, s_t) - V(s_t). \tag{43}$$

Since the $Q(a_t, s_t)$ value cannot be determined in A3C, the discounted returns is used as the estimate of $Q(a_t, s_t)$ to generate an estimate of the advantage. Then, the advantage function is given by

$$\begin{aligned} A(a_t, s_t; \theta, \theta_v) &= R_t(\theta_v) - V(s_t;\theta_v) \\ &= \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V(s_{t+k};\theta_v) - V(s_t;\theta_v). \end{aligned} \tag{44}$$

The policy $\pi(a_t|s_t;\theta)$ and the value function $V(s_t;\theta_v)$ are approximated by using a single convolutional neural network. Especially, the policy function is output by a softmax layer, and the estimate of the value function is output by a linear layer. In A3C, all network weights are stored in a central parameter server [45]. In the beginning, each actor-learner sets its network parameters to those of the server. Then, multiple actor-learners learn concurrently and optimize the convolutional neural network through asynchronous gradient descent. After computing the gradient, the actor-learners send the updates to the server. Then, the server propagates new weights to the actor-learners to ensure they share a common policy.

Two loss functions are associated with the two convolutional neural network outputs. For policy loss function, we have

$$f_\pi(\theta) = \log \pi(a_t \mid s_t;\theta)(R_t - V(s_t;\theta_v)) + \beta H(\pi(s_t;\theta)), \tag{45}$$

where $H(\pi(s_t;\theta))$ is the entropy. β is a hyperparameter that controls the strength of the entropy regularization term.

Differentiating the policy loss function in (45) with respect to the parameter θ, we have

$$\begin{aligned} \nabla_\theta f_\pi(\theta) = \nabla_\theta \log \pi(a_t \mid s_t;\theta)(R_t - V(s_t;\theta_v)) \\ + \beta \nabla_\theta H(\pi(s_t;\theta)). \end{aligned} \tag{46}$$

The loss function for estimated value function is given by

$$f_v(\theta_v) = (R_t - V(s_t;\theta_v))^2. \tag{47}$$

Similarly, differentiating the value loss function in (47) with respect to θ_v yields

$$\nabla_{\theta_v} f_v(\theta_v) = 2(R_t - V(s_t;\theta_v))\nabla_{\theta_v} V(s_t;\theta_v). \tag{48}$$

The loss function can be minimized by adopting RMSProp algorithm that has been widely used in the deep learning algorithms. Then, the estimate of the gradient under RMSProp is given by

$$g = \alpha g + (1 - \alpha)\Delta\theta^2, \tag{49}$$

where α is the momentum, and $\Delta\theta$ is the accumulated gradients of the loss function.

Then, the RMSProp algorithm can be updated according to the following estimated gradient downhill.

$$\theta \leftarrow \theta - \eta \frac{\triangle\theta}{\sqrt{g + \epsilon}}, \tag{50}$$

where η is the learning rate, and ϵ is small positive number.

Based on (46) and (48), the detail of the A3C algorithm used in our proposed approach is shown in Algorithm 1.

Algorithm 1 A3C-Based Computation Offloading and Resource Allocation Algorithm

```
Initialization:
  • Assume that $\theta$ and $\theta_v$ are parameters the actor network
    and critic network in global network.
  • Assume that $\theta'$ and $\theta'_v$ are parameters the actor network
    and critic network in local network.
  • Set global counter $T = 0$ and local step counter $t = 1$.
  • Set $T_{max}$, $t_g$, $\gamma$, learning rate $\eta$, $\epsilon$, and $t_{max}$.
  • Set the number of agents $W$.
Iteration:
 1: while $T < T_{max}$ do
 2:   for $w = 1$ to $W$ do
 3:     Reset global gradient $d\theta = 0$ and $d\theta_v = 0$.
 4:     Synchronize local parameters $\theta' = \theta$ and $\theta'_v = \theta_v$.
 5:     Set $t_0 = t$ and obtain system state $\mathcal{S}(t)$.
 6:     repeat
 7:       Obtain action $\mathcal{A}(t)$ according to policy
          $\pi(\mathcal{A}(t)|\mathcal{S}(t);\theta')$.
 8:       Execute action $\mathcal{A}(t)$, observe reward $\mathcal{R}(t)$, and
          observe next state $\mathcal{S}(t+1)$.
 9:       $t = t + 1$.
10:     until $t - t_0 == t_{max}$
11:     if $t\%t_g == 0$ then
12:       $R = V(\mathcal{S}(t);\theta'_v)$.
13:     end if
14:     for $i = t-1$ to $t_0$ do
15:       $R = \mathcal{R}(t) + \gamma R$.
16:       Compute policy gradient $\nabla_{\theta'} f_\pi(\theta')$ according to
          (46).
17:       Compute accumulate gradient $d\theta = d\theta +$
          $\nabla_{\theta'} f_\pi(\theta')$.
18:       Compute value gradient $\nabla_{\theta'_v} f_v(\theta'_v)$ according to
          (48).
19:       Compute accumulate gradient $d\theta_v = d\theta_v +$
          $\nabla_{\theta'_v} f_\pi(\theta'_v)$.
20:     end for
21:     Asynchronous update weight parameter $\theta$ and $\theta_v$
        according to (50).
22:   end for
23: end while
```

VII. Simulation Results and Analysis

In this section, we evaluate the performance of the proposed algorithm under different parameter settings. Simulation is performed using Tensorflow [46] on a Python-based simulator. To verify the performance of the proposed algorithm, we consider the following schemes: 1) *Proposed scheme without local computing (Only-offloading)*: the computation tasks are only offloaded to MEC servers. 2) *Proposed scheme with fixed block size (FBS)*: the size of the blocks generated by the block producers is the same. 3) *Proposed scheme with fixed block interval (FBT)*: the frequency of generating blocks is the same.

TABLE II: Summary of the Simulation Parameters

Parameters	Definition	Values
B	Bandwidth	180 KHz [50]
P_T	Maximum power available	1 W [51]
φ_n	Transmit time	0.4 s [47]
N_0	Noise power density	-174 dBm/Hz [3]
χ	Average transaction size	200 KB
$\bar{S}_b$	Block size	8 MB [15]
ε	Tolerable maximum delay	1 s [5]
L_n	Processing density	737.5 cycle/bit [5]
F	Computation capability	2.5 GHz [5]
ϖ_2,ϖ_1	The weighted values	0.2, 0.0001
η_a,η_c	Learning rate	10^{-3},10^{-2}
ξ	Shadowing standard deviation	10 dB [52]

A. Simulation Parameters

We consider a network that consists of an MEC system and a blockchain system, which comprises 30 mobile devices scattering over a 2×2 km^2 area [47]. The number of relay nodes within the coverage of each BS is 5. The CPU-cycle frequency of mobile devices and MEC servers is 1 GHz [5] and 2.4 GHz [48], respectively. Other simulation parameters are summarized in Table II, where the path loss models and the shadowing fading are standard settings provided by 3GPP [49]. In our simulations, we use a computer, which has 6 CPU cores. The CPU is Intel Core i5-8400 with 32G memory. The software environment we used is Tensorflow 1.10.0 with Python 3.6 on Ubuntu 18.04.2 LTS. For the blockchain system, we used virtualization for distributed ledger technology (vDLT) we developed, which is a service-oriented blockchain system with virtualization and decoupled management/control and execution. Different block sizes can be dynamically set in vDLT by chaning the parameters in vDLT. For more details, please go to http://vdlt.io/approach.html.

By using Tensorflow's built-in module TensorBoard, we show the visualization of our A3C architecture, as shown in Fig. 3. In Fig. 3, the architecture of the proposed algorithm consists of a global network and six worker agents. We can observe that the proposed algorithm starts with constructing the global network. Then, the parameters in the global network are propagated synchronously to each worker agent. In Fig. 4, we show the internal structure of one of the worker agents. Every worker agent has its own network and environment. By interacting with their own environment, worker agents update the global network parameters.

B. Performance of the Proposed Algorithm

We first show the convergence of the proposed algorithm under different actor's learning rate η_a, which the critic's learning rate is set to a fixed value $\eta_c = 10^{-1}$, as shown in Fig. 5. As can be seen from the figure, when the actor's learning rate is large, the proposed algorithm has a fast convergence rate (i.e., $\eta_a = 0.0001$). Similarly, Fig. 6 shows the convergence of the proposed algorithm under different critic's learning rate η_c, which we fix the actor's learning rate $\eta_a = 10^{-4}$. From

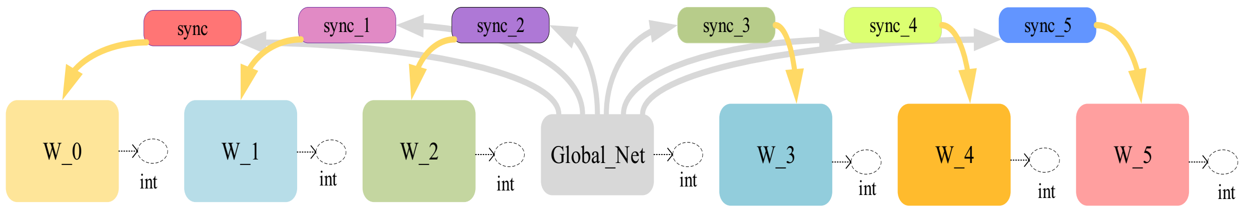


Fig. 3: Visualization of the proposed deep reinforcement learning algorithm using TensorBoard.

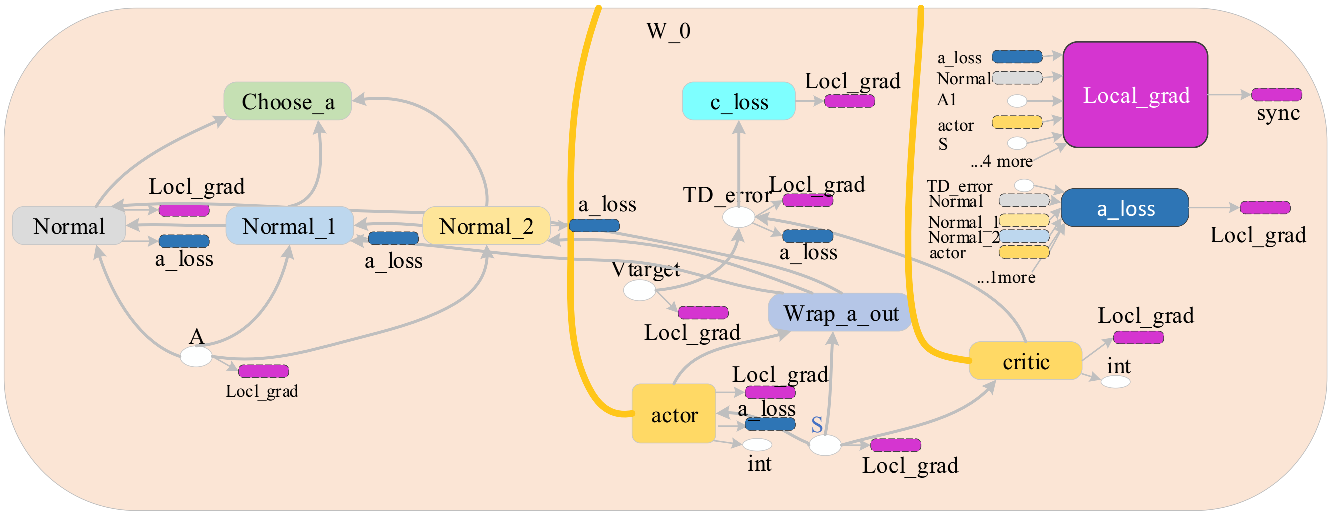
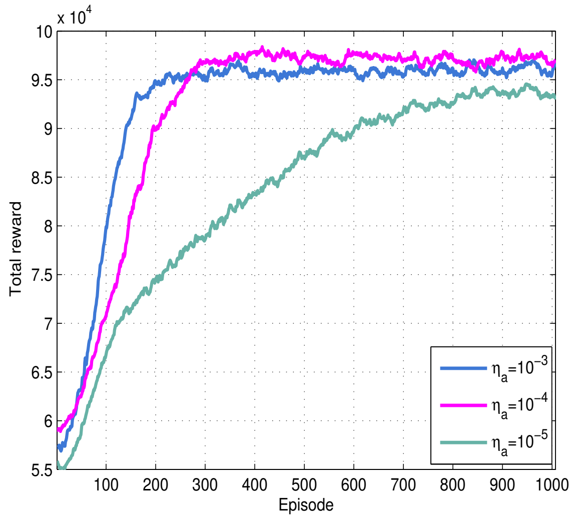


Fig. 4: Visualization of interaction in the worker agent.



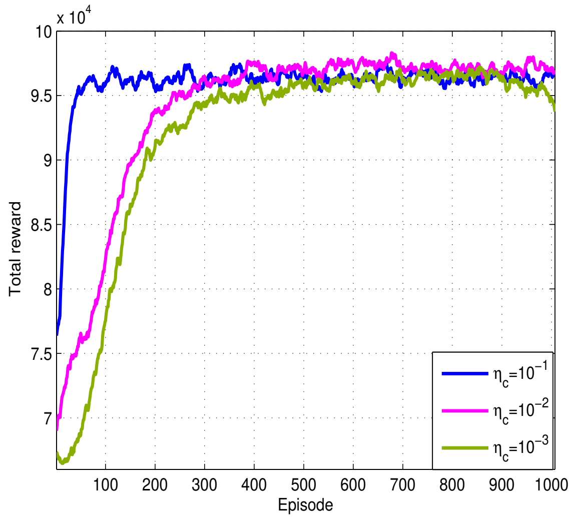


Fig. 5: The total reward with different learning of the actor network. Fig. 6: The total reward with different learning of the critic network.

the figure, we can observe that when the critic's learning rate $\eta_c = 0.1$, the proposed algorithm first converges, followed by $\eta_c = 0.01$ and $\eta_c = 0.001$.

In Fig. 7, we illustrate how the sum of the power available P_T affects the average reward. We can observe that the average reward increases when P_T increases. However, the proposed algorithm performs better than other schemes. From the figure, the average reward of all schemes grows slowly when the value of the sum of power available, P_T, is greater than 0.7. The reason is that as the transmit power increases, although the transmit rate increases, the overhead of communication increases, such as energy consumption, which can affect the computation rate.

Fig. 8 and Fig. 9 show the impact of the CPU-cycle frequency of the MEC servers s_n on average computation rate and average transaction throughput, respectively. From the Fig. 8, we can observe that the average computation rate of all schemes increases slowly with the increase in the CPU-cycle

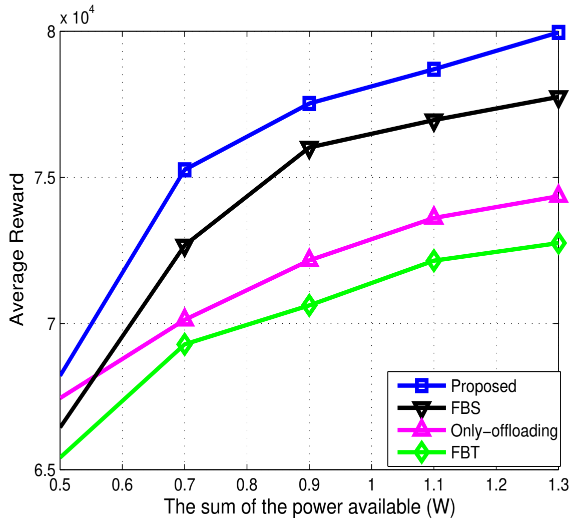


Fig. 7: Average reward versus the total power available P_T.

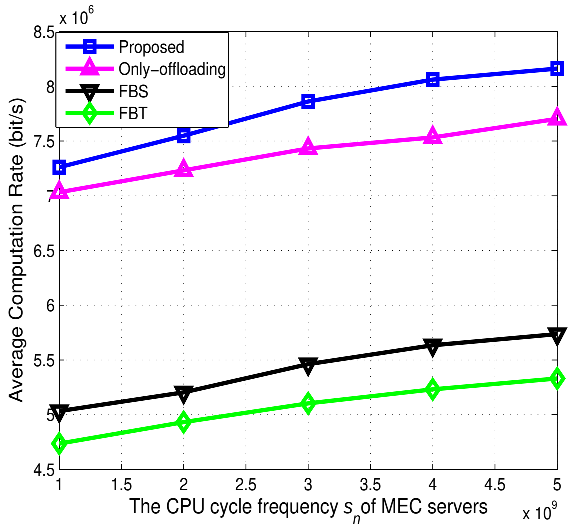


Fig. 8: Average computation rate versus the CPU-cycle frequency s_n.

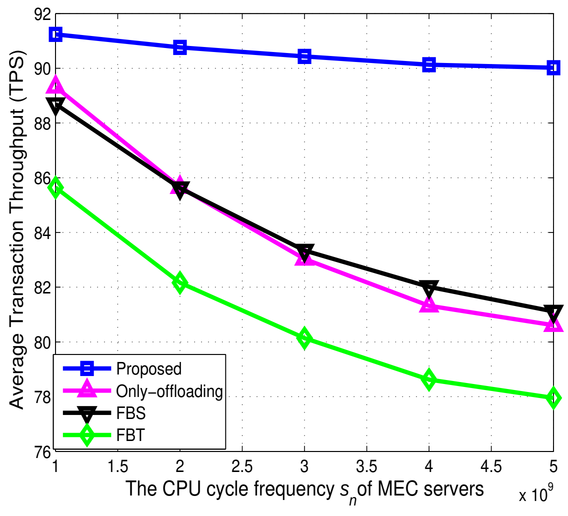


Fig. 9: Average transaction throughput versus the CPU-cycle frequency s_n.

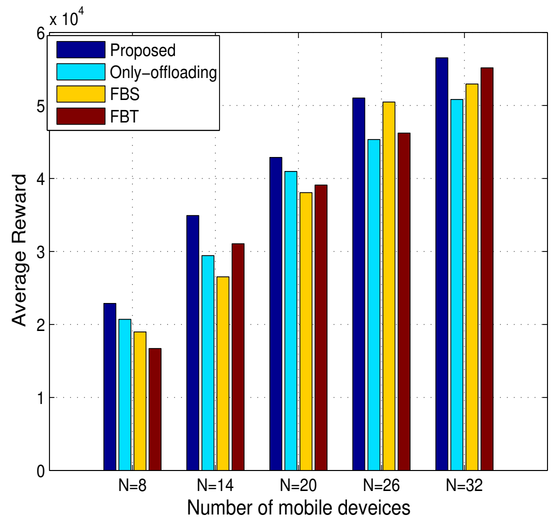


Fig. 10: Average reward versus number of mobile devices N

frequency s_n. However, from Fig. 9, it is observed that the average transaction throughput decreases with the increase in the CPU-cycle frequency s_n for all schemes. That is because the computing resource of MEC servers is limited. When the MEC server consumes more computing resource to perform the offloading tasks, the computing resource available to the blockchain system become less.

Fig. 10 shows the comparison of the average reward versus the number of mobile devices. As can be seen from the figure, with the number of mobile devices increases, the average reward keeps increasing. Due to the joint optimization of offloading decision, the allocation of transmit power, block size, and block interval, the proposed algorithm can always benefit compared with other algorithms that only optimize part of the optimization items.

In Fig. 11, we examine the average reward under different maximum block interval $\dot{T}_b$. It can be observed that the average reward of all the schemes decreases with the increase in the maximum block interval. That is because the transaction throughput decreases with the increase in maximum block interval when other parameters are unchanged. To verify the impact of the average transaction size χ on the average reward, we evaluate the performance obtained by the proposed scheme under different average transaction size, as shown in Fig. 12. From the figure, we can observe that the average reward for all schemes decreases with the increasing average transaction size. The reason is that one block can only contain a small number of transactions with large-size transactions. Furthermore, we can also find that the proposed scheme can obtain highest average reward with the variation of average transaction size, and then follows the FBS, the Only-offloading scheme, and the lowest scheme is the FBT. Similarly, we also evaluate the impact of block size $\dot{S}_b$ on the average reward, as shown in Fig. 13. Observe that, the average reward

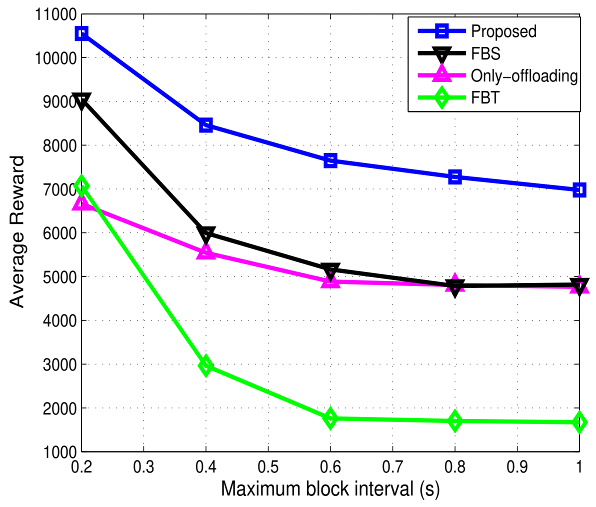


Fig. 11: Average reward versus maximum block interval $\dot{T}_b$.

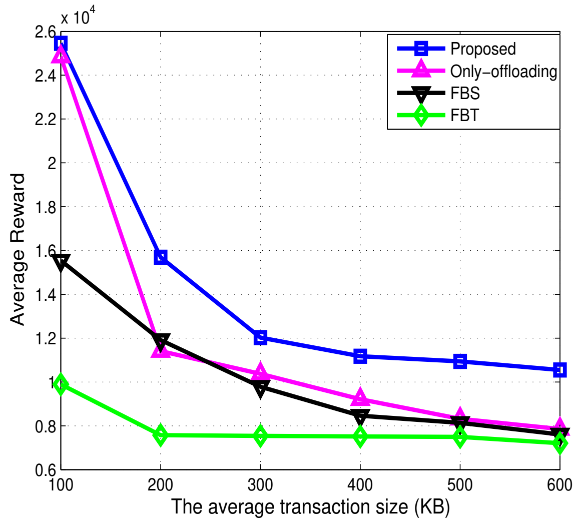


Fig. 12: Average reward versus transaction size χ.

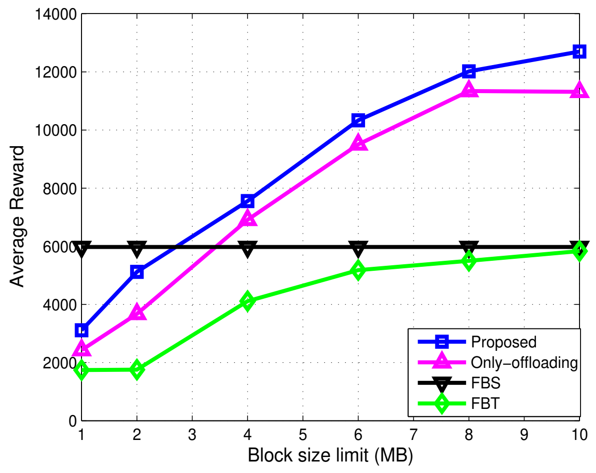


Fig. 13: Average reward versus block size limit $\dot{S}_b$.

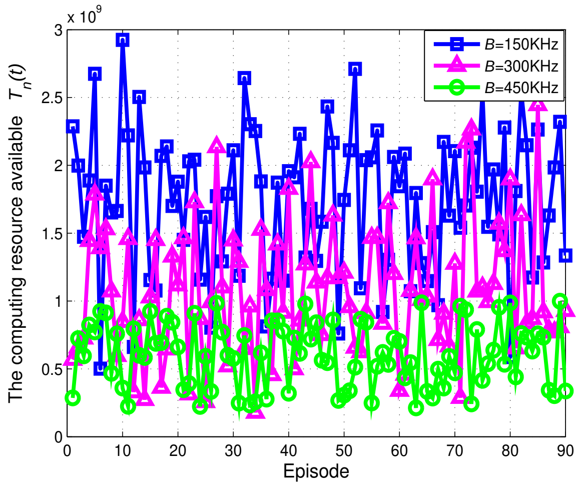


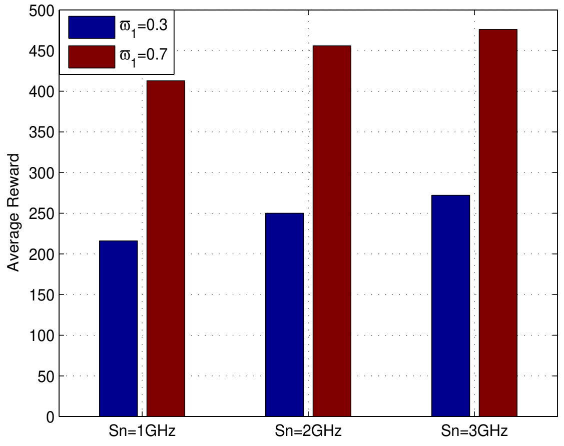
Fig. 14: The computing resource of the randomly chosen blockchain nodes at the randomly selected 90 episode for the proposed algorithm.

slowly increase with the increase in block size except for FBS. That is because the LTF constraint limits the maximum number of transactions in a block. Another observation is that the proposed scheme always performs the best, followed by Only-offloading and FBT. Moreover, we randomly choose a blockchain node in the blockchain system to display its computing resource at randomly selected 90 episodes during $B = 150KHz$, $B = 300KHz$, and $B = 450KHz$ in Fig. 14. From the figure, the queue length of the blockchain nodes at different episodes is finite and because the transmit rate is different. Besides, we can observe that the computing resource available of blockchain node decreases with the increase in bandwidth B. That is because the transmit rate increases with the increase in bandwidth.

In Figs. 15 and 16, we show the impact of the CPU-cycle frequency s_n on the average reward, computation rate of the MEC system, and throughput of the blockchain system, respectively. From Fig. 15, it is in accordance with our intuition that the performance of the reward improves for a given ϖ_1 as CPU-cycle frequency s_n increases. Besides, we can observe that the average reward increase as ϖ_1 increases. That is because the performance of the MEC system is mainly affected by changes in the CPU-cycle frequency, and the performance of the blockchain system is almost constant, as shown in Fig. 16. Then we can achieve the tradeoff between the performance of the MEC system and the performance of the blockchain system based on Fig. 16.

VIII. Conclusions and Future Work

In this paper, we studied a blockchain-enabled MEC system and, considering the trust value of nodes (i.e., relay nodes and consensus nodes), investigated the computation rate of the MEC system and transaction throughput of the blockchain system maximization problem. To satisfy the performance requirements of the system, we jointly optimized cooperative offloading decision, power allocation, block size, and block interval. Due to the dynamic characteristics of the wireless channels and available resources, the formulated optimization



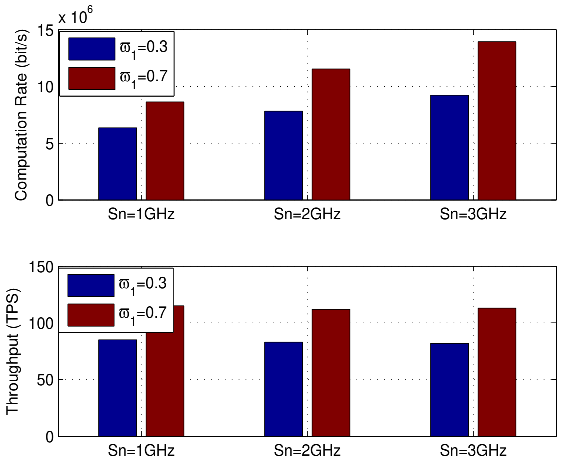


Fig. 15: The impact of different parameter settings on average reward. Fig. 16: The tradeoff under different parameter setting of ϖ_1 and s_n.

problem was modeled as an MDP. An A3C algorithm was developed to cope with the MDP problem, which can stably train neural networks. Simulation results have shown the efficiency of our proposed algorithm, which has fast convergence and better performance than other algorithms under different parameter settings. Meanwhile, we can also observe that the algorithm can achieve the optimal trade-off between the computation rate of the MEC system and transaction throughput in the blockchain system. In future work, we will study interference management in blockchain-enabled MEC systems.

References

[1] M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta, A. Neal *et al.*, "Mobile-edge computing introductory technical white paper," *White paper, mobile-edge computing (MEC) industry initiative*, pp. 1089–7801, 2014.

[2] J. Kwak, Y. Kim, J. Lee, and S. Chong, "Dream: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 12, pp. 2510–2523, Dec. 2015.

[3] J. Du, L. Zhao, X. Chu, F. R. Yu, J. Feng, and C. I, "Enabling low-latency applications in LTE-A based mixed fog/cloud computing systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1757–1771, Feb. 2019.

[4] J. Feng, Q. Pei, F. R. Yu, X. Chu, and B. Shang, "Computation offloading and resource allocation for wireless powered mobile edge computing with latency constraint," *IEEE Wireless Communications Letters*, accepted, 2019.

[5] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 16, no. 9, pp. 5994–6009, Sept. 2017.

[6] 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.

[7] F. Zhou, Y. Wu, R. Q. Hu, and Y. Qian, "Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 1927–1941, Sep. 2018.

[8] Z. Xiong, Y. Zhang, D. Niyato, P. Wang, and Z. Han, "When mobile blockchain meets edge computing," *IEEE Comm. Mag.*, vol. 56, no. 8, pp. 33–39, Aug. 2018.

[9] D. Miller, "Blockchain and the Internet of things in the industrial sector," *IT Professional*, vol. 20, no. 3, pp. 15–18, May 2018.

[10] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Distributed resource allocation in blockchain-based video streaming systems with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 695–708, Jan. 2019.

[11] N. Zhao, H. Wu, and Y. Chen, "Coalition game-based computation resource allocation for wireless blockchain networks," *IEEE Internet of Things J.*, pp. 1–1, 2019.

[12] T. Chen and G. B. Giannakis, "Bandit convex optimization for scalable and dynamic iot management," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 1276–1286, 2018.

[13] M. H. Amini, H. Arasteh, and P. Siano, "Sustainable smart cities through the lens of complex interdependent infrastructures: Panorama and state-of-the-art," in *Sustainable Interdependent Networks II*, 2019, pp. 45–68.

[14] 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.

[15] M. Liu, F. Yu, Y. Teng, V. Leung, and M. Song, "Performance optimization for blockchain-enabled industrial internet of things (IIoT) systems: A deep reinforcement learning approach," *IEEE Trans. Ind. Info.*, pp. 1–1, 2019.

[16] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[17] G. A. Rummery and M. Niranjan, *On-line Q-learning using connectionist systems*. University of Cambridge, Department of Engineering Cambridge, England, 1994, vol. 37.

[18] 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.*, Accepted, 2019.

[19] 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.

[20] X. Cao, F. Wang, J. Xu, R. Zhang, and S. Cui, "Joint computation and communication cooperation for energy-efficient mobile edge computing," *IEEE Internet of Things J.*, vol. 6, no. 3, pp. 4188–4200, Jun. 2019.

[21] S. Guo, J. Liu, Y. Yang, B. Xiao, and Z. Li, "Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing," *IEEE Trans. Mobile Comput.*, vol. 18, no. 2, pp. 319–333, Feb. 2019.

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

[23] J. Kang, R. Yu, X. Huang, M. Wu, S. Maharjan, S. Xie, and Y. Zhang, "Blockchain for secure and efficient data sharing in vehicular edge computing and networks," *IEEE Internet of Things J.*, accepted, 2019.

[24] J. Xu, S. Wang, B. Bhargava, and F. Yang, "A blockchain-enabled trustless crowd-intelligence ecosystem on mobile edge computing," *IEEE Trans. Ind. Infor.*, pp. 1–1, accept, 2019.

[25] 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.*, pp. 1–1, 2019.

[26] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Secure computation offloading in blockchain based iot networks with deep reinforcement learning," *arXiv preprint arXiv:1908.07466*, 2019.

[27] M. S. Alam, J. W. Mark, and X. S. Shen, “Relay selection and resource allocation for multi-user cooperative ofdma networks,” *IEEE Trans. Wireless Commun.*, vol. 12, no. 5, pp. 2193–2205, May 2013.

[28] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, “Mobile-edge computing: Partial computation offloading using dynamic voltage scaling,” *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.

[29] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, “Vehicular edge computing and networking: A survey,” *arXiv preprint arXiv:1908.06849*, 2019.

[30] G. Han, J. Jiang, L. Shu, and M. Guizani, “An attack-resistant trust model based on multidimensional trust metrics in underwater acoustic sensor network,” *IEEE Trans. Mobile Comput.*, vol. 14, no. 12, pp. 2447–2459, Dec. 2015.

[31] Z. Yao, D. Kim, and Y. Doh, “Plus: Parameterized and localized trust management scheme for sensor networks security,” in *2006 IEEE International Conference on Mobile Ad Hoc and Sensor Systems*, 2006, pp. 437–446.

[32] A. P. Miettinen and J. K. Nurminen, “Energy efficiency of mobile clients in cloud computing.” in *Proc. USENIX Conf. Hot Topics Cloud Comput. (HotCloud)*. Boston, MA, USA, Jun. 2012, pp. 1–7.

[33] M. S. Alam, J. W. Mark, and X. S. Shen, “Relay selection and resource allocation for multi-user cooperative ofdma networks,” *IEEE Trans. Wireless Commun.*, vol. 12, no. 5, pp. 2193–2205, May 2013.

[34] “Antshares digital assets for everyone,” [Online]. Available:https://www.antshares.org.

[35] A. Clement, E. L. Wong, L. Alvisi, M. Dahlin, and M. Marchetti, “Making byzantine fault tolerant systems tolerate byzantine faults.” in *NSDI*, vol. 9, 2009, pp. 153–168.

[36] G. Han, J. Jiang, L. Shu, and M. Guizani, “An attack-resistant trust model based on multidimensional trust metrics in underwater acoustic sensor network,” *IEEE Trans. Mobile Comput.*, vol. 14, no. 12, pp. 2447–2459, Dec. 2015.

[37] R. A. Shaikh, H. Jameel, B. J. d’Auriol, H. Lee, S. Lee, and Y.-J. Song, “Group-based trust management scheme for clustered wireless sensor networks,” *IEEE trans. parallel distrib. syst.*, vol. 20, no. 11, pp. 1698–1712, 2009.

[38] A. Josang, “An algebra for assessing trust in certification chains,” in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 1999, pp. 1–10.

[39] Q. Liu, Y. Liao, B. Tang, and L. Yu, “A trust model based on subjective logic for multi-domains in grids,” in *Proc. Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, 2008, pp. 882–886.

[40] N. Oren, T. J. Norman, and A. Preece, “Subjective logic and arguing with evidence,” *Artificial Intelligence*, vol. 171, no. 10-15, pp. 838–854, 2007.

[41] M. Chen, Y. Zhou, and L. Tang, “Ray projection method and its applications based on grey prediction,” *Chin. J. Stat. Decis*, vol. 1, no. 1, pp. 13–20, 2007.

[42] M. Simsek, M. Bennis, and I. Güvenç, “Learning based frequency-and time-domain inter-cell interference coordination in hetnets,” *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4589–4602, 2014.

[43] Y. Wei, F. R. Yu, M. Song, and Z. Han, “User scheduling and resource allocation in hetnets with hybrid energy supply: An actor-critic reinforcement learning approach,” *IEEE Trans. Wireless Commun.*, vol. 17, no. 1, pp. 680–692, 2018.

[44] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, “Asynchronous methods for deep reinforcement learning,” in *International conference on machine learning*, 2016, pp. 1928–1937.

[45] M. Babaeizadeh, I. Frosio, S. Tyree, J. Clemons, and J. Kautz, “Reinforcement learning through asynchronous advantage actor-critic on a gpu,” *arXiv preprint arXiv:1611.06256*, 2016.

[46] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, “Tensorflow: Large-scale machine learning on heterogeneous distributed systems,” *arXiv preprint arXiv:1603.04467*, 2016.

[47] C. You, K. Huang, H. Chae, and B. Kim, “Energy-efficient resource allocation for mobile-edge computation offloading,” *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, March 2017.

[48] Y. Kim, H. Lee, and S. Chong, “Mobile computation offloading for application throughput fairness and energy efficiency,” *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 3–19, Jan 2019.

[49] “3GPP TR 36.839 v11.0.0,” *3GPP, Tech. Rep.*, Sept. 2012.

[50] “Evolved universal terrestrial radio access (E-UTRA): Physical channels and modulation,” *3GPP TS 36.211 V8.6.0 Std.*, Mar., 2009.

[51] P. Guo, W. Hou, L. Guo, W. Sun, C. Liu, H. Bao, L. H. K. Duong, and W. Liu, “Fault-tolerant routing mechanism in 3d optical network-on-chip based on node reuse,” *IEEE Transactions on Parallel and Distributed Systems*, Accepted, 2019.

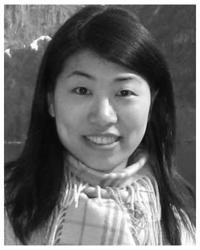
[52] D. Zhai, R. Zhang, Y. Wang, H. Sun, L. Cai, and Z. Ding, “Joint user pairing, mode selection, and power control for d2d-capable cellular networks enhanced by nonorthogonal multiple access,” *IEEE Internet of Things J.*, vol. 6, no. 5, pp. 8919–8932, Oct. 2019.

Jie Feng is currently pursuing the Ph.D. degree in Communication and Information System at Xidian University, Xian, China. She is also with Carleton University as Visiting Ph.D Student since January 2019. Her current research interests include mobile edge computing, Blockchain, deep reinforcement learning, Device to Device communication, resource allocation and convex optimization and stochastic network optimization.

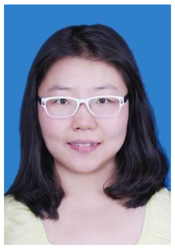
F. Richard Yu (S00-M04-SM08-F18) received the PhD degree in electrical engineering from the University of British Columbia (UBC) in 2003. From 2002 to 2006, he was with Ericsson (in Lund, Sweden) and a start-up in California, USA. He joined Carleton University in 2007, where he is currently a Professor. He received the IEEE Outstanding Service Award in 2016, IEEE Outstanding Leadership Award in 2013, Carleton Research Achievement Award in 2012, the Ontario Early Researcher Award (formerly Premiers Research Excellence Award) in 2011, the Excellent Contribution Award at IEEE/IFIP TrustCom 2010, the Leadership Opportunity Fund Award from Canada Foundation of Innovation in 2009 and the Best Paper Awards at IEEE ICNC 2018, VTC 2017 Spring, ICC 2014, Globecom 2012, IEEE/IFIP TrustCom 2009 and Intl Conference on Networking 2005. His research interests include wireless cyber-physical systems, connected/autonomous vehicles, security, distributed ledger technology, and deep learning.

He serves on the editorial boards of several journals, including Co-Editor-in-Chief for Ad Hoc & Sensor Wireless Networks, Lead Series Editor for IEEE Transactions on Vehicular Technology, IEEE Transactions on Green Communications and Networking, and IEEE Communications Surveys & Tutorials. He has served as the Technical Program Committee (TPC) Co-Chair of numerous conferences. Dr. Yu is a registered Professional Engineer in the province of Ontario, Canada, a Fellow of the Institution of Engineering and Technology (IET), and a Fellow of the IEEE. He is a Distinguished Lecturer, the Vice President (Membership), and an elected member of the Board of Governors (BoG) of the IEEE Vehicular Technology Society.

Qingqi Pei received his B.S., M.S. and Ph.D. degrees in Computer Science and Cryptography from Xidian University, in 1998, 2005 and 2008, respectively. He is now a Professor and member of the State Key Laboratory of Integrated Services Networks, also a Professional Member of ACM and Senior Member of IEEE, Senior Member of Chinese Institute of Electronics and China Computer Federation. His research interests focus on privacy preserving, blockchain and edge computing security.



Xiaoli Chu (M06CSM15) received the B.Eng. degree in electronic and information engineering from Xian Jiao Tong University, Xian, China, in 2001, and the Ph.D. degree in electrical and electronic engineering from the Hong Kong University of Science and Technology, Hong Kong, in 2005. She is a Senior Lecturer with the Department of Electronic and Electrical Engineering, University of Sheffield, Sheffield, U.K. From September 2005 to April 2012, she was with the Centre for Telecommunications Research, Kings College London. She has published more than 100 peer-reviewed journal and conference papers. She is the Lead Editor/author of the book Heterogeneous Cellular Networks: Theory, Simulation and Deployment (Cambridge University Press, 2013) and the book 4G Femtocells: Resource Allocation and Interference Management (Springer 2013).



Jianbo Du received the B.S. degree and M.S. degree from Xi'an University of Posts and Telecommunications in 2007 and 2013, respectively, and the Ph.D. in communication and information systems at Xidian University, Xian, Shaanxi, China, in 2018. She is now a teacher with the department of Communication and Information Engineering, Xian University of Posts and Telecommunications. Her research interests include mobile edge computing, resource management, NOMA, deep reinforcement learning, convex optimization, stochastic network optimization and heuristic algorithms and their applications in wireless communications.

Li Zhu received the Ph.D. degree in traffic control and information engineering from Beijing Jiaotong University, Beijing, China, in 2012. He is currently a Faculty Member at Beijing Jiaotong University and a Visiting Scholar at Carleton University, Ottawa, ON, Canada, and The University of British Columbia, Vancouver, BC, Canada. His research interests include intelligent transportation systems, train-ground communication technology in communication base train ground communication systems, and cross layer design in train-ground communication systems.