Computing Resource Allocation for Hybrid Applications of Blockchain and Mobile Edge Computing

Yuqi Fan[1(✉)], Jun Zhang[1], Xu Ding[2], Zhifeng Jin[1], and Lei Shi[1]

[1] School of Computer Science and Information Engineering, Anhui Province Key Laboratory of Industry Safety and Emergency Technology, Hefei University of Technology, Hefei 230601, Anhui, China
{yuqi.fan,zhang-jun,2019170960,shilei}@hfut.edu.cn
[2] Institute of Industry and Equipment Technology, Hefei University of Technology, Hefei 230009, Anhui, China
dingxu@hfut.edu.cn

Abstract. In mobile edge computing (MEC), each user chooses and then offloads the task to an edge server, whereas data security is a concern in MEC due to the lack of trust between users and edge servers. Blockchain is introduced to provide a reliable environment for MEC. In blockchain-based MEC, edge servers are used as the nodes in both MEC and blockchain. After processing the users' tasks, the edge servers upload the results and other task-related information to the blockchain. The edge servers simultaneously execute two kind of tasks, i.e., the tasks offloaded by the users and the blockchain tasks. Therefore, the user offloading decision affects the processing latency of MEC tasks, and there is a trade-off between the resource allocation for MEC and blockchain tasks. However, most existing studies optimize the resource allocation for blockchain and MEC individually, which leads to the suboptimal performance of blockchain-based MEC. In this paper, we study the problem of user offloading decision and the computing resource allocation of edge servers for MEC and blockchain tasks, with the objective to minimize the total processing delay of MEC and blockchain tasks. We propose an algorithm for joint computing resource allocation for MEC and blockchain (JMB). Theoretical analysis proves that JMB is a 3.16-approximation algorithm. Simulation results show that JMB can effectively reduce the delay in blockchain-based MEC.

Keywords: Edge computing · Blockchain · Computing offloading · Resource allocation

This work is partially supported by Major Science and Technology Projects in Anhui Province of China (202003a05020009) and National Natural Science Foundation of China (62002097).

H. Gao et al. (Eds.): CollaborateCom 2023, LNICST 561, pp. 187–206, 2024.
https://doi.org/10.1007/978-3-031-54521-4_11

1 Introduction

With the rapid development of computation-intensive mobile applications such as voice control, face recognition, interactive games, etc., it is difficult for users with limited resources to obtain satisfactory user experience [17]. In order to improve the quality of service (QoS) of users, mobile edge computing (MEC) rises rapidly.

Data security is a concern in MEC due to the lack of trust between users and edge servers [9], since the edge servers and the network environments are not fully trusted in MEC. The data may belong to multiple owners, and the data stored in the edge servers may be modified or misused by unauthorized users [22]. As a promising solution for data security protection, blockchain is able to provide a reliable environment for MEC [10], since blockchain has the characteristics of decentralization, immutability, traceability, transparency, etc.

A blockchain-based MEC system consists of two subsystems, i.e., MEC and blockchain, as shown in Fig. 1. In the MEC subsystem, each user offloads the computation-intensive task to an edge server through wireless communication, and the edge server executes the MEC tasks offloaded by the users. A large number of tasks offloaded to an edge server result in long task execution time. A large amount of computing resources on an edge server lead to short task execution time. Therefore, a user needs to choose an appropriate edge server for offloading the task. In the blockchain subsystem, the edge server uploads the results and other task-related information to the blockchain. The edge servers treat the records uploaded from the MEC as transactions, which are verified and packaged into a block [14]. When the entire blockchain network reaches consensus, the new block is added to the blockchain.

In blockchain-based MEC, edge servers are used as the nodes in both MEC and blockchain, executing MEC tasks and blockchain tasks simultaneously. The execution of both kinds of tasks consumes computing resources, and hence the rational allocation of edge server computing resources is necessary to reduce the task processing latency of blockchain-based MEC. The problem of computing resource allocation in MEC [1,3,6,12] or blockchain [2,8,11,16,19] has been studied previously. However, the existing methods optimize the resource allocation for blockchain and MEC individually. That is, the resource allocation for one subsystem assumes that it exclusively occupies the computing resources of each edge server, which leads to the imbalance between blockchain and MEC. On the one hand, when the MEC occupies excessive computing resources, the execution of MEC tasks on the edge servers is fast. However, the blockchain shares few computing resources, rendering slow block generation, which results in the high latency and low throughput of the entire blockchain-based MEC. On the other hand, when the blockchain uses too many computing resources, the delay of MEC task execution increases, which leads to the slow generation of data to be packaged into the blocks. Accordingly, the generation of blocks becomes slow and the throughput of the whole blockchain-based MEC system is low. In both of the above two cases, the system throughput decreases, which will lead to the degradation of QoS of mobile users.

It is obvious that there is a trade-off between the resource allocation for MEC and blockchain tasks. Therefore, random resource allocation will lead to the suboptimal performance of blockchain-based MEC. In this paper, we study the user offloading decision and the computing resource allocation of edge servers for both MEC and blockchain tasks to effectively reduce the total latency in blockchain-based MEC. The main contributions of this paper are as follows:

1. In blockchain-based MEC, we formulate the problem of user offloading decision and the computing resource allocation of edge servers for MEC and blockchain tasks, with the aim to minimize the total processing delay of both MEC and blockchain tasks.
2. We propose an algorithm for joint computing resource allocation for MEC and blockchain (JMB), which consists of 5 steps. First, JMB relaxes the problem constraints to obtain the relaxed problem. Second, JMB solves the relaxed problem and obtains the fractional optimal solution. Third, JMB modifies the fractional optimal solution to obtain the feasible fractional solution. Fourth, JMB performs the clustering operation on the users and the edge servers according to the feasible fractional solution to form multiple initial clusters. Finally, JMB maps the users to the edge servers in each initial cluster and obtains the feasible integer solution satisfying the relaxed constraints. Theoretical analysis proves that JMB is a 3.16-approximation algorithm.
3. We conduct experiments through simulations, and the experimental results show that JMB can effectively reduce the delay in blockchain-based MEC.

The rest of this paper is organized as follows. Section 2 introduces the related work. The system model is formulated and analyzed in Sect. 3. Section 4 presents in detail the proposed algorithm and analyzes the algorithm theoretically. The experiments are given in Sect. 5, and Sect. 6 concludes the paper.

2 Related Work

In a blockchain-based MEC system, edge servers need to execute both MEC and blockchain tasks, and both kinds of tasks consume computing resources. Therefore, it is crucial to allocate the computing resources of edge servers reasonably to reduce the total system delay.

In terms of computing resource allocation for MEC, Rahma et al. [1] proposed a blockchain-based distributed MEC framework to support low-latency, secure and anonymous data communication in on-demand data sharing scenarios. Liu et al. [12] proposed an adaptive block size based framework, which considers two offloading modes, namely offloading to a server or a nearby device, and formulated the issues of resource allocation, scheduling of offloading, and adaptive block size as an optimization problem. Cui et al. [3] proposed a blockchain based containerized edge computing platform for Internet of Vehicles. The platform was integrated with blockchain to improve the security of the network. A heuristic container scheduling algorithm was developed to schedule computing tasks to appropriate edge servers to reduce the computing latency. He et al. [7]

proposed a blockchain based framework for edge computing resource allocation in Internet of Things (IoT). The framework specifies the step-by-step process of a single transaction between IoT endpoints and edge servers. Furthermore, the work designed a smart contract in a private blockchain network that utilizes a machine learning algorithm, Asynchronous Advantage Actor-Critic (A3C), to allocate edge computing resources. Fan et al. [6] formulated a Stackelberg game with the cloud/edge computing service provider (CESP) as the leader and users as the followers for cloud/edge computing resource management. They also modeled the resource allocation and pricing at the CESP as a mixed-integer programming problem (MIP) with the objective to optimize the CESP's revenue and proposed an iterative greedy-and-search based resource allocation and pricing algorithm.

In terms of resource allocation for blockchain, Kang et al. [9] proposed a reputation-based data sharing scheme in the vehicle edge network, which introduced consortium blockchain and smart contracts to achieve secure data storage and prevent unauthorized data sharing. Liu et al. [11] modeled the joint optimization problem of mining task offloading and cryptographic hash caching for blocks, and proposed an alternating direction method of multipliers for the problem. Sharma et al. [16] proposed a new blockchain-based distributed mobility management scheme to meet the distributed security requirements of fog networks. Jiao et al. constructed an auction-based market model to determine the computing offloading strategy for miners and the allocation of computing resources to edge servers [8]. Xiao et al. [19] proposed a blockchain edge network trust mechanism and an edge server computing resource allocation algorithm based on reinforcement learning. Chang et al. [2] proposed an edge computing based blockchain incentive mechanism for miners to purchase edge server computing resources, and established a two-stage Stackelberg game between miners and edge servers.

In a blockchain-based MEC system, the execution of both MEC and blockchain tasks require computing resources. The random user offloading decision and the arbitrary resource allocation for MEC and blockchain will lead to a large system processing delay. Therefore, it is crucial to allocate computing resources of edge servers reasonably. However, most of the current research optimizes the resource allocation for blockchain and MEC individually, and little literature jointly considers the optimization of blockchain and MEC, which results in the suboptimal performance of blockchain-based MEC. In this paper, we study the problem of the user offloading decision and the computing resource allocation of edge servers for both MEC and blockchain tasks to effectively reduce the total latency in blockchain-based MEC.

3 Problem Definition

3.1 System Model of MEC and Blockchain Task Execution

In a blockchain-based MEC system, there are multiple users (IoT devices, sensors, wearables, etc.) and multiple edge servers. Each user needs to run a

computation-intensive task. The user chooses and then offloads the task to an edge server. The edge servers process the tasks offloaded by the users and simultaneously act as blockchain nodes communicating in a P2P manner [4] to form a blockchain. After executing the task offloaded by the user, the edge server treats the task execution result and other task-related information as a "transaction" to be recorded in the blockchain [14]. Multiple transactions are packaged into a block, and the edge server broadcasts the generated block to all the blockchain nodes which verify the received block. We generally call block generation and

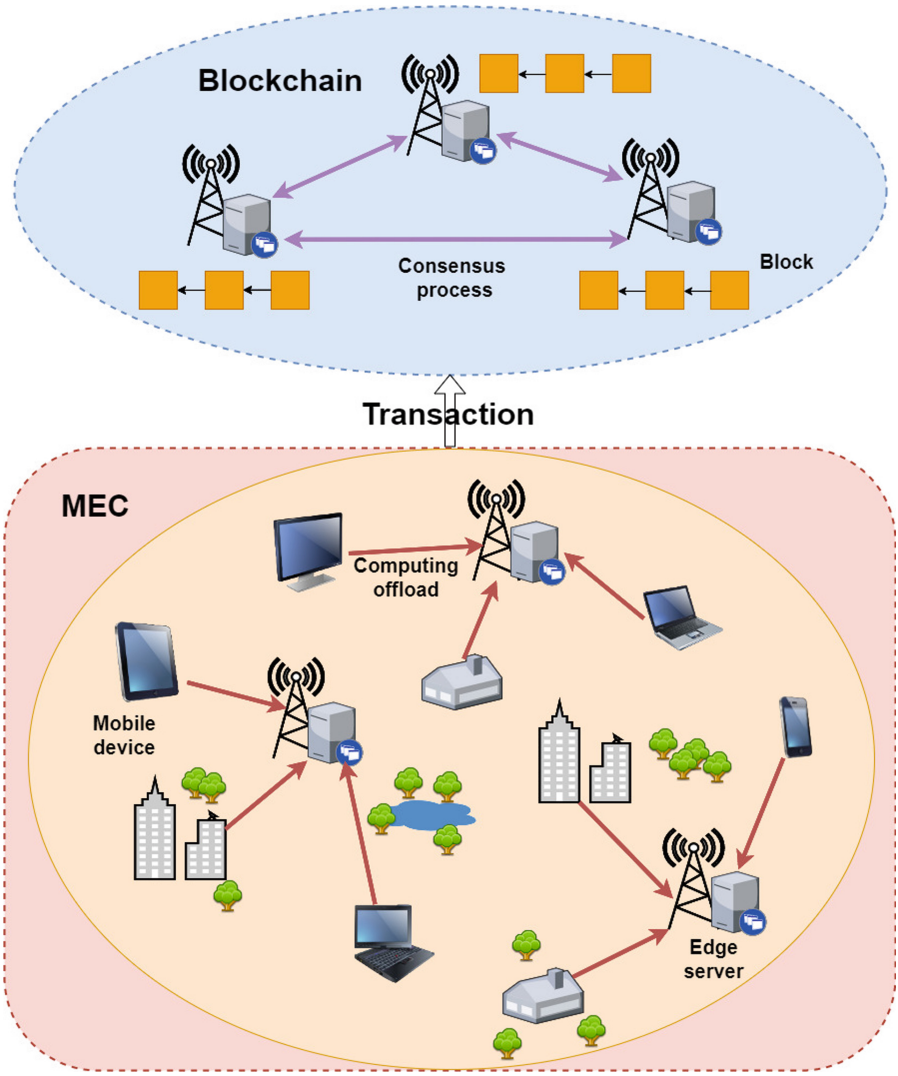

Fig. 1. System model.

consensus as blockchain tasks. Therefore, each edge server needs to complete two kinds of tasks: the offloaded tasks from the users in the MEC and the blockchain tasks, as shown in Fig. 1. The symbols used in this paper are shown in Table 1.

Table 1. Table of symbols and notations

Notation	Definition
u	User u
m	Edge servers m
U	User set
M	Edge servers set
l_u	The size of the computing task of user u
f_m	CPU clock cycles required by edge server m to process 1 bit of data in user computing tasks
$x_{u,m}$	User u's offloading decision regarding edge server m (0: the task of u is not offload to edge server m, 1: otherwise)
B_m	Bandwidth allocated by edge server m to the connected users
$p_{u,m}$	Transmission power from user u to edge server m
$g_{u,m}$	Channel gain between user u and edge server m
θ^2	Noise power
y_m	Proportion of computing resources obtained by the MEC task on edge server m (value 0 to 1)
F	CPU cycle frequency of edge server
S_b	Size of the blocks in the blockchain
C	Transfer rate of the link in the blockchain network
ω	Correlation coefficient of sizes of user tasks and blockchain transactions
K	Total number of blocks generated by the blockchain
$T_{u,m}$	Total latency of MEC tasks
$T^t_{u,m}$	Transmission delay of user u sending the task to edge server m
$T^c_{u,m}$	Execution time of the computing task of user u on edge server m
T_k	Total latency of blockchain tasks
T_g	Block generation time
T_t	Block transmission time

MEC Task Execution. A large number of tasks offloaded to an edge server result in long task execution time. Therefore, a user needs to choose an appropriate edge server for task offloading. Assuming that user u offloads the task with size l_u to server m, the task execution delay is calculated as

$$T_{u,m} = x_{u,m}(T^t_{u,m} + T^c_{u,m}) \tag{1}$$

where $T^t_{u,m}$ represents the transmission delay of user u sending data to edge server m, and $T^c_{u,m}$ denotes the execution time of the task offloaded by user u to edge server m.

We use $p_{u,m}$ to denote the transmission power from user u to edge server m, and use $g_{u,m}$ to represent the channel gain between user u and edge server m. The transmission rate from user u to edge server m is calculated by (2) [20]:

$$r_{u,m} = B_m \log_2(1 + \tfrac{p_{u,m} g_{u,m}}{\theta^2}) \tag{2}$$

where B_m is the network bandwidth between edge server m and user u, and θ^2 is the noise power. The transmission delay of user u sending data to edge server m is:

$$T^t_{u,m} = \tfrac{l_u}{r_{u,m}} \tag{3}$$

A large amount of computing resources on an edge server lead to short task execution time. The execution time of the offloaded task by user u on edge server m is calculated as [20]:

$$T^c_{u,m} = \tfrac{l_u f_m}{y_m F} \tag{4}$$

where F is the CPU cycle frequency of the edge server, f_m is the CPU clock cycles required by edge server m to process 1 bit of data in the tasks, and y_m is the proportion of computing resources allocated to the MEC tasks on edge server m. A large y_m leads to fast MEC task processing.

Blockchain Task Execution. The total delay of data processing in the blockchain includes the delay of block generation and block transmission, which is as follows:

$$T_k = T_g + T_t \tag{5}$$

where T_g is the time required for the blockchain to generate a new block and T_t is the block transmission time during the consensus process.

The time for the blockchain to generate a new block is calculated as:

$$T_g = \tfrac{1}{K} \sum_{m \in M} \sum_{u \in U} \tfrac{x_{u,m} l_u f_m \omega}{(1 - y_m) F} \tag{6}$$

where ω is the correlation coefficient between the MEC tasks and the blockchain tasks. More tasks offloaded to an edge server leads to larger MEC task processing latency and more blockchain transactions. A large y_m results in slow blockchain task processing, although it can reduce the MEC task running time.

In the blockchain, we assume that the time of data transmission during consensus is:

$$T_t = \frac{S_b}{C} \tag{7}$$

where S_b is the size of a block, and C denotes the data transmission rate on the link between the edge servers.

3.2 Problem Model

Each edge server in the system is not only an entity that runs tasks in the MEC, but also a node in the blockchain. There is a trade-off between the resource allocation for MEC and blockchain tasks, and the user offloading decision also affects the processing latency of MEC tasks. In this paper, we minimize the total processing delay of offloaded tasks in the MEC and blockchain tasks by deciding the offloading choices of computing tasks for users and the allocation of computing resource of edge servers for MEC and blockchain tasks. That is, our objective is to

$$\begin{aligned} &\min_{x_{u,m}, y_m} \Big(\sum_{m \in M} \sum_{u \in U} T_{u,m} + \sum_{k \in K} T_k \Big) \\ &s.t. \\ &\quad \text{(C1)}: \sum_{m \in M} x_{u,m} = 1, \forall u \in U \\ &\quad \text{(C2)}: x_{u,m} \in \{0, 1\}, \forall u \in U, m \in M \\ &\quad \text{(C3)}: 0 < y_m < 1, \forall m \in M \end{aligned} \tag{8}$$

Constraint (C1) requires that the computing task of each user is offloaded to one and only one edge server. Constraint (C2) dictates the mapping relationship between user u and edge server m, where 1 means that the task of user u is offloaded to edge server m and 0 otherwise. Constraint (C3) demands that the proportion of computing resource of each edge server allocated for MEC is in the range of (0, 1).

4 Algorithm for Joint Computing Resource Allocation for MEC and Blockchain

It is known that the problem of user offloading decision and computing resource allocation of edge servers in MEC is NP-hard. The problem in this paper needs to consider the resource allocation for both MEC and blockchain. Therefore, the problem in this paper is also NP-hard.

In this section, we propose JMB for the problem of user offloading decision and the computing resource allocation of edge servers for MEC and blockchain tasks. JMB consists of 5 steps. First, JMB relaxes the problem constraints to obtain the relaxed problem. Second, JMB solves the relaxed problem and obtains the fractional optimal solution. Third, JMB modifies the fractional optimal solution to obtain the feasible fractional solution. Fourth, JMB performs the clustering operation on the users and the edge servers according to the feasible

fractional solution to form multiple initial clusters. Finally, JMB maps the users to the edge servers in each initial cluster and obtains the feasible integer solution satisfying the relaxed constraints.

4.1 User's Offloading Decision and Edge Server's Computing Resource Allocation Algorithm

We take (1) and (5) into (8), and rewrite the problem model (8) as (9).

$$\begin{aligned} &\min_{x_{u,m},y_m} \sum_{m\in M}\sum_{u\in U}\left(\frac{l_u x_{u,m}}{r_{u,m}}+\frac{l_u f_m x_{u,m}}{y_m F}+\frac{l_u f_m \omega x_{u,m}}{(1-y_m)F}\right)+K\frac{S_b}{C} \\ &s.t. \\ &\text{(C1)}: \sum_{m\in M} x_{u,m}=1, \forall u\in U \\ &\text{(C2)}: x_{u,m}\in\{0,1\}, \forall u\in U, m\in M \\ &\text{(C3)}: 0<y_m<1, \forall m\in M \end{aligned} \tag{9}$$

The objective function (9) includes both integer variables $x_{u,m}$ and continuous variables y_m. Therefore, the problem is a mixed integer nonlinear programming problem, which is difficult to solve. We introduce a new variable z_m, and let

$$z_m=\frac{\sum\limits_{u\in U} x_{u,m}l_u}{y_m}+\frac{\sum\limits_{u\in U} x_{u,m}l_u\omega}{1-y_m}, \forall u\in U, \forall m\in M \tag{10}$$

The problem defined by (9) can then be transformed into:

$$\begin{aligned} &\min_{x_{u,m},z_m} \sum_{m\in M}\sum_{u\in U}\frac{l_u x_{u,m}}{r_{u,m}}+\sum_{m\in M}\frac{f_m z_m}{F}+K\frac{S_b}{C} \\ &s.t. \\ &\text{(C1)}: \sum_{m\in M} x_{u,m}=1, \forall u\in U \\ &\text{(C2)}: x_{u,m}\in\{0,1\}, \forall u\in U, m\in M \end{aligned} \tag{11}$$

We obtain $z_m=0$, if and only if $\sum\limits_{u\in U} x_{u,m}l_u=0$. We can get $\sum\limits_{u\in U} x_{u,m}l_u>0$ and $\sum\limits_{u\in U} x_{u,m}l_u\omega>0$ because $x_{u,m}\in\{0,1\}$, $\omega\in(0,1]$ and $l_u>0$ ($\forall u\in U, m\in M$), when $\sum\limits_{u\in U} x_{u,m}l_u\neq 0$. We can obtain $z_m\geq\left(\sqrt{\omega}+1\right)^2\sum\limits_{u\in U} x_{u,m}l_u$ because $0<y_m<1, \forall m\in M$, and $\omega\in(0,1]$. Without loss of generality, let $\omega=1$ and we can obtain $z_m\geq 4\sum\limits_{u\in U} x_{u,m}l_u$.

Taking the value range of z_m into (11), and we can obtain the mixed-integer linear programming problem defined as (12).

$$\min_{x_{u,m},z_m} \left(\sum_{m\in M}\sum_{u\in U} d_{u,m}x_{u,m} + \sum_{m\in M} a_m z_m + e \right)$$
$$s.t.$$
$$\begin{aligned} &(\text{C1}): \sum_{m\in M} x_{u,m} = 1, \forall u \in U \\ &(\text{C2}):4 \sum_{u\in U} x_{u,m} l_u \le z_m \\ &(\text{C3}): x_{u,m} \in \{0,1\}, \forall u \in U, m \in M \\ &(\text{C4}): z_m \ge 0, \forall m \in M \end{aligned} \tag{12}$$

where $a_m = \frac{f_m}{F}$, $d_{u,m} = \frac{l_u}{r_{u,m}}$ and $e = K\frac{S_b}{C}$.

We relax the integer variable $x_{u,m}$ to obtain the linear programming problem (13).

$$\min_{x_{u,m},z_m} \left(\sum_{m\in M}\sum_{u\in U} d_{u,m}x_{u,m} + \sum_{m\in M} a_m z_m + e \right)$$
$$s.t.$$
$$\begin{aligned} &(\text{C1}): \sum_{m=1}^{M} x_{u,m} = 1, \forall u \in U \\ &(\text{C2}):4 \sum_{u\in U} x_{u,m} l_u \le z_m \\ &(\text{C3}): x_{u,m} \ge 0, z_m \ge 0, \forall u \in U, m \in M \end{aligned} \tag{13}$$

Given a feasible fractional solution (x, z) to the linear programming problem (13) and a parameter $\alpha \in (0, 1)$, we define $d_u(\alpha)$ for any user u as follows: Consider the ordering π such that $d_{\pi(1)u} \le d_{\pi(2)u} \le \ldots \le d_{\pi(M)u}$, $d_u(\alpha) := d_{\pi(m^*)u}$, where $m^* := \min\{m : \sum_{i=1}^{m} x_{\pi(i)u} \ge \alpha\}$. If edge server m and user u satisfy $0 < x_{u,m} < 1$ in the fractional solution (x, z), edge server m is fractionally connected to user u.

The edge servers are sorted in the non-ascending order of the distance to user u, and $\pi(i^*)$ is the first edge server satisfying $\sum_{i=1}^{m} x_{\pi(i)u} > \alpha$.

We denote $F_{u'}$ as the edge server fractionally connected with user u', and $G_{u'}$ (including u') as the set of users fractionally connected with the edge server in $F_{u'}$.

Algorithm 1 shows the process of JMB. First, JMB solves the relaxed problem (Lines 1–2). JMB randomly chooses α which is uniformly distributed in $(\beta, 1)$, and solves the linear programming problem (11) (such as the simplex method) to obtain the optimal solution (x^*, z^*). Second, JMB modifies the fractional optimal solution (Lines 3–16). For each user u, JMB computes $d_u(\alpha)$, where $d_u(\alpha) := d_{\pi(m*)u}$ and $m* := \min\{m : \sum_{i=1}^{m} x_{\pi(i)u} \ge \alpha\}$, and then computes β_u^{α} for

Algorithm 1. JMB

Input: Parameter $\beta \in (0, 1)$.
Output: Integer feasible solution $(\hat{x}, \hat{z})$.

1: Randomly choose α that follows a uniform distribution on $(\beta, 1)$;
2: Solve problem (13) to obtain (x^*, z^*);
3: **for** $u \in U$ **do**
4: Calculate $d_u(\alpha)$;
5: **end for**
6: **for** $u \in U$ **do**
7: Compute $\beta_u^\alpha := \sum_{i=1}^{m^*} x^*_{\pi(i)u} \geq \alpha$;
8: **for** $m \in M$ **do**
9: **if** $d_{u,m} \leq d_u(\alpha)$ **then**
10: $\bar{x}_{u,m} := \frac{x^*_{u,m}}{\beta_u^\alpha}$;
11: **else**
12: $\bar{x}_{u,m} := 0$;
13: **end if**
14: **end for**
15: **end for**
16: $\bar{z}_m := z^*_m$;
17: **for** $u \in U$ **do**
18: $F_u := \{m \in M : \bar{x}_{u,m} > 0\}$;
19: **end for**
20: $U' := \varnothing$;
21: **while** $U \neq \varnothing$ **do**
22: $u' := \arg\min \{d_u(\alpha)\}$, where u' is the cluster center;
23: $U' := U' \cup \{u'\}$;
24: $G_{u'} := \{u \in U : F_u \cap F_{u'} \neq \varnothing\}$;
25: $U := U \backslash G_{u'}$;
26: **end while**
27: **for** $u' \in U'$ **do**
28: **for** $u \in G_{u'}$ **do**
29: Connect u to the nearest edge server, and the corresponding integer feasible solution is denoted as $(\hat{x}, \hat{z})$;
30: **end for**
31: **end for**
32: **return** $(\hat{x}, \hat{z})$.

each user u based on $\beta_u^\alpha := \sum_{i=1}^{m^*} x^*_{\pi(i)u} \geq \alpha$. JMB modifies the fractional optimal solution (x^*, z^*) by comparing $d_{u,m}$ and $d_u(\alpha)$, and obtains the corresponding feasible fractional solution $(\bar{x}, \bar{z})$. Third, JMB forms the initial clusters (Lines 17–26). For each user u, JMB computes the set F_u of edge servers fractionally connected with u. In the unassigned user set U, JMB selects user u' with the smallest $d_u(\alpha)$ as the cluster center. JMB works out the set $G_{u'}$ of users fractionally connected with the edge server in $F_{u'}$, and removes the users in set $G_{u'}$ from U. We call $(\{u'\}, F_{u'}, G_{u'})$ the initial cluster centered on u'. The process of the

third stage is repeated until U is empty (each node is put into an initial cluster). Finally, JMB maps the users to the edge servers (Lines 27–31). For each cluster, JMB connects the users in each initial cluster to the nearest edge server in the cluster to obtain a feasible integer solution that satisfies the relaxed constraints.

4.2 Algorithm Analysis

Based on the feasible solution obtained by Algorithm 1, we denote the total computing latency of edge servers as Z ($Z = \sum_{m \in M} a_m \hat{z}_m$) and the total network delay caused by task offloading as X ($X = \sum_{m \in M} \sum_{u \in U} d_{u,m} x_{u,m}$).

Lemma 1. *In the computing resource allocation scheme obtained by Algorithm 1, the total computing latency of edge servers satisfies:*

$$Z \leq \frac{1}{\alpha} \sum_{m \in M} a_m z_m^* \tag{14}$$

Proof. From Algorithm 1, it can be concluded that $\{F_{u'}\}_{u' \in U'}$ is mutually disjoint. Each edge server exists independently, and $(\bar{x}, \bar{z})$ is feasible. Therefore,

$$\begin{aligned} Z &= \sum_{u' \in U'} a_m \leq \sum_{u' \in U'} \sum_{m \in F_{u'}} a_m \bar{x}_{u',m} < \sum_{u' \in U'} \sum_{m \in F_{u'}} a_m \bar{z}_m \\ &< \frac{1}{\alpha} \sum_{u' \in U'} \sum_{m \in F_{u'}} a_m z_m^* \leq \frac{1}{\alpha} \sum_{m \in M} a_m z_m^* \end{aligned} \tag{15}$$

The lemma is proven.

It can be seen from Algorithm 1 that the number of edge servers is always no less than the number of clusters, and each cluster has users offloading the computing tasks to the edge servers. The number of edge servers in a cluster is no less than 1, and always less than the number of users.

Lemma 2. $d_u(\alpha) \leq \frac{1}{1-\alpha} \sum_{m \in M} d_{u,m} x_{u,m}^*$.

Proof. From the definition of $d_u(\alpha)$, we obtain:

$$\sum_{i=m^*}^{M} x_{\pi(i)u}^* \geq 1 - \alpha \tag{16}$$

Therefore,

$$\begin{aligned} d_u(\alpha) &\leq \frac{1}{1-\alpha} \sum_{i=m^*}^{M} d_{\pi(i)u} x_{\pi(i)u}^* \leq \frac{1}{1-\alpha} \sum_{i=1}^{M} d_{\pi(i)u} x_{\pi(i)u}^* \\ &= \frac{1}{1-\alpha} \sum_{m \in M} d_{u,m} x_{u,m}^* \end{aligned} \tag{17}$$

The lemma is proven.

Lemma 3. *In the computing resource allocation scheme obtained by Algorithm 1, the total network delay caused by task offloading satisfies*

$$X \leq \frac{3}{1-\alpha} \sum_{m \in M} \sum_{u \in U} d_{u,m} x^*_{u,m} \tag{18}$$

Proof. For any user $u \in U$, we consider the following two cases.

(1) Case 1: $u \in U'$. We connect u to the edge server with the least network delay, and hence the network delay incurred by the task offloading decision of u does not exceed $d_u(\alpha)$.

(2) Case 2: $u \notin U'$. It can be obtained from Step 4 of Algorithm 1 that there exists $u \in U'$, $\tilde{m} \in N_{u'} \cap N_u$ and $u \in G_{u'}$. We connect u to m'. According to the triangle inequality and Lemma 2, we can know that the network delay incurred by the task offloading decision of u, i.e., $d_{m',u}$, satisfies

$$\begin{aligned} d_{m',u} &\leq d_{m',u'} + d_{\tilde{m},u'} + d_{\tilde{m},u} \leq 2d_{u'}(\alpha) + d_u(\alpha) \leq 3d_u(\alpha) \\ &\leq \frac{3}{1-\alpha} \sum_{m \in M} d_{u,m} x^*_{u,m} \end{aligned} \tag{19}$$

The lemma is proven from the above two cases.

From Lemma 1 and Lemma 3, we can derive:

$$Z + X + e \leq \frac{1}{\alpha} \sum_{m \in M} a_m z^*_m + e + \frac{3}{1-\alpha} \sum_{m \in M} \sum_{u \in U} d_{u,m} x^*_{u,m} \tag{20}$$

Lemma 4. *For any user u, we have*

$$\int_0^1 d_u(\alpha) d\alpha = \sum_{m=1}^{M} d_{u,m} x^*_{u,m} \tag{21}$$

Proof. Without loss of generality, we assume $d_{u,1} \leq d_{u,2} \leq ... \leq d_{u,M}$, and let $m_1 < m_2 < ... < m_i$ be all the edge servers satisfying the condition of $x^*_{u,m} > 0$. Let $x^*_{u,m_0} = 0$, and then function $d_u(\alpha)$ is a step function.

$$d_u(\alpha) = d_{u,m_k}, \alpha \in (\sum_{s=0}^{k-1} x^*_{u,m_s}, \sum_{s=1}^{k} x^*_{u,m_s}], k = 1, ..., i \tag{22}$$

We verify (22) through the following two value ranges of α.

First, we assume that the value of α satisfies $\alpha \in (0, x^*_{u,m_1}]$. According to the definitions $d_u(\alpha) := d_{\pi(m*)u}$ and $m* := \min\{m : \sum_{i=1}^{m} x_{\pi(i)u} \geq \alpha\}$ of $d_u(\alpha)$, we know $m_1 = \min\{m : \sum_{i=1}^{m} x_{\pi(i)u} \geq \alpha\}$. Therefore, the value of $d_u(\alpha)$ is d_{u,m_1} and the value of k is 1, satisfying (22).

Second, we assume that the value of α satisfies $\alpha \in (x^*_{u,m_1}, x^*_{u,m_1} + x^*_{u,m_2}]$. According to the definitions of $d_u(\alpha)$, we know $m_2 = \min\{m : \sum_{i=1}^{m} x_{\pi(i)u} \geq \alpha\}$. Therefore, the value of $d_u(\alpha)$ is d_{u,m_2} and the value of k is 2, satisfying (22).

The analysis of the other values of α within $(0, 1)$ is similar to that above, and we omit the analysis.

According to the geometric meaning of definite integral, we derive

$$\int_0^1 d_u(\alpha)d\alpha = \sum_{k=1}^{i} d_{u,m_k} x^*_{u,m_k} = \sum_{m=1}^{M} d_{u,m} x^*_{u,m} \tag{23}$$

The lemma is proven.

Theorem 1. *Let $\beta = e^{-3}$, and the expected delay of the feasible solution obtained by Algorithm 1 does not exceed 3.16 times the optimal solution to the problem defined as (12).*

Proof. For a given α from Lemma 1 and Lemma 3, we know that the delay of the solution obtained by Algorithm 1 dose not exceed $\frac{1}{\alpha}\sum_{m\in M} a_m z^*_m + 3\sum_{u\in U} d_u(\alpha)$.

Therefore, the expectation of the delay of the solution obtained by Algorithm 1 does not exceed

$$\begin{aligned}
&E[\frac{1}{\alpha}\sum_{m\in M} a_m z^*_m + 3\sum_{u\in U} d_u(\alpha)] \\
&= E[\frac{1}{\alpha}]\sum_{m\in M} a_m z^*_m + 3\sum_{u\in U} E[d_u(\alpha)] \\
&= \int_\beta^1 \frac{1}{1-\beta}\frac{1}{\alpha}d\alpha \sum_{m\in M} a_m z^*_m + 3\sum_{u\in U}(\int_\beta^1 \frac{1}{1-\beta} d_u(\alpha)d\alpha) \\
&\leq \frac{\ln(1/\beta)}{1-\beta}\sum_{m\in M} a_m z^*_m + \frac{3}{1-\beta}\sum_{u\in U}\int_0^1 d_u(\alpha)d\alpha \\
&= \frac{\ln(1/\beta)}{1-\beta}\sum_{m\in M} a_m z^*_m + \frac{3}{1-\beta}\sum_{u\in U}\sum_{m\in M} d_{u,m} x^*_{u,m} \\
&\leq \max\{\frac{\ln(1/\beta)}{1-\beta}, \frac{3}{1-\beta}\}(\sum_{m\in M} a_m z^*_m + \sum_{u\in U}\sum_{m\in M} d_{u,m} x^*_{u,m})
\end{aligned} \tag{24}$$

When $\beta = e^{-3}$, we have

$$\frac{\ln(1/\beta)}{1-\beta} = \frac{3}{1-\beta} = \frac{3}{1-e^{-3}} \leq 3.16 \tag{25}$$

For any user u, $d_u(\alpha)$ is a step function. By examining the function values at all segment points, α can be chosen such that $\frac{1}{\alpha}\sum_{m\in M} a_m z^*_m + 3\sum_{u\in U} d_u(\alpha)$ reaches the minimum value, and we obtain

$$\frac{1}{\alpha}\sum_{m\in M} a_m z_m^* + 3\sum_{u\in U} d_u(\alpha)$$
$$\leq E[\frac{1}{\alpha}\sum_{m\in M} a_m z_m^* + 3\sum_{u\in U} d_u(\alpha)] \leq 3.16(\sum_{m\in M} a_m z_m^* + \sum_{m\in M}\sum_{u\in U} d_{u,m} x_{u,m}^*) \tag{26}$$

The theorem is proven.

Theorem 2. *Algorithm 1 is a 3.16-approximation algorithm for the problem of user offloading decision and the computing resource allocation of edge servers for MEC and blockchain tasks.*

Proof. According to the feasible integer solution $(\hat{x}, \hat{z})$ obtained by Algorithm 1, the values of all $x_{u,m}$ and z_m can be known. We can calculate the value of y_m by taking z_m into (10). In the interval $(0, 1)$, one z_m value may correspond to two y_m values at the same time. If and only if y_m has two values in the interval $(0, 1)$, without loss of generality, we choose to take the larger number of the two, since taking either of the two corresponding values of y_m has no influence on the feasible solution $(\hat{x}, \hat{z})$. Therefore, we can obtain the feasible solution to the problem defined as (8). That is, Algorithm 1 is a 3.16-Approximate algorithm for the problem of user offloading decision and the computing resource allocation of edge servers for MEC and blockchain tasks.

The theorem is proven.

5 Performance Evaluation

5.1 Experimental Setup

In the experiments, we consider a blockchain-based MEC system, which includes 40 users and 5 edge servers. The parameters of the system are shown in Table 2. We evaluate the performance of JMB by using the following 2 benchmark algorithms: FMB and RMB.

(1) FMB [18]: The edge servers allocate computing resources to MEC and blockchain tasks at a fixed ratio, i.e., (0.7:0.3), and FMB uses the users' offloading decision obtained by JMB.
(2) RMB [20]: Each user randomly selects an edge server to offload the task and the edge servers allocate computing resources in the same way as JMB.

5.2 Total Processing Latency of MEC and Blockchain Tasks

Figure 2 shows the impact of different numbers of users on the total system latency caused by each algorithm, when the number of users varies from 20 to 100. It can be observed that the total delay incurred by JMB, FMB and RMB

Table 2. Table of simulation parameters

Simulation Parameters	Ranges
Bandwidth B_m	179 KHz - 180 KHz [5]
Computing task size l_u	200 KB-1000 KB [13]
Noise power θ^2	-174 dBm/Hz [15]
The transmission rate of the link in the blockchain network C	$15 * 10^6$ bit/s [15]
The number of CPU cycles required for 1 bit f_m	995 cycle/bit - 1000 cycle/bit [13]
Block size, S_b	8 MB [13]
The CPU cycle frequency of edge server, F	16 GHz [21]

increases with the increase of the number of users. More computing tasks need to be run with more users. Therefore, each edge server needs to execute more computing tasks, resulting in an increase in the total system latency.

In Fig. 2, both FMB and RMB incur a higher total latency than JMB at a given number of users. The total system latency obtained by JMB is approximately 92% of that by FMB and 93% of that by RMB. As the number of users increases, the number of computing tasks each edge server needs to process increases. RMB uses a random policy to make the user offloading decisions, and does not make a reasonable offloading decision based on the network latency between the users and the edge servers. Each edge server in FMB does not allocate the computing resources appropriately based on the processing capability of the edge servers and the task sizes. Therefore, as the number of computing tasks in the system increases, JMB can always execute more tasks in a given amount of time. Figure 3 shows the impact of the number of edge servers on the total latency in blockchain-based MEC, where the number of edge servers increases from 3 to 7. The results show the total latency of JMB and FMB decreases as the number of edge servers increases. The total computing power within the system

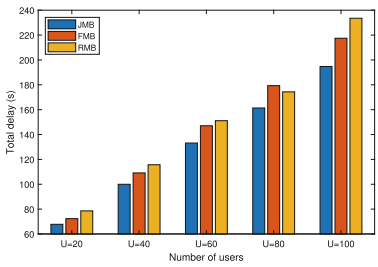

Fig. 2. The impact of the number of users on the total latency.

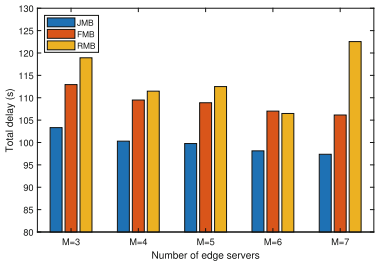

Fig. 3. The impact of the number of edge servers on total latency.

increases with the increasing number of the edge servers, and hence the system can execute more user tasks. Therefore, the total latency of the system decreases as the number of edge servers increases. However, as the number of edge servers continues to increase, the total latency decreases slowly, since the users have a limited resource requirement from the edge servers. Therefore, the reduce in the total latency is not significant with the continuous increase of the number of edge servers. RMB randomly selects the edge servers for the users, and the computing tasks are not necessarily offloaded to the newly added edge servers with the increasing number of edge servers. Therefore, the total system latency of RMB does not always decrease as the number of edge servers increases.

In Fig. 3, JMB always obtains the lowest total system delay. In addition, the total latency of JMB is about 9% and 8% lower than that of RMB and FMB, respectively. As the number of edge servers increases, the number of task offloading options available to the users in the system increases. Compared with the other two algorithms, JMB can offload the tasks to the edge servers that are appropriate for the whole system according to the users' tasks and network latency. For each edge server, a reasonable proportion of computing resources is allocated to the MEC and the blockchain tasks, respectively, according to the processing rate of the edge servers and the task sizes. Therefore, JMB can make better use of the edge servers than FMB and RMB. As a result, JMB obtains the lowest total system latency. Figure 4 shows the impact of users' computational tasks on the total latency incurred by each algorithm. It can be seen that the total delay of the three algorithms increases with the increase of users' computing task sizes. The edge servers need more time to process the users' tasks, when the sizes of users' tasks increase. As a result, the total latency also increases.

Figure 4 shows that the delay incurred by algorithm JMB is always less than that by FMB and RMB. The total latency of JMB is less than that of FMB by 5.0%, 7.1%, 8.3%, 9.1%, and 9.6%, and less than that of RMB by 8.0%, 11.5%, 16.4%, 17.2%, and 14.5%, when the task size is 200KB, 400KB, 600KB, 800KB, and 1000KB, respectively. JMB takes into account the sizes of users' tasks and the transmission rate between edge servers, when selecting edge servers for users to offload computing tasks. Therefore, with the increase of users' computing task sizes, JMB significantly reduces the transmission delay of the system compared with RMB. JMB allocates computing resources according to the sizes of the users' tasks processed by edge servers tasks. Compared with FMB, JMB allocates the computing resources more reasonably and efficiently.

Figure 5 shows the impact of the number of clock cycles that the edge server takes to process 1 bit of data (f_m) on the total latency of the blockchain-based MEC system. Figure 5 shows that the total delay of each algorithm increases with the increase of the number of clock cycles that the edge servers process 1 bit data. f_m affects the speed of task processing. A small value of f_m means that the edge server will spend less CPU cycles on processing users' and blockchain tasks. Therefore, the total latency obtained by each algorithm increases with the increase of the number of clock cycles of the edge servers processing 1 bit data.

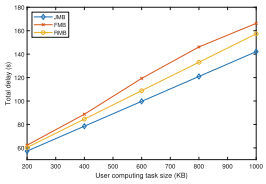

Fig. 4. The impact of user computing task size on total latency.

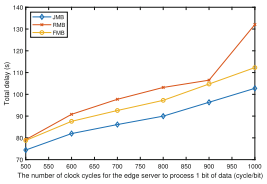

Fig. 5. The impact of the number of clock cycles that the edge server processes 1 bit data on the total delay.

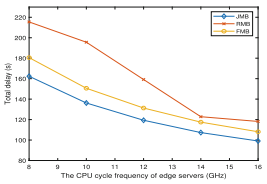

Fig. 6. The impact of CPU cycle frequency on the total system latency.

As can be seen from Fig. 5, the total system delay of JMB is lower than that of FMB and RMB. When f_m is 900 and 1000, the total system latency of JMB is respectively 8.0% and 8.5% lower than that of FMB. When f_m is 800 and 1000, JMB's total system latency is respectively 12.8% and 15.8% lower than RMB's. JMB makes the user offload decision based on f_m and the transmission rate of sending the user's task to the edge server. For each user, JMB arranges the most reasonable edge servers for task offloading, reducing the total delay of task transmission and task processing in the system. Therefore, the performance of RMB is inferior to that of JMB. In addition, JMB adjusts the computing resource allocation ratio based on f_m. As a result, JMB's total system latency is lower than FMB's.

Figure 6 illustrates the impact of the edge server's CPU cycle frequency on the total system delay. Obviously, the total latency of each algorithm decreases with the increase of the edge server CPU cycle frequency. The higher the CPU cycle frequency, the stronger the computing power of the edge server, and the faster the processing of MEC and blockchain tasks. As a result, the latency of each algorithm decreases as the CPU cycle frequency increases. Therefore, the total latency of each algorithm decreases with the increase of the edge server CPU cycle frequency.

Figure 6 demonstrates that the total system latency of JMB is always lower than that of FMB and RMB. JMB achieves lower total system latency than FMB by up to 10%. The total system latency of JMB is better than that of RMB by up to 30%. JMB optimizes users' offloading decisions and dynamically adjusts the computing resource allocation for the MEC and the blockchain tasks based on the processing rate of the edge servers and the transmission delay between the users and the edge servers. Therefore, JMB can reduce the network delay and processing latency of users' tasks in the blockchain-based MEC system, which makes JMB achieve better performance than RMB and FMB.

6 Conclusions

In this paper, we studied the problem of user offloading decision and the computing resource allocation of edge servers for MEC and blockchain tasks, with the

objective to minimize the total processing delay of MEC and blockchain tasks in blockchain-based MEC. We proposed an algorithm for joint computing resource allocation for MEC and blockchain (JMB) which consists of 5 steps. First, JMB relaxes the problem constraints to obtain the relaxed problem. Second, JMB solves the relaxed problem and obtains the fractional optimal solution. Third, JMB modifies the fractional optimal solution to obtain the feasible fractional solution. Fourth, JMB performs the clustering operation on the users and the edge servers according to the feasible fractional solution to form multiple initial clusters. Finally, JMB maps the users to the edge servers in each initial cluster and obtains the feasible integer solution satisfying the relaxed constraints. Theoretical analysis proved that JMB is a 3.16-approximation algorithm. The simulation results demonstrated that JMB could achieve the favorable balance between the MEC and the blockchain task processing. JMB obtains superior performance in terms of total system latency under different parameters of the number of users, the sizes of user tasks, the number of clock cycles that the edge server processes 1 bit data, the number of edge servers, and the CPU cycle frequency of edge servers.

References

1. Rahman, M.A., et al.: Blockchain-based mobile edge computing framework for secure therapy applications. IEEE Access **6**, 72469–72478 (2018)
2. Chang, Z., Guo, W., Guo, X., Zhou, Z., Ristaniemi, T.: Incentive mechanism for edge-computing-based blockchain. IEEE Trans. Industr. Inf. **16**(11), 7105–7114 (2020)
3. Cui, L., et al.: A blockchain-based containerized edge computing platform for the Internet of vehicles. IEEE Internet Things J. **8**(4), 2395–2408 (2020)
4. Demers, A.J., et al.: Epidemic algorithms for replicated database maintenance. In: Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing, pp. 1–12. ACM, Vancouver, British Columbia, Canada, August 1987
5. Du, J., Zhao, L., Chu, X., Yu, F.R., Feng, J., I, C.L.: Enabling low-latency applications in LTE-A based mixed fog/cloud computing systems. IEEE Trans. Veh. Technolo. 68(2), 1757–1771 (2018)
6. Fan, Y., Wang, L., Wu, W., Du, D.: Cloud/edge computing resource allocation and pricing for mobile blockchain: an iterative greedy and search approach. IEEE Trans. Comput. Soc. Syst. **8**(2), 1–13 (2021)
7. He, Y., Wang, Y., Qiu, C., Qiuzhen Lin, Li, J., Ming, Z.: Blockchain-based edge computing resource allocation in IoT: a deep reinforcement learning approach. IEEE Internet Things J. **8**(4), 2226–2237 (2020)
8. Jiao, Y., Wang, P., Niyato, D., Suankaewmanee, K.: Auction mechanisms in cloud/fog computing resource allocation for public blockchain networks. IEEE Trans. Parallel Distrib. Syst. **30**(9), 1975–1989 (2019)
9. Kang, J., et al.: Blockchain for secure and efficient data sharing in vehicular edge computing and networks. IEEE Internet Things J. **6**(3), 4660–4670 (2018)
10. Li, Z., Kang, J., Yu, R., Ye, D., Deng, Q., Zhang, Y.: Consortium blockchain for secure energy trading in industrial Internet of Things. IEEE Trans. Indus. Inform. **14**(8), 3690–3700 (2017)

11. Liu, M., Yu, F.R., Teng, Y., Leung, V.C.M., Song, M.: Computation offloading and content caching in wireless blockchain networks with mobile edge computing. IEEE Trans. Veh. Technol. **67**(11), 11008–11021 (2018)
12. Mengting Liu, Yu, F.R., Teng, Y., Leung, V.C.M., Song, M.: Distributed resource allocation in blockchain-based video streaming systems with mobile edge computing. IEEE Trans. Wirel. Commun. **18**(1), 695–708 (2018)
13. Liu, M., Yu, F.R., Teng, Y., Leung, V.C.M., Song, M.: Performance optimization for blockchain-enabled industrial Internet of things (IIoT) systems: a deep reinforcement learning approach. IEEE Trans. Indus. Inform. **15**(6), 3559–3570 (2019)
14. Ma, Z., Wang, X., Jain, D.K., Khan, H., Gao, H., Wang, Z.: A blockchain-based trusted data management scheme in edge computing. IEEE Trans. Indus. Inf. **16**(3), 2013–2021 (2020)
15. Mao, Y., Zhang, J., Song, S., Letaief, K.B.: Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems. IEEE Trans. Wirel. Commun. **16**(9), 5994–6009 (2017)
16. Sharma, V., You, I., Palmieri, F., Jayakody, D.N.K., Li, J.: Secure and energy-efficient handover in fog networks using blockchain-based DMM. IEEE Commun. Mag. **56**(5), 22–31 (2018)
17. Suankaewmanee, K., et al.: Performance analysis and application of mobile blockchain. In: 2018 International Conference on Computing. Networking and Communications, ICNC, pp. 642–646. IEEE Computer Society, Maui, HI, USA, March 2018
18. Tang, Q., Fei, Z., Zheng, J., Li, B., Guo, L., Wang, J.: Secure aerial computing: convergence of mobile edge computing and blockchain for UAV networks. IEEE Trans. Veh. Technol. **71**(11), 12073–12087 (2022)
19. Xiao, L., Ding, Y., Jiang, D., Huang, J., Wang, D., Li, J., Vincent Poor, H.: A reinforcement learning and blockchain-based trust mechanism for edge networks. IEEE Trans. Commun. **68**(9), 5460–5470 (2020)
20. Xu, S., et al.: Deep reinforcement learning assisted edge-terminal collaborative offloading algorithm of blockchain computing tasks for energy Internet. Int. J. Elect. Power Energy Syst. **131**, 107022 (2021)
21. Xu, Y., Zhang, H., Ji, H., Yang, L., Li, X., Leung, V.C.M.: Transaction throughput optimization for integrated blockchain and MEC system in IoT. IEEE Trans. Wirel. Commun. **21**(2), 1022–1036 (2021)
22. Yu, W., et al.: A survey on the edge computing for the Internet of Things. IEEE Access **6**, 6900–6919 (2018)