Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2024.047295

ARTICLE

Computing Resource Allocation for Blockchain-Based Mobile Edge Computing

Wanbo Zhang[1], Yuqi Fan[1], Jun Zhang[1], Xu Ding[2,*] and Jung Yoon Kim[3,*]

[1]School of Computer Science and Information Engineering, Intelligent Interconnected Systems Laboratory of Anhui Province, Hefei University of Technology, Hefei, 230601, China

[2]Anhui Province Key Lab of Aerospace Structural Parts Forming Technology and Equipment, Hefei University of Technology, Hefei, 230009, China

[3]College of Future Industry, Gachon University, Seongnam, 13120, South Korea

*Corresponding Authors: Xu Ding. Email: dingxu@hfut.edu.cn; Jung Yoon Kim. Email: kjyoon@gachon.ac.kr

Received: 01 November 2023 Accepted: 17 January 2024 Published: 16 April 2024

ABSTRACT

Users and edge servers are not fully mutually trusted in mobile edge computing (MEC), and hence blockchain can be introduced to provide trustable MEC. In blockchain-based MEC, each edge server functions as a node in both MEC and blockchain, processing users' tasks and then uploading the task related information to the blockchain. That is, each edge server runs both users' offloaded tasks and blockchain tasks simultaneously. Note that there is a trade-off between the resource allocation for MEC and blockchain tasks. Therefore, the allocation of the resources of edge servers to the blockchain and the MEC is crucial for the processing delay of blockchain-based MEC. Most of the existing research tackles the problem of resource allocation in either blockchain or MEC, which leads to unfavorable performance of the blockchain-based MEC system. In this paper, we study how to allocate the computing resources of edge servers to the MEC and blockchain tasks with the aim to minimize the total system processing delay. For the problem, we propose a computing resource Allocation algorithm for Blockchain-based MEC (ABM) which utilizes the Slater's condition, Karush-Kuhn-Tucker (KKT) conditions, partial derivatives of the Lagrangian function and subgradient projection method to obtain the solution. Simulation results show that ABM converges and effectively reduces the processing delay of blockchain-based MEC.

KEYWORDS

Mobile edge computing; blockchain; resource allocation

1 Introduction

Numerous mobile devices have become the indispensable instruments in people's work, study, and living due to the fast development of mobile networks [1]. Intelligent bracelets, smart garbage disposals, smart signal light controls, smart home appliances, and other intelligent applications have quickly garnered popularity and public attention. It is widely acknowledged that these intelligent applications have significantly improved social efficiency, raised human productivity, and improved people's quality of life [2,3]. Innumerable innovative applications call for considerable computing

power, which imposes a significant barrier for the IoT devices with limited resources and energy [4,5]. Cloud computing offers the users on-demand services through a pool of computing resources. However, cloud data centers are typically located far from mobile devices (users), and hence mobile cloud computing faces the issues of high latency, slow channel transmission, etc. [6,7].

Mobile Edge Computing (MEC) [8–10] has been developing to address the issues mentioned above by offloading computing tasks to edge servers. Therefore, the quality of service for task processing, including lower energy consumption and reduced execution delay, can be considerably enhanced by offloading computing workloads from users to edge servers [11,12]. However, the users and edge servers are not fully mutually trusted [13]. Blockchain is considered as a viable strategy to solve the problem of trust [14,15]. Blockchain uses community verification to synchronize a decentralized ledger replicated over several nodes, unlike conventional digital ledger systems that depend on a reliable central authority [16–18]. Accordingly, both academia and business are paying attention to building MEC based on blockchain [19].

In the blockchain-based MEC system shown in Fig. 1, each edge server functions as a node in both MEC and blockchain. As an MEC node, each edge server processes users' tasks and uploads the information about each task, e.g., user, time, task workload, results, etc., to the blockchain. As a blockchain node, the edge server treats the information of each task as a transaction and generates the blocks, each containing several transactions. Each transaction contains the encrypted file and the hash of the data and the result related to the task execution. The sources of the transactions in a block are all the users served by the edge server. That is, the transactions in a block may come from several users. After creating a block, the edge server propagates it to other edge servers which will verify the blocks. In this way, blockchain can provide trust between users and edge servers by transparent task-related information sharing. The transactions recording the information of the tasks are stored in blockchain, such that they cannot be modified. The anti-tampering characteristics of blockchain guarantee the authenticity and legitimacy of each transaction, thus allowing the users to check and track the details of the task related information. Each user can also evaluate the trustworthiness of other users based on their transaction history on the blockchain. If there is a dispute or controversy over a transaction, the transactions recorded in the blockchain can be used as a reliable evidence, which is used to resolve the dispute [20].

In blockchain-based MEC, each edge server runs both blockchain tasks and users' offloaded tasks simultaneously. Note that there is a trade-off between the resource allocation for MEC and blockchain tasks. Therefore, the resource allocation for the blockchain and the MEC is crucial for the processing delay of blockchain-based MEC. Most of the existing research tackles the problem of resource allocation in either MEC [21–23] or blockchain [24–26], which leads to the unfavorable performance of blockchain-based MEC. In this paper, we tackle the problem of how to allocate the computing resources of edge servers to the MEC and blockchain tasks with the objective of minimizing the total system processing delay. The contributions of this paper are as follows:

(1) We study the problem of resource allocation for both MEC and blockchain tasks to reduce the total processing delay of blockchain-based MEC, different from the existing research that focuses on only one of MEC and blockchain.

(2) We propose a computing resource Allocation algorithm for Blockchain-based MEC (ABM), which consists of four steps. First, we prove that the problem is convex. Second, we prove that the problem satisfies the Slater's condition, and hence the problem and its dual satisfy strong duality. Third, we use the Karush-Kuhn-Tucker (KKT) conditions and the partial derivatives of the Lagrangian

function to obtain the solution with the dual variables. Finally, we decide the values of the dual variables via the subgradient projection method, which obtains the solution to the problem.

(3) Simulation results demonstrate that ABM converges and effectively reduces the processing delay of blockchain-based MEC.

The rest of the paper is organized as follows. The related work is introduced in Section 2. The problem is defined in Section 3. Section 4 proposes the algorithm. In Section 5, the experimental results are presented. The paper is summarized in Section 6.

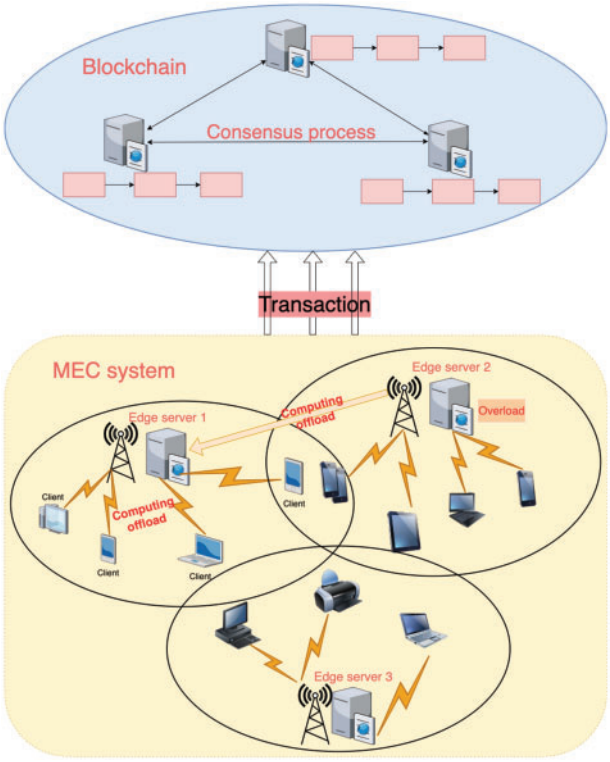

Figure 1: Blockchain-based MEC system

2 Related Work

In blockchain-based MEC, each edge server executes both MEC and blockchain tasks. Both of these two kinds of tasks consume computational resources, and there is a trade-off between the resource allocation for MEC and blockchain tasks. Therefore, the resource allocation for the blockchain and the MEC is crucial for the processing delay of blockchain-based MEC. Nevertheless, most of the existing research tackles the problem of resource allocation in either blockchain or MEC.

Some research studied the problem of computing resource allocation for MEC. Ren et al. [21] proposed a hybrid communication and computing resource allocation approach for delay minimization in a multi-user time-division multiple-access MEC offloading system. Nosrati et al. [27] proposed a community-based solution for the management of data replication based on the model of communication latency between computing and storage nodes. Feng et al. [28] proposed a method to maximize the user utility while reducing the operator's energy usage in a MEC system with multi-user

radios, by jointly optimizing the radio power allocation at the base station and the power allocation for processing users' offloaded data. Hong et al. [29] proposed an optimal resource allocation strategy that primarily addresses the issue of assigning MEC computing resources to a group of cars within a time window, so as to reduce the overall task processing delay. Liu et al. [30] proposed an edge framework that contains four groups under different locations between mobile edge nodes and users for the resource allocation problem to reduce the total cost of users and edge servers. Gao et al. [31] built a task offloading and resource allocation model which uses the overall computational cost as the objective function. Spallina et al. [22] proposed a deep Q-learning network for the resource allocation problem to reduce the task offloading energy usage while maintaining users' quality of service. Guo et al. [32] developed an online greedy heuristic algorithm for the resource optimization problem with the aim to maximize the long-term overall performability. Zhang et al. [33] investigated the user association and offloading decision problems for satelliteaerial integrated computing networks, where users can offload their tasks to high-altitude platforms (HAPs) and the satellite with edge computing servers. Zhang et al. [34] 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. Gao et al. [35] proposed the resource control algorithm to improve the system performance under the unstable workload in vehicular networks. The proposed resource control algorithm adjusts the priority and proportion of the resource utilization according to the system status obtained from the resource monitor, such as transaction workload or CPU utilization status. Fan et al. [23] proposed a joint task offloading and resource allocation scheme for MEC. Aghapour et al. [36] proposed a deep reinforcement learning-based technique that divides the task offloading and resource allocation problem into two sub-problems, to reduce the delay and power consumption related to work offloading. Xiao et al. [37] proposed an algorithm to reduce users' task-offloading overhead, by optimizing the computational offloading choice, the wireless resource allocation policy and the computational resource allocation strategy.

Some scholars conducted research on resource allocation for blockchain. Xia et al. [38] presented a three-phase auction method to allocate resources for mobile blockchain, which introduces a group buying mechanism to encourage edge servers to join the blockchain network. Kang et al. [13] proposed a reputation-based data-sharing scheme that uses contract technology and federation chains to accomplish secure data storage and avoid inappropriate data sharing. Toulouse et al. [39] investigated the problem of balancing workload of account-based blockchains such as Ethereum and proposed the method to predict transaction processing times. Jiao et al. [40] built an auction-based market model to determine the computing offloading strategy for miners and the allocation of computing resources to edge servers. Alfakeeh et al. [41] proposed the algorithms for the resource allocation of mining nodes, which utilizes the backpack algorithm to allocate the caching space of the mining nodes to resolve the problem of how to efficiently allocate resources so that data can be placed equitably in these fog mining nodes while maintaining the prioritization and sensitivity of patient data. Chang et al. [42] 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. Fan et al. [43] constructed a computation offloading model composed of multiple miners, multiple ESPs, and a CSP for mobile blockchain and proposed a three-stage Stackelberg game for optimal pricing-based edge computing resource management. Zhang et al. [24] proposed a group-agent strategy with trust computing to ensure the reliability of edge devices during interactions and improve transmission efficiency. Wang et al. [25] proposed an incentive mechanism, in which the client chooses the amount of computational power allocated to each task via a two-stage Stackelberg game based on

the rewards. Zhou et al. [44] proposed a blockchain-enabled spectrum trading system to maximize user benefits and designed a spectrum trading matching method. A Dynamic Credit Aggregate Signature Byzantine Fault Tolerant (DCABFT) algorithm based on Boneh-Lynn-Shacham (BLS) aggregated signatures was proposed, which aims to reduce the transaction latency and increase the user quality of service while addressing the latency and energy consumption of blockchain consensus algorithms. Zhang et al. [26] proposed two auction mechanisms for the blockchain network to maximize societal welfare and offer an efficient resource allocation method for edge computing service providers. Baranwal et al. [45] proposed a distributed auction-based resource allocation system in the context of edge computing-enabled industrial IoT. The technique eliminates the need of a trustworthy third party (the auctioneer) using consortium blockchain and smart contracts. Gao et al. [8] proposed a hierarchical resource scheduling scheme for the Vehicular Internetworking System (VINS) to allocate the computational resources, in order to increase the flexibility and efficiency of the system's resource utilization and to boost the performance of the blockchain-based VINS.

Both MEC and blockchain tasks require computational resources for execution in blockchain-based MEC. Due to the limited computational capability of each edge server, an imbalance in the allocation of computational resources to blockchain and MEC will result in a large system processing delay. However, most of the current research tackles the problem of resource allocation for either blockchain or MEC, and little research has been conducted on the optimization of resource allocation for both blockchain and MEC simultaneously. Feng et al. [46] proposed a joint optimization framework for blockchain-enabled MEC by jointly optimizing user offloading decisions and computational resource allocation to achieve the trade-off between the energy consumption of the MEC system and the latency of the blockchain system. In the framework, some of the users' tasks cannot be offloaded to the edge servers when they are making the choices of what tasks to be processed. However, this kind of offloading decisions are not reasonable, when the users have to but are unable to process some computation intensive tasks, e.g., real-time video stream processing, etc. As for the energy consumption, it is often difficult for the edge to obtain the factors of power, usage time, etc., of the large amount of heterogeneous users in the applications, which makes it hard to optimize the system energy consumption [47]. Furthermore, the energy consumption of users has no impact on the edge's utility. Therefore, in this paper, we study the problem of how to allocate the computing resources of edge servers for the MEC and the blockchain tasks with the objective of minimizing the total system processing delay.

3 Problem Definition

3.1 System Model of Blockchain-Based MEC

There are multiple users (IoT devices, sensors, wearables, etc.) and edge servers in the blockchain-based MEC system. Each edge server with constrained computing resources serves an area, and the users in the area can offload the tasks to the edge server. At the same time, the edge servers comprise the blockchain. After running the users' tasks, each edge server uploads the information related to the task execution, e.g., user, time, task workload, result, etc., to the blockchain. Each edge server, as a blockchain node, treats the task related information as a transaction and generates blocks, each containing several transactions. Each transaction contains the encrypted file and the hash of the data and the result related to the task execution. The sources of the transactions in a block are all the users served by the edge server. That is, the transactions in a block may come from several users. The blocks are broadcast to all blockchain nodes which verify the blocks. The block generation and verification are referred to as blockchain tasks. Accordingly, as shown in Fig. 1, each edge server runs two kinds of

tasks: those offloaded by users and those in blockchain. Table 1 lists the symbols and notations used in this paper.

Table 1: Table of symbols and notations

Notation	Definition
$\mathcal{N}$	User set
$\mathcal{M}$	Edge server set
D_n	Size of the computing task of user (mobile device) n
C_m	CPU cycles required for edge server m to process 1 bit data in a user's computing task
$x_{n,m}$	Indicate whether user n is in the service area of edge server m; 1, yes; 0, otherwise
$r_{n,m}$	Transmission rate from user n to edge server m
U_m	Total number of users served by edge server m
$P_{n,m}$	Transmission power from user n to edge server m
$g_{n,m}$	Channel gain between user n and edge server m
θ^2	Noise power consumption
S_b	Size of a block on the blockchain
I_n	Size of the transaction related to the task from user n, including the hash of the task related data and encrypted data
$\gamma_{n,m}$	Noise ratio from user n to edge server m plus channel interference
$J^m_{\max}$	Maximum processing capacity of edge server m
$T_{n,m}$	Latency of executing the tasks offloaded by user n on edge server m
$T^t_{n,m}$	Delay of transmitting the task related data from user n to edge server m
T^l	Latency of generating a new block on blockchain
T_k	Latency of task execution in blockchain for block k
B	Bandwidth allocated by edge servers to the served users
K	Total number of blocks generated by blockchain
E	Transfer rate of the link in the blockchain network
R_n	Size of the execution result of the task from user n
Q_n	Size of the encrypted files related to the task from user n
T^v	Latency of validating a new block in blockchain
$f_{b,m}$	Frequency of CPU cycles allocated to edge server m for processing block generation tasks
$f_{n,m}$	Frequency of CPU cycles allocated to edge server m for processing the tasks of user n

3.1.1 Latency of Task Execution in MEC

Denote the sets of users and edge servers as $\mathcal{N} = \{1, 2, \cdots, n, \cdots, N\}$ and $\mathcal{M} = \{1, 2, \cdots, m, \cdots, M\}$, respectively. The users offload their tasks to the edge servers using wireless communication. The system employs a shared frequency multiplexing method, wherein all edge servers can utilize identical wireless resources. The users adopt orthogonal spectrums for data transmission, with a collective bandwidth of B Hz. $x_{n,m}$ is a binary variable indicating if user n is in the service area of edge server m. The execution delay of the task is expressed as:

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

where $T^t_{n,m}$ and $T^c_{n,m}$ represent the delay of transmitting the task related data from user n to edge server m and the latency of executing the task of user n on edge server m, respectively. $T^t_{n,m}$ and $T^c_{n,m}$ are calculated as follows:

$$T^t_{n,m} = \frac{D_n}{r_{n,m}} \tag{2}$$

$$T^c_{n,m} = \frac{D_n C_m}{f_{n,m}} \tag{3}$$

where D_n is the task size of user n, C_m specifies the number of CPU cycles that edge server m requires to process 1 bit input data, and $f_{n,m}$ represents the frequency of CPU cycles allocated to edge server m for processing the task of user n. Note that the variations in the bandwidth and power of the channel are small, when the user is within the coverage area of an edge server and has low mobility in the applications. In this case, the transmission rate is fixed [48]. The rate of data transmission from user n to edge server m can be calculated as:

$$r_{n,m} = \frac{B}{U_m} \log_2 \left(1 + \gamma_{n,m}\right) \tag{4}$$

where U_m represents the number of users served by edge server m. $\gamma_{n,m}$ is the channel interference plus noise ratio (CINR) between user n and edge server m:

$$\gamma_{n,m} = \frac{P_{n,m} g_{n,m}}{\sum_{i,j \neq n,m} P_{\max} g_{i,j} + \theta^2} \tag{5}$$

where P_{max} indicates the maximum transmission power of each user, and θ^2 denotes the noise power. $P_{n,m}$ and $g_{n,m}$ respectively represent the transmission power and channel gain from user n to edge server m.

3.1.2 Latency of Blockchain Task Execution

The total latency T_k of task execution in blockchain for block k comes from three sources: block generation delay, block transmission delay and block verification delay:

$$T_k = T^I + T^t + T^v \tag{6}$$

where T^I represents the time required for the blockchain to generate a new block, and T^t denotes the transmission delay of the block during consensus. T^v denotes the block verification delay, which is a constant. Therefore, T^v has no impact on the computing resource allocation of edge servers. The time required for the blockchain to generate a new block is calculated as follows:

$$T^I = \frac{1}{K} \sum_{m \in \mathcal{M}} \sum_{n \in \mathcal{N}} \frac{x_{n,m} I_n C_m}{f_{b,m}} \tag{7}$$

where $f_{b,m}$ represents the CPU cycle frequency allocated by edge server m to process the block generation tasks. K is the total number of blocks generated by the blockchain, which is determined by the consensus round time (block intervals) [49]. I_n denotes the size of the transaction related to the task from user n, which is calculated as:

$$I_n = Hash\left(D_n + R_n\right) + Q_n \tag{8}$$

where $Hash(\cdot)$ is a hash function that computes the hash value of the task related information. R_n and Q_n denote the sizes of the execution result of the task from user n and the encrypted files related to the task from user n, respectively.

The block transmission delay T^t is calculated as:

$$T^t = \frac{S_b}{E} \tag{9}$$

where S_b represents the block size, i.e., the number of bytes in the block, and E denotes the data transmission rate between the edge servers.

3.2 Problem Model

Each edge server executes the tasks in both MEC and blockchain. In this paper, we study how to allocate the computing resources of each edge server to MEC and blockchain, with the aim to minimize the total system processing latency. That is, our objective is to

$$\min_{f_{n,m},f_{b,m}} \left(\sum_{m\in\mathcal{M}} \sum_{n\in\mathcal{N}} T_{n,m} + \sum_{k\in K} T_k \right)$$

$s.t.$

$$\begin{aligned} &\text{C1: } f_{b,m} + \sum_{n\in\mathcal{N}} x_{n,m} f_{n,m} \le J^m_{\max} \quad \forall m \in \mathcal{M} \\ &\text{C2: } f_{n,m} > 0, f_{b,m} > 0 \qquad \forall n \in \mathcal{N}, m \in \mathcal{M} \end{aligned} \tag{10}$$

Constraint C1 stipulates that the computational resources assigned by each edge server must be within the processing capacity of the edge server. Constraint C2 states that the edge server should allocate the computing resources to both MEC and blockchain.

4 Algorithm

The variables in the computing resource allocation problem in this paper are $f_{n,m}$ and $f_{b,m}$, and there exists capacity constraint C1. The fractional planning problem, a well-known NP-hard problem, can be reduced to our problem. Therefore, our problem is also NP-hard.

In this section, we propose a computing resource Allocation algorithm for Blockchain-based MEC (ABM), which consists of four steps as shown in Fig. 2. First, we prove that the problem is convex. Second, we prove that the problem satisfies the Slater's condition, and hence the problem and its dual satisfy strong duality. Third, we use the KKT conditions and the partial derivatives of the Lagrangian function to obtain the solution with the dual variables. Finally, we decide the values of the dual variables via the subgradient projection method, which obtains the solution to the problem.

We denote $f_1 = f_{n,m}$ and $f_2 = f_{b,m}$. The total system processing delay can be expressed as:

$$O = \sum_{m\in\mathcal{M}} \sum_{n\in\mathcal{N}} x_{n,m} \left(\frac{D_n}{r_{n,m}} + \frac{D_n C_m}{f_1} \right) + \sum_{k\in K} \left(\frac{1}{K} \sum_{m\in\mathcal{M}} \sum_{n\in\mathcal{N}} \frac{x_{n,m} I_n C_m}{f_2} \right) + \frac{S_b}{E} + T^v \tag{11}$$

where $\frac{S_b}{E}$ and T^v are constant. The original problem can be rewritten as:

$$\min_{f_1,f_2} O$$

$$s.t.$$

$$\text{C1}: f_2 + \sum_{n\in\mathcal{N}} x_{n,m} f_1 \le J_{\max}^m \quad \forall m \in \mathcal{M}$$

$$\text{C2}: f_1 > 0, f_2 > 0 \qquad \forall n \in \mathcal{N}, m \in \mathcal{M} \tag{12}$$

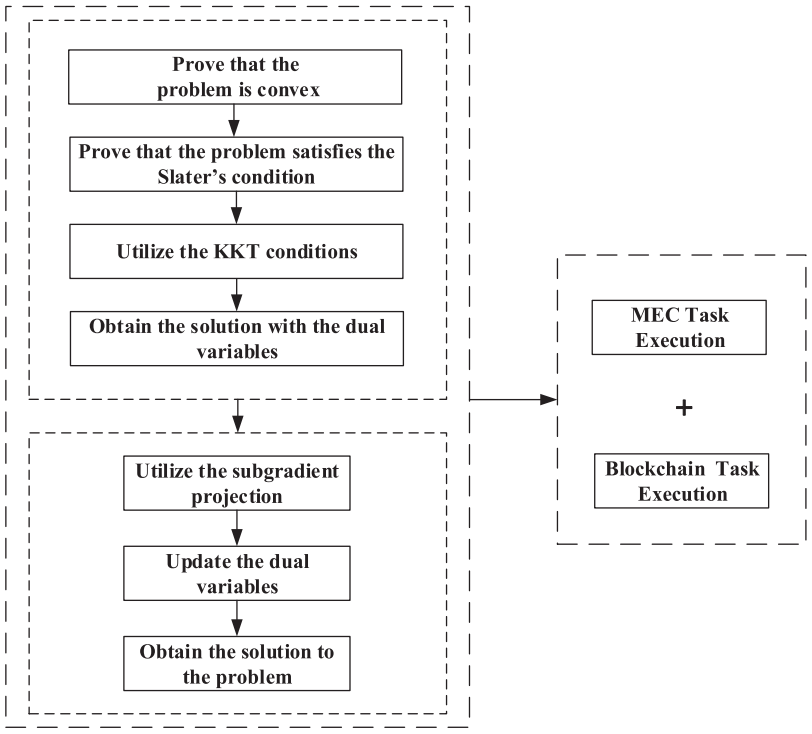

Figure 2: Diagram of ABM

The following equations can be obtained by calculating the first-order derivative of objective function O:

$$\frac{\partial O}{\partial f_1} = \sum_{m\in\mathcal{M}} \sum_{n\in\mathcal{N}} x_{n,m} \frac{D_n C_m}{\left(-f_1^2\right)} \tag{13}$$

$$\frac{\partial O}{\partial f_2} = \sum_{k\in K} \frac{1}{K} \sum_{m\in\mathcal{M}} \sum_{n\in\mathcal{N}} \frac{x_{n,m} I_n C_m}{\left(-f_2^2\right)} \tag{14}$$

The second-order derivative of objective function O can be calculated as follows:

$$\frac{\partial^2 O}{\partial f_1^2} = \sum_{m\in\mathcal{M}} \sum_{n\in\mathcal{N}} x_{n,m} \frac{2 D_n C_m}{f_1^3} \tag{15}$$

$$\frac{\partial^2 O}{\partial f_2^2} = \sum_{k\in K} \frac{1}{K} \sum_{m\in \mathscr{M}} \sum_{n\in \mathscr{N}} \frac{2x_{n,m} I_n C_m}{f_2^3} \tag{16}$$

The principal subequations of the Hessian matrix for each order have values greater than 0, and hence we can derive that the Hessian matrix is positive definite. Consequently, according to Theorem 4.1, objective function O is convex. Furthermore, the function in constraint C1 exhibits convexity, and hence the original problem is also convex.

Theorem 4.1. For a function $f(x)$, if the Hessian matrix of $f(x)$ is positive definite, then $f(x)$ is a convex function.

Proof. The Hessian matrix $H(x)$ of function $f(x)$ is represented as follows:

$$H(x) = \nabla^2 f(x) = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \cdots & \dfrac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \tag{17}$$

where $\nabla^2 f(x)$ represents the second-order partial derivative of function $f(x)$. Given that $H(x)$ is positive definite, the primary subequations of every order in the upper left corner of $H(x)$ are greater than 0. The expressions for the first-order and second-order principal subequations are as follows:

$$\frac{\partial^2 f}{\partial x_1^2} > 0, \quad \begin{vmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} \end{vmatrix} > 0 \tag{18}$$

Consider any two points, denoted as x and y, belonging to the definition domain. Given that $H(x)$ is positive definite, we can derive the following equation according to Taylor's equation

$$f(y) = f(x) + \nabla f(x)^T (y - x) + \frac{1}{2}(y - x)^T H(\xi)(y - x) > f(x) + \nabla f(x)^T (y - x)$$

where $\nabla f(x)^T$ represents the transpose of the first-order derivative of function $f(x)$. The inequality above satisfies the definition of a convex function, and hence we can conclude that function $f(x)$ is convex.

Lemma 4.1. The resource allocation problem in this paper satisfies the Slater's condition.

Proof. The Slater's condition holds for the definition domain of x, if there exists an interior point (not a point on the boundary)x^* satisfying the inequality constraints in which the equal sign is taken. According to the analysis, there must be f_1 and f_2 in our solution, such that $f_2 + \sum_{n\in \mathscr{N}} x_{n,m} f_1 < J_{\max}^m$. Therefore, the resource allocation problem in this paper satisfies the Slater's condition.

According to Lemma 4.1, we can derive that the original problem and its dual problem meet strong duality, since they both satisfy the Slater's condition and are mutually adequate to satisfy strong duality. The optimal solution using the KKT conditions can be found, if the original problem is convex and fulfills strong duality. We find the optimal solution by calculating the partial derivative of the Lagrangian dual function via the KKT conditions, using dual variables. The original problem's Lagrangian function can be written as follows:

$$L(f_1, f_2, \mu_n) = O + \sum_{n \in \mathcal{N}} \mu_n \left(f_2 + \sum_{n \in \mathcal{N}} x_{n,m} f_1 - J_{\max}^m \right) \tag{19}$$

where μ_n is a Lagrange multiplier and satisfies $\mu_n \geq 0$. The Lagrange dual function is expressed as:

$$D(\mu) = \min_{f_1, f_2 \in C1, C2} L(f_1, f_2, \mu_n) \tag{20}$$

where $\mu = (\mu_1, \mu_2, \cdots, \mu_n) \geq 0$. We can relax the original problem as an unconstrained optimization problem:

$$\min_{f_1, f_2} \sum_{m \in \mathcal{M}} \sum_{n \in \mathcal{N}} x_{n,m} \left(\frac{D_n}{r_{n,m}} + \frac{D_n C_m}{f_{n,m}} \right) + \sum_{k \in K} \left(\frac{1}{K} \sum_{m \in \mathcal{M}} \sum_{n \in \mathcal{N}} \frac{x_{n,m} I_n C_m}{f_{b,m}} \right)$$
$$+ \frac{S_b}{E} + T^v + \sum_{n \in \mathcal{N}} \mu_n \left(f_2 + \sum_{n \in \mathcal{N}} x_{n,m} f_1 - J_{\max}^m \right) \tag{21}$$

Following the definition of KKT conditions, we can derive

$$\frac{\partial L(f_1, f_2, \mu_n)}{\partial f_1} = \sum_{m \in \mathcal{M}} \sum_{n \in \mathcal{N}} x_{n,m} \frac{D_n C_m}{\left(-f_1^2\right)} + + \sum_{n \in \mathcal{N}} \mu_n \sum_{n \in \mathcal{N}} x_{n,m} = 0 \tag{22}$$

$$\frac{\partial L(f_1, f_2, \mu_n)}{\partial f_2} = \sum_{k \in K} \frac{1}{K} \sum_{m \in \mathcal{M}} \sum_{n \in \mathcal{N}} \frac{x_{n,m} I_n C_m}{\left(-f_2^2\right)} + \sum_{n \in \mathcal{N}} \mu_n = 0 \tag{23}$$

We can further obtain

$$f_1^* = \left(\frac{\sum_{m \in \mathcal{M}} \sum_{n \in \mathcal{N}} x_{n,m} D_n C_m}{\sum_{n \in \mathcal{N}} \mu_n \sum_{n \in \mathcal{N}} x_{n,m}} \right)^{\frac{1}{2}} \tag{24}$$

$$f_2^* = \left(\frac{\sum_{k \in K} \frac{1}{K} \sum_{m \in \mathcal{M}} \sum_{n \in \mathcal{N}} x_{n,m} I_n C_m}{\sum_{n \in \mathcal{N}} \mu_n} \right)^{\frac{1}{2}} \tag{25}$$

where f_2^* and f_1^* denote the optimal values of the CPU cycle frequencies allocated by the edge server to process MEC tasks and blockchain tasks, respectively. Therefore, we need to determine the dual variables μ_n, so as to decide the values of f_1^* and f_2^*. The dual problem of the original problem is as follows:

$$\max_{\mu} D(\mu)$$
$$\text{s.t. } \mu \geq 0 \tag{26}$$

According to Theorem 4.2, we can obtain the dual variables via the subgradient projection method, since the objective function and constraints in the original problem are convex.

Theorem 4.2. $\Delta\mu_n(t) = J_{\max}^m - (f_2 + \sum_{n \in \mathcal{N}} x_{n,m} f_1)$ and $\mu_n(t+1) = \mu_n(t) - i(t) \cdot \Delta\mu_n(t)$, where t is the iteration index, $i(t) = \dfrac{\alpha}{t}$ is the step size, and $\alpha (> 0)$ is a pre-determined constant.

Proof. From $D(\mu)$, we have the following inequality:

$$D(\mu') \geq \sum_{m\in\mathcal{M}}\sum_{n\in\mathcal{N}} x_{n,m}\left(\frac{D_n}{r_{n,m}} + \frac{D_n C_m}{f_1^*}\right) + \sum_{k\in K}\left(\frac{1}{K}\sum_{m\in\mathcal{M}}\sum_{n\in\mathcal{N}}\frac{x_{n,m} I_n C_m}{f_2^*}\right) + \frac{S_b}{E} + T^v + \sum_{n\in\mathcal{N}}\mu_n\left(f_2^* + \sum_{n\in\mathcal{N}} x_{n,m} f_1^* - J_{\max}^m\right) \tag{27}$$

After substitution, we have the following inequality:

$$D(\mu') \geq D(\mu) + \sum_{n\in\mathcal{N}}(\mu_n' - \mu_n)\left(f_2 + \sum_{n\in\mathcal{N}} x_{n,m} f_1 - J_{\max}^m\right) \tag{28}$$

Subgradient means that points x and y belonging to the definition domain of the function satisfy:

$$f(y) \geq f(x) + g^T(y - x) \tag{29}$$

where g is the subgradient of f. For $D(\mu)$, its subgradient is

$$f_2 + \sum_{n\in N} x_{n,m} f_1 - J_{\max}^m \tag{30}$$

Therefore, we can use the subgradient projection method to decide the values of the dual variables. That is

$$\Delta\mu_n(t) = J_{\max}^m - \left(f_2 + \sum_{n\in\mathcal{N}} x_{n,m} f_1\right) \tag{31}$$

and

$$\mu_n(t+1) = \mu_n(t) - i(t)\cdot\Delta\mu_n(t) \tag{32}$$

where t is the iteration index and $i(t)$ is the step size.

Algorithm 1: Algorithm of Computing resource Allocation for Blockchain-based MEC (ABM)

Input: Binary value $x_{n,m}$.
Output: Frequencies of CPU cycles allocated to the MEC and blockchain tasks, i.e., ${f_{n,m}}^*$ and ${f_{b,m}}^*$.

```
Initialize μ, t_max, δ
while: t ≤ t_max do
    Determine the CPU cycle frequency, f_1^*, allocated by the edge server for processing the users' tasks via Eq. (24);
    Calculate the edge server's CPU cycle frequency, f_2^*, allocated to the blockchain tasks via Eq. (25);
    Obtain Δμ_n(t) and μ_n(t+1) according to Theorem 4.2;
    if: ‖μ(t+1) − μ(t)‖_2 < δ then
        f_{n,m}^* ← f_1^*, f_{b,m}^* ← f_2^*;
        break;
    else
        t ← t + 1;
    end
end
```

The users in the service area of edge server m are the inputs of Algorithm 1. The outputs are the CPU cycle frequencies $f_{n,m}^*$ and $f_{b,m}^*$ that are allocated by the edge server to process the MEC and blockchain tasks, respectively. Step 1 initializes μ, t_{max}, δ. Steps 2−4 use Eqs. (24) and (25) to determine the CPU cycle frequency f_1^* assigned to the MEC tasks and the CPU cycle frequency f_2^* allocated to the blockchain tasks, respectively. Step 5 derives $\Delta\mu_n(t)$ according to Theorem 4.2 and then uses the subgradient projection method to calculate binary variable $\mu_n(t+1)$ based on step size $i(t)$ and binary variable $\mu_n(t)$. Steps 6−11 iteratively update the dual variables until the two-paradigm of the difference between $\mu_n(t+1)$ and $\mu_n(t)$ is less than the pre-defined threshold δ. In each iteration, the algorithm updates $f_{n,m}^*$ and $f_{b,m}^*$ as f_1^* and f_2^*, respectively.

5 Simulation

5.1 Simulation Setup

In the simulations, there are 60 users and 6 edge servers in the blockchain-based MEC system. The parameters of the system are listed in Table 2. The performance of ABM is evaluated by utilizing three benchmark algorithms, namely IADA [46], JCCR [33] and DSRC [24].

Table 2: Table of simulation parameters

Experimental parameters	Range of values
Bandwidth B	180 KHz [46]
Noise power θ^2	−174 dBm/Hz [46]
Computing task size D_n	200–1000 KB [33]
Link transmission rate E	15×10^6 bit/s [33]
Edge server's computing power $J_{\max}^m$	8 GHz [46]
Transaction size I_n	700 B [46]
Block size S_b	8 MB [24]
Number of CPU cycles, C_m, required to process 1 bit data	737.5 cycle/bit [46]

(1) IADA [46]: The algorithm first adopts binary search to obtain the resource allocation for processing MEC tasks, and then solves the stationing point to obtain the resource allocation for processing blockchain tasks.

(2) JCCR [33]: Each edge server allocates a fixed amount of computing resources to process MEC tasks.

(3) DSRC [24]: Each edge server allocates a fixed amount of computing resources to process blockchain tasks.

5.2 Simulation Results

Fig. 3 shows the total latency performance of ABM with different step sizes. ABM converges after 4 iterations, when step size $i(t)$ is $\frac{0.1}{t}$. ABM requires 7 iterations to converge, when step size $i(t)$ is $\frac{0.1}{2t}$. However, ABM cannot converge, when $i(t) = \frac{0.2}{t}$. The results demonstrate that a large step size

may result in oscillation or even failure to converge during the iteration process, while a too small step size may lead to slow convergence. Therefore, we set $i(t) = \frac{0.1}{t}$ in the following simulations.

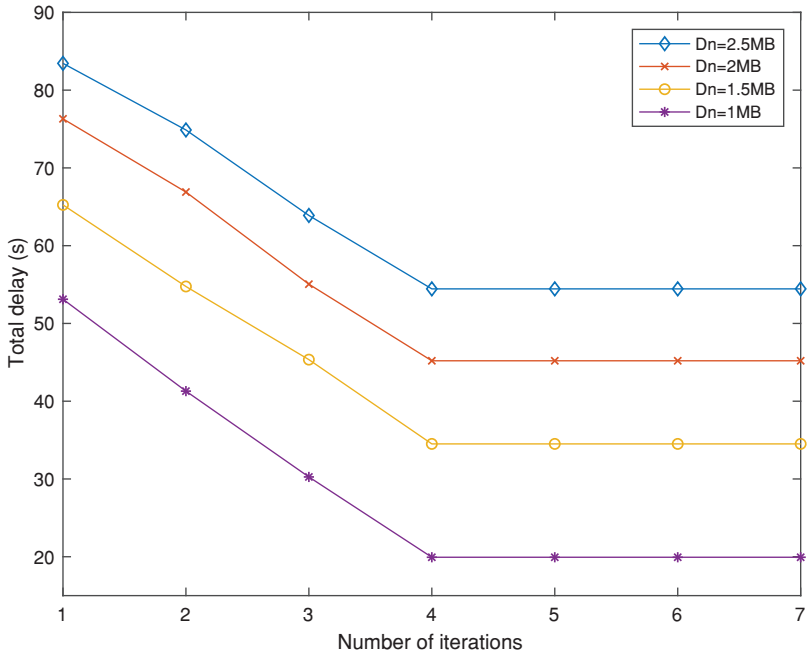

Figure 3: Impact of the initial step size in ABM on the total latency

The convergence performance of the proposed algorithm is illustrated in Fig. 4. It is noteworthy that the latency of the algorithm exhibits a decreasing trend as the number of iterations increases and the algorithm converges to a stable value. For example, the total latency decreases by 40.78% from the first to the fourth iteration, when the user task size is 2 MB. After the fourth iteration, the total latency stabilizes and does not change with the increase in the number of iterations. Furthermore, the latency of our algorithm is positively correlated to the computing task size. The total latency rises as the computing task size increases. For example, the total latency with the task size of 2.5 MB is 62.45% higher than that with the task size of 1 MB, after 4 iterations.

It can be observed from Fig. 5 that the total latency of IADA, JCCR, and DSRC is higher than that of ABM with a given number of users. The total latency of ABM is 4.85% and 16.45% lower than that of IADA, 15.21% and 29.11% lower than that of JCCR, and 17.02% and 15.21% lower than that of DSRC, when the numbers of users are 60 and 90, respectively. The total computing task size that the edge server needs to process increases with the increasing number of users. JCCR and DSRC use a fixed resource allocation strategy. In contrast, ABM utilizes the KKT conditions and the subgradient projection method to dynamically allocate the computing resources of the edge servers, such that the computing resources of the edge server can be dynamically allocated for MEC and blockchain tasks according to the number of users. The dynamic resource allocation is better able to adapt to the changes in the types and the number of tasks, which leads to improved resource utilization and system performance. IADA considers dynamic allocation. However, the resource allocation for MEC and blockchain tasks is performed individually, which results in the inferior performance to our algorithm.

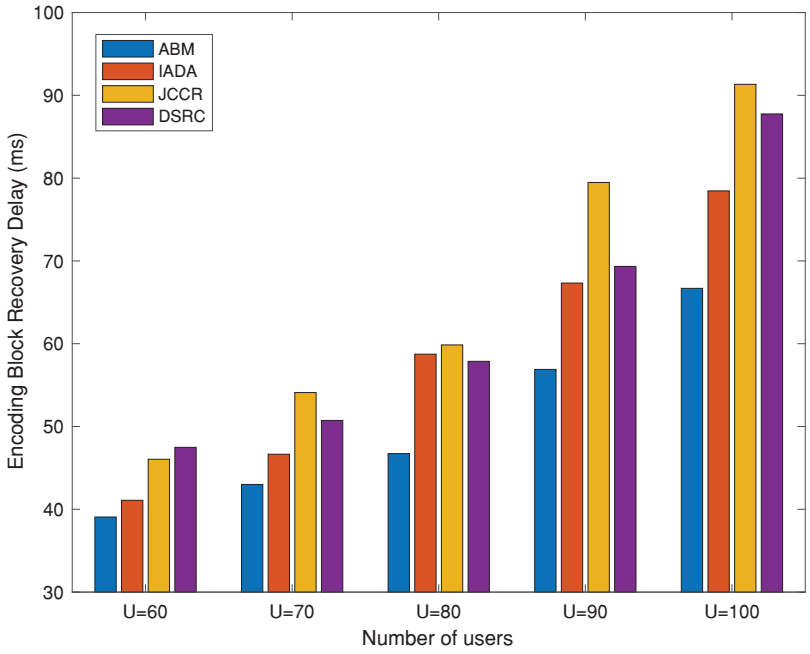

Figure 4: Convergence performance of ABM

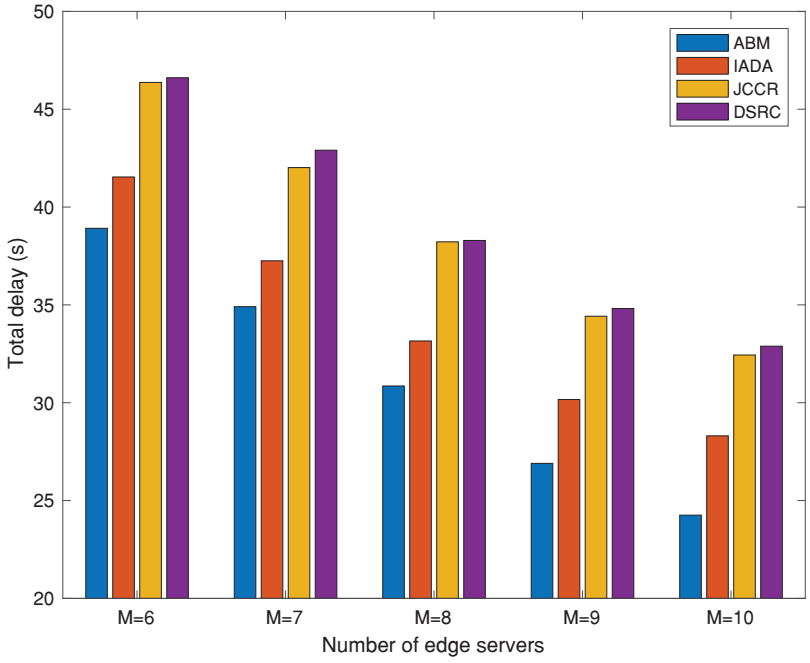

Figure 5: Impact of the number of users on the total delay

Fig. 6 illustrates the impact of the number of servers on the overall delay, when the number of edge servers increases from 6 to 10. The total latency of all algorithms reduces with more edge servers. As the number of edge servers increases, the number of users served by each edge server decreases, which

reduces the total size of the tasks to be processed by each edge server. Therefore, the overall latency is reduced. ABM achieves up to 8.15%, 17.39% and 18.64% lower system latency than IADA, JCCR and DSRC, respectively. ABM jointly optimizes the computing resource allocation for the MEC and the blockchain tasks, which leads to superior performance.

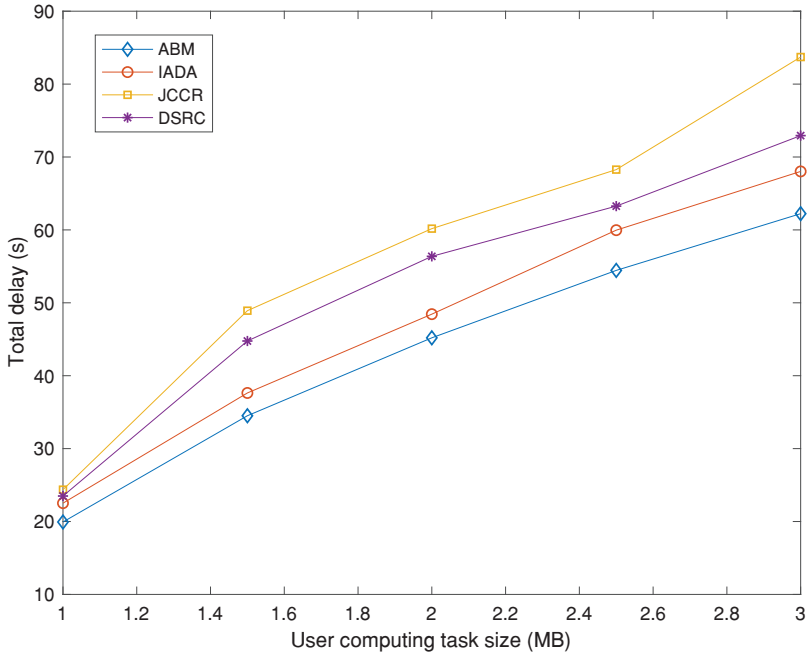

Figure 6: Impact of the number of edge servers on the total latency

Fig. 7 demonstrates the impact of the user's task size on the total delay. We can observe that the total delay of the four algorithms exhibits an increasing trend with the increase in the user's task size. The increase in the user's task size requires more computing resources and time for processing and more time for data and block transmission. Consequently, the overall delay experiences an increase. Fig. 7 illustrates that ABM consistently outperforms IADA, JCCR and DSRC in terms of the total latency. When the user's task sizes are 1, 2, and 3 MB, the total latency of ABM is respectively lower than that of IADA by 13.63%, 6.81% and 8.82%, less than that of JCCR by 20.52%, 25.33% and 25.30%, and lower than that of DSRC by 17.39%, 19.64% and 13.88%. ABM can dynamically allocate computing resources according to the change in task sizes by jointly optimizing the resource allocation for MEC and blockchain tasks. Therefore, ABM achieves a notable decrease in the overall system latency compared to IADA, JCCR and DSRC.

Fig. 8 shows the total latency of the four algorithms with various block sizes. It can be observed that the total delay increases with the increasing block size. The block with a large size contains more transactions, which means that the edge server needs more time to aggregate enough MEC task related information for block generation. A large block size also means that the blockchain needs more time to transmit and validate the block. ABM jointly optimize the resource allocation for MEC and blockchain tasks, such that the resource allocation can adapt to various block sizes to achieve a favorable total latency performance. Our algorithm outperforms IADA, JCCR and DSRC. Specifically, when the block sizes are 4, 6, and 8 MB, the total latency of ABM is 8.18%, 7.69% and

9.75% less than that of IADA, 25.65%, 28.57% and 28.84% lower than that of JCCR, and 26.08%, 25.64% and 26.88% less than that of DSRC, respectively.

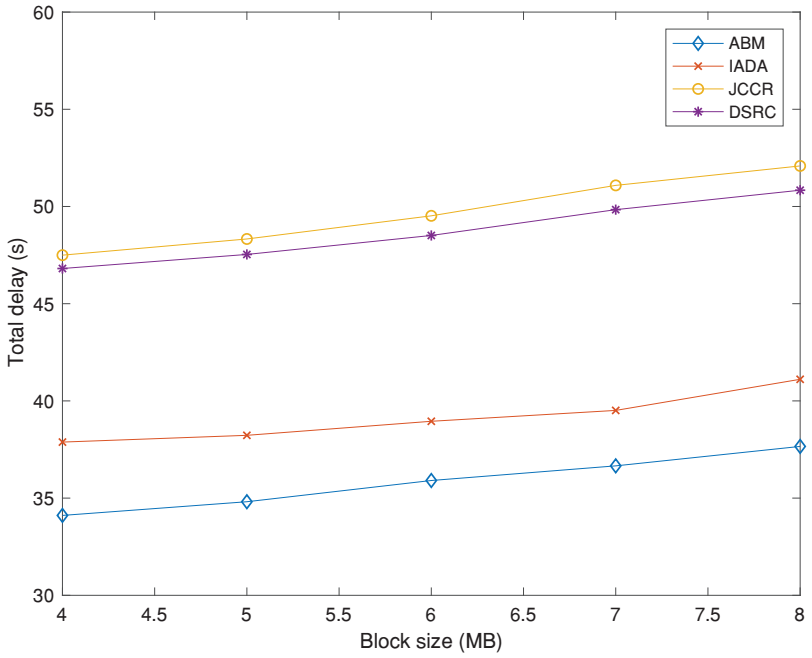

Figure 7: Impact of the user's task size on the total latency

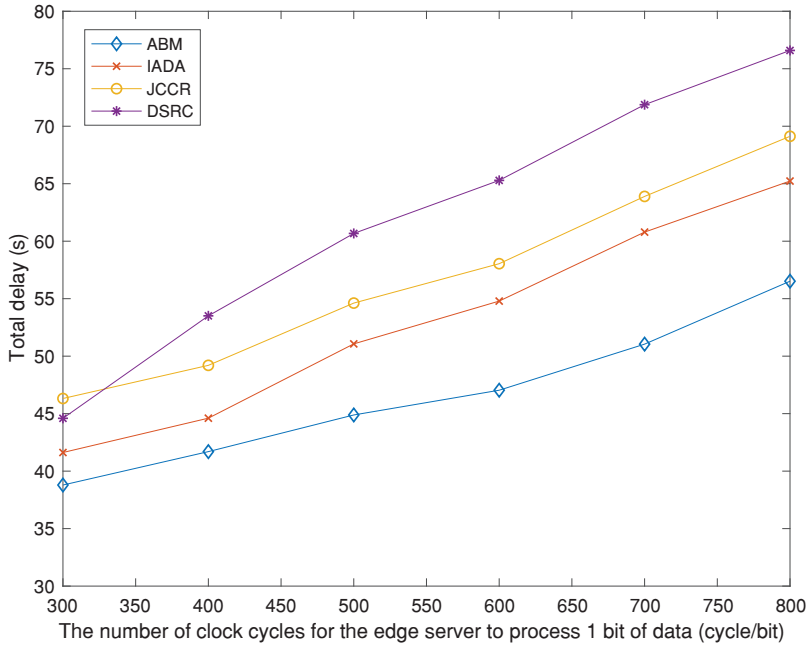

Figure 8: Impact of the block size on the total latency

Fig. 9 shows the total latency with different clock cycles required for the edge server to process 1 bit data. The total latency increases with the increase of the number of clock cycles C_m required for the edge server to process 1 bit data. A large C_m means that each edge server takes long time to process the MEC and blockchain tasks. ABM uses the KKT conditions and the subgradient projection method to optimize the allocation of edge servers' computing resources, which can flexibly adjust the resource allocation according to the task processing complexity. Fig. 9 indicates that ABM achieves lower total latency than IADA, JCCR and DSRC. When C_m is 400, 600 and 800, the total latency of ABM is 7.56%, 12.77% and 13.79% less than that of IADA, 17.39%, 18.96% and 18.84% lower than that of JCCR, and 22.64%, 26.15% and 26.31% less than that of DSRC, respectively.

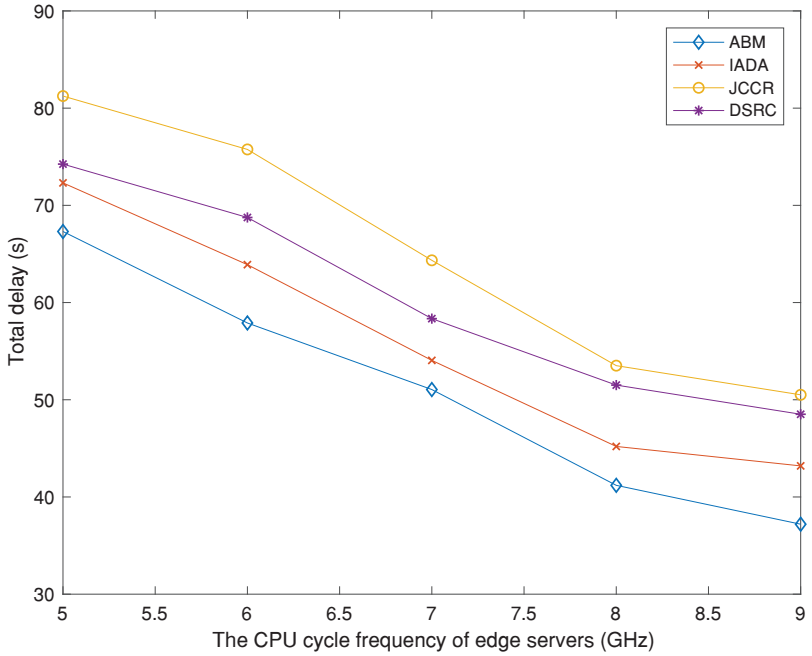

Figure 9: Impact of the number of clock cycles to process 1 bit data in the edge server on the total latency

Fig. 10 depicts the total system delay with different processing capacities of the edge server. The total latency declines with the increasing processing capacity of the edge server. A higher processing capacity leads to short processing time of MEC and blockchain tasks. Accordingly, the total delay of each algorithm reduces, as the processing capacity of the edge server increases. Fig. 10 illustrates that ABM achieves better total latency performance than IADA, JCCR and DSRC. Specifically, the total latency of ABM is up to 7.04%, 7.69% and 10.25% lower than that of IADA, 18.51%, 22.58% and 25.53% less than that of JCCR, and 9.58%, 14.28% and 22.21% lower than that of DSRC, respectively.

Fig. 11 illustrates the total latency *vs*. the transaction size. The total latency rises, as the transaction size increases. The edge server requires more time to generate the blocks with a large transaction size. ABM always incurs less total latency than IADA, JCCR and DSRC, due to the joint optimization of resource allocation for MEC and blockchain tasks. The total latency of ABM is less than that of IADA by 8.33%, 9.67% and 8.62%, and lower than that of JCCR by 12.38%, 10.34% and 14.52%, and less than that of DSRC by 14.56%, 13.46% and 15.62%, when the task sizes are 400, 700 and 1000 B, respectively.

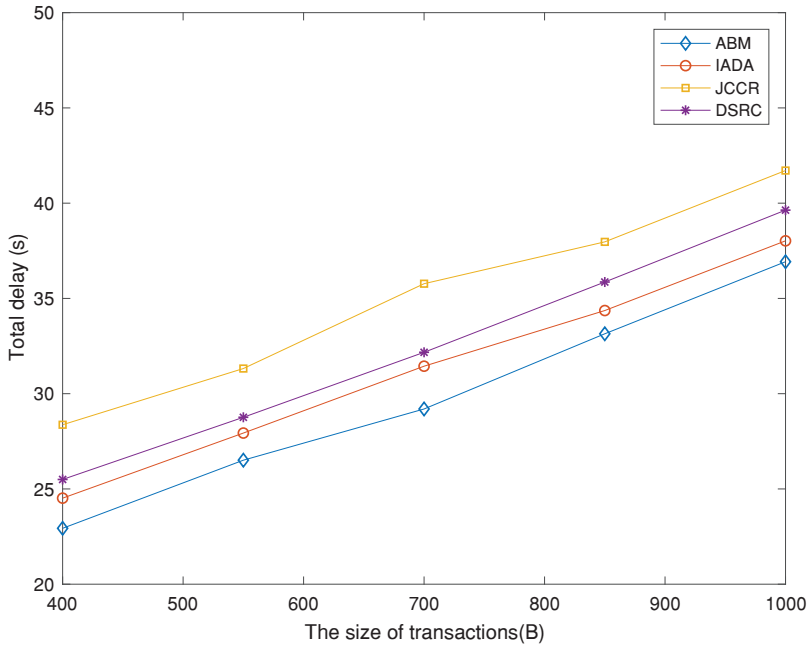

Figure 10: Impact of the processing capacity of the edge server on the total latency

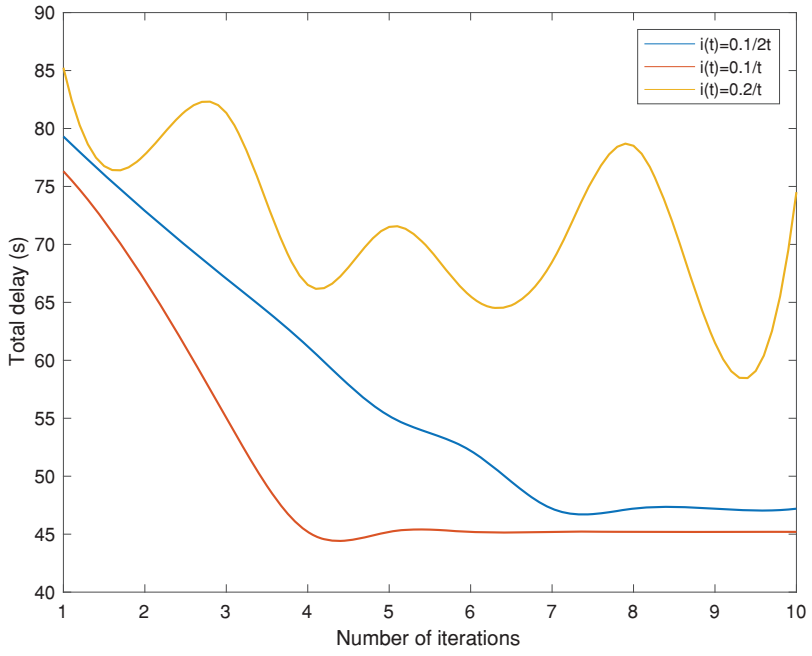

Figure 11: Impact of the transaction size on the total latency

6 Conclusions and Future Work

In this paper, we studied how to allocate the computing resources of edge servers to the MEC and blockchain tasks with the aim to minimize the total processing delay of blockchain-based MEC. For

the problem, we proposed a computing resource Allocation algorithm for the Blockchain-based MEC system (ABM), which consists of four steps. First, we prove that the problem is convex. Second, we prove that the problem satisfies the Slater's condition, and hence the problem and its dual satisfy strong duality. Third, we use the KKT conditions and the partial derivatives of the Lagrangian function to obtain the solution with the dual variables. Finally, we decide the values of the dual variables via the subgradient projection method, which obtains the solution to the problem. We conducted experiments through simulations. ABM demonstrates superior performance in terms of the total system latency under various parameters, including the number of users, user task sizes, block sizes, number of clock cycles required for edge servers to process 1 bit data, number of edge servers, processing capacity of the edge servers, block sizes and step sizes.

The users' tasks may have different priorities, which have a significant impact on the resource allocation. In our future work, we will work on how to allocate the computing resources of edge servers, to provide the QoS of different tasks.

Acknowledgement: The authors would like to express their gratitude to all who supported this research. We are thankful for the insightful comments from anonymous reviewers, which have greatly improved this manuscript. We also appreciate the Anhui Province Key Laboratory of Industry Safety and Emergency Technology for technical support.

Funding Statement: This work is partially supported by the Key Research and Development Project in Anhui Province of China (Grant No. 202304a05020059), the Fundamental Research Funds for the Central Universities of China (Grant No. PA2023GDSK0055) and the Project of Anhui Province Economic and Information Bureau (Grant No. JB20099).

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Yuqi Fan, Wanbo Zhang; algorithm design: Wanbo Zhang, Yuqi Fan; analysis and interpretation of results: Wanbo Zhang, Jun Zhang, Xu Ding, Jung Yoon Kim; draft manuscript preparation: Wanbo Zhang, Yuqi Fan. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: All data is synthesized using MATLAB and the generation method is described in detail in the simulation section.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Cau, E., Corici, M., Edmonds, A. (2016). Efficient exploitation of mobile edge computing for virtualized 5G in EPC architectures. *2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, USA.
2. Borgia, E., Bruno, R., Conti, M. (2016). Mobile edge clouds for information-centric IoT services. *IEEE Symposium on Computers and Communication*, Italy.
3. Gao, H., Wang, X., Wei, W. (2023). Com-DDPG: Task offloading based on multiagent reinforcement learning for information-communication-enhanced mobile edge computing in the Internet of Vehicles. *IEEE Transactions on Vehicular Technology, 73(1),* 348–361.
4. Orsini, G., Dirk, B., Winfried, L. (2015). Edge intelligence: The confluence of edge computing and artificial intelligence. *2015 8th IFIP Wireless and Mobile Networking Conference*, Canada.

5. Khan, W. Z., Ahmed, E., Ejaz, A. (2019). Edge computing: A survey. *Future Generation Computer Systems, 97(39),* 219–235.
6. Gao, H., Fang, D., Xiao, J. (2023). CAMRL: A joint method of channel attention and multidimensional regression loss for 3D object detection in automated vehicles. *IEEE Transactions on Intelligent Transportation Systems, 24(8),* 8831–8845.
7. Jararweh, Y., Doulat, A., AlQudah, O. (2016). The future of mobile cloud computing: Integrating cloudlets and mobile edge computing. *2016 23rd International Conference on Telecommunications*, Greece.
8. Gao, H., Huang, J., Tao, Y. (2022). Enabling low-latency applications in LTE-A based mixed fog/cloud computing systems. *IEEE Transactions on Computational Social Systems, 9(6),* 1725–1735.
9. Tran, T. X., Hajisami, Pandey, P. (2017). Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges. *IEEE Communications Magazine, 55(4),* 54–61.
10. Liu, J., Mao, Y., Zhang, J. (2016). Delay-optimal computation task scheduling for mobile-edge computing systems. *IEEE International Symposium on Information Theory*, USA.
11. Mao, Y., Zhang, J., Song, S. (2017). Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems. *IEEE Transactions on Wireless Communications, 16(9),* 5994–6009.
12. You, C., Huang, K., Chae, H. (2016). Energy-efficient resource allocation for mobile-edge computation offloading. *IEEE Internet of Things Journal, 16(3),* 1397–1411.
13. Kang, J., Yu, R., Huang, X. (2018). Blockchain for secure and efficient data sharing in vehicular edge computing and networks. *IEEE Transactions on Wireless Communications, 6(3),* 4660–4670.
14. Xiong, Z., Zhang, Y., Niyato, D. (2018). When mobile blockchain meets edge computing. *IEEE Communications Magazine, 56(8),* 33–39.
15. Dang, T., Duong, T. A. (2021). An effective and elastic blockchain-based provenance preserving solution for the open data. *International Journal of Web Information Systems, 17(5),* 480–515.
16. Miller, D. (2018). Blockchain and the Internet of Things in the industrial sector. *IT Professional, 20(3),* 15–18.
17. Liang, X., Zhao, J., Shetty, S. (2017). Towards data assurance and resilience in IoT using blockchain. *IEEE Military Communications Conference*, USA.
18. Aitzhan, N., Svetinovic, D. (2018). Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams. *IEEE Transactions on Dependable and Secure Computing, 17(5),* 840–852.
19. Li, Z., Kang, J., Zhang, Y. (2017). Consortium blockchain for secure energy trading in industrial Internet of Things. *IEEE Transactions on Industrial Informatics, 14(8),* 3690–3700.
20. Lang, P., Tian, D. X., Duan, X. T., Zhou, J. S., Sheng, Z. G. et al. (2022). Cooperative computation offloading in blockchain based vehicular edge computing networks. *IEEE Transactions on Intelligent Vehicles, 7(3),* 783–798.
21. Ren, J., Yu, G., He, Y. (2018). Latency optimization for resource allocation in mobile edge computation offloading. *IEEE Transactions on Wireless Communications, 17(8),* 5506–5519.
22. Spallina, A., Araldo, A., Chahed, T. (2022). Energy-efficient resource allocation in multi-tenant edge computing using Markov decision processes. *Computer Networks, 140(24),* 1–5.
23. Fan, W., Su, Y., Huang, W. (2023). Joint task offloading and resource allocation for vehicular edge computing based on V2I and V2V modes. *IEEE Transactions on Intelligent Transportation Systems, 24(4),* 4277–4292.
24. Zhang, L., Zou, Y., Su, Y. (2021). Resource allocation and trust computing for blockchain-enabled edge computing system. *Computers and Security, 105(41),* 102249.
25. Wang, Z., Hu, Q., Xiong, Z. (2023). Incentive mechanism design for joint resource allocation in blockchain-based federated learning. *IEEE Transactions on Parallel and Distributed System, 34(5),* 1536–1547.

26. Zhang, J., Lou, W., Li, W. (2022). Truthful auction mechanisms for resource allocation in the Internet of Vehicles with public blockchain networks. *Future Generation Computer Systems, 132(39),* 625–629.
27. Nosrati, M., Fazlali, M. (2018). Community-based replica management in distributed systems. *International Journal of Web Information Systems, 14(1),* 41–61.
28. Feng, J., Pei, Q., Yu, F. (2019). Computation offloading and resource allocation for wireless powered mobile edge computing with latency constraint. *IEEE Wireless Communications Letters, 8(5),* 1320–1323.
29. Hong, G., Wen, Q., Wu, P. (2019). An optimal resource allocation mechanism in vehicular MEC systems. *International Conference on Networking and Network Applications*, South Korea.
30. Liu, J. H., Wu, Z. B., Liu, J. J., Zou, Y. (2022). Cost research of internet of things service architecture for random mobile users based on edge computing. *International Journal of Web Information Systems, 18(4),* 217–235.
31. Zhang, Y., Zhang, M., Li, B. (2021). Computing resource allocation scheme of IOV using deep reinforcement learning in edge computing environment. *EURASIP Journal on Advances in Signal Processing, 2021,* 33.
32. Guo, Y., Liu, F., Xiao, N., Li, Z. G., Cai, Z. P. et al. (2022). PARA: Performability-aware resource allocation on the edges for cloud native services. *International Journal of Intelligent Systems, 37(11),* 8523–8547.
33. Zhang, L., Zhang, H. L., Guo, C., Xu, H. T., Song, L. Y. et al. (2020). Satellite aerial integrated computing in disasters: User association and offloading decision. *IEEE International Conference on Communications*, Ireland.
34. Cui, L. Z., Chen, Z. T., Yang, S., Ming, Z. X., Li, Q. et al. (2020). A blockchain-based containerized edge computing platform for the Internet of Vehicles. *IEEE Internet of Things Journal, 8(4),* 2395–2408.
35. Gao, L., Wu, C., Du, Z., Yoshinaga, T., Zhong, L. et al. (2022). Toward efficient blockchain for the Internet of Vehicles with hierarchical blockchain resource scheduling. *Electronics, 11(5),* 832.
36. Aghapour, Z., Sharifian, S., Taheri, H. (2023). Task offloading and resource allocation algorithm based on deep reinforcement learning for distributed AI execution tasks in IoT edge computing environments. *Computer Networks, 223(24),* 213–223.
37. Xiao, J., Gao, Q., Wang, H. (2023). Multi-round auction-based resource allocation for edge computing: Maximizing social welfare. *Future Generation Computer Systems, 223(39),* 365–375.
38. Xia, C., Chen, H., Chen, L. (2018). ETRA: Efficient three-stage resource allocation auction for mobile blockchain in edge computing. *IEEE International Conference on Parallel and Distributed Systems*, China.
39. Toulouse, M., Dai, H. K., Le, T. G. (2022). Distributed load-balancing for account-based sharded blockchains. *International Journal of Web Information Systems, 18(2),* 100–116.
40. Jiao, Y., Wang, P., Niyato, D., Suankaewmanee, K. (2019). Auction mechanisms in cloud/fog computing resource allocation for public blockchain networks. *IEEE Transactions on Parallel and Distributed Systems, 30(9),* 1975–1989.
41. Alfakeeh, A. S., Javed, M. A. (2022). Efficient resource allocation in blockchain assisted health care systems. *Applied Sciences, 13(17),* 9625.
42. Chang, Z., Guo, W. L., Guo, X. J., Zhou, Z. Y., Ristaniemi, T. (2020). Incentive mechanism for edge-computing-based blockchain. *IEEE Transactions on Industrial Informatics, 16(11),* 7105–7114.
43. Fan, Y. Q., Jin, Z. F., Shen, G. M., Hu, D. H., Shi, L. et al. (2021). Three-stage stackelberg game based edge computing resource management for mobile blockchain. *Peer-to-Peer Networking and Applications, 14,* 1431–1445.
44. Zhou, Y., Yu, L., Maharjan, S. (2023). An improved spectrum trading design based on dynamic credit aggregate signature blockchain. *IEEE Wireless Communications Letters, 12(4),* 625–629.
45. Baranwal, G., Kumar, D., Vidyarthi, D. (2022). BARA: A blockchain-aided auction-based resource allocation in edge computing enabled Industrial Internet of Things. *Future Generation Computer Systems, 135(39),* 333–347.

46. Feng, J., Yu, F., Pei, Q. (2020). Joint optimization of radio and computational resources allocation in blockchain-enabled mobile edge computing systems. *IEEE Transactions on Wireless Communications, 19(6),* 4321–4334.
47. Li, P., Wang, Z. X., Wang, N., Yang, W. H., Li, M. Z. et al. (2021). Stochastic robust optimal operation of community integrated energy system based on integrated demand response. *International Journal of Electrical Power and Energy Systems, 128(14),* 106735.
48. Yan, S., Hanly, S. I. B. (2021). Optimal transmit power and flying location for UAV covert wireless communications. *IEEE Journal on Selected Areas in Communications, 39(11),* 3321–3333.
49. Zuo, Y. P., Jin, S., Zhang, S. L. (2021). Computation offloading in untrusted MEC-aided mobile blockchain IoT systems. *IEEE Transactions on Wireless Communications, 20(12),* 8333–8347.