Energy-Efficient Blockchain-enabled User-Centric Mobile Edge Computing

Langtian Qin, Hancheng Lu, *Senior Member, IEEE*, Yuang Chen, Zhuojia Gu, Dan Zhao, and Feng Wu, *Fellow, IEEE*

Abstract—In the traditional mobile edge computing (MEC) system, the availability of MEC services is greatly limited for the edge users of the cell due to serious signal attenuation and inter-cell interference. User-centric MEC (UC-MEC) can be seen as a promising solution to address this issue, as well as provide sufficient resources for resource-consuming mobile applications. In UC-MEC, each user is served by a dedicated access point (AP) cluster enabled with MEC capability instead of a single MEC server, however, at the expense of noticeable energy consumption. To achieve reliable and efficient resource utilization with user-centric services, we propose an energy efficient blockchain-enabled UC-MEC where blockchain operations and resource optimization are jointly performed. Firstly, we design a resource-aware RAFT consensus mechanism followed by a smart contract to implement secure and reliable resource trading. Then, an optimization framework based on alternating direction method of multipliers (ADMM) is proposed to minimize the total energy consumed by wireless transmission, consensus and task computing, where APs clustering, computing resource allocation and bandwidth allocation are jointly considered. Simulation results show superiority of the proposed UC-MEC system over reference schemes, at most 33.96% reduction in the total delay and 63.8% reduction in the total energy consumption.

Index Terms—Mobile edge computing, user-centric networks, blockchain, task offloading, alternating direction method of multipliers.

1 Introduction

With the rapid development of mobile Internet, the growing variety of mobile applications put forward higher requirements for resource capability of smart devices [1]. By moving the services and functions originally located in the cloud to the user side, mobile edge computing (MEC) provides powerful computing, storage, networking and communication capabilities at the edge of wireless access networks [2]. However, in traditional cellular-based MEC networks that adopted by most of the existing works about MEC, each user will only be provided with transmission and computing services by a single BS integrated with MEC servers. Users at the edge of the cell will prone to suffer serious signal attenuation and inter-cell interference, which may significantly increase the transmission delay or even cause the offloading failure. Moreover, resources of a single cell may not be able to meet the requirements of users with resource-consumption mobile applications such as augmented reality (AR), virtual reality (VR), et al.

As an emerging technology of 5G and beyond, user-centric network can be seen as a reliable solution to overcome the above problems [3], [4]. In UCN, each user will be served by a dynamically divided AP set, which is called AP cluster [5]. Each AP cluster can be divided adaptively according to the location and network condition of users to provide seamless wireless transmission service. By integrating MEC servers in the AP, user-centric MEC (UC-MEC) breaks the concept of "cell" in traditional cellular-based MEC, and can further expand the computing and communication resources for task offloading in MEC. In UC-MEC, When a user requests MEC services, it will be collaboratively served by a dynamically divided set of APs, i.e., AP cluster. Each AP in the cluster processes part of the offloaded tasks of the user. Through UC-MEC, users can be provided with efficient and reliable wireless transmission and task processing services wherever they are.

In spite of the overarching merits, user-centric service mode will make the issues of security and privacy more prominent. Firstly, users need to transmit signals to multiple APs through wireless channels, which increases the risk of being attacked compared with traditional wireless networks due to the openness of wireless channel. Secondly, since the AP cluster will change dynamically with the user location and network condition, the access and authorization management of the AP cluster will become more important. If any AP in the AP cluster is hijacked, the user's privacy information may be easily disclosed during the offloading process. However, the traditional centralized security solutions has the hidden danger of single point of failure. When the network controller fails or is maliciously hijacked, the security and privacy of users in the network will be difficult to protect. Also, the centrality of the controller increases communication overhead and network latency [6].

Blockchain can be viewed as an promising technology to solve these problems in UC-MEC due to its immutability, decentralization, transparency, security, and privacy features [7], [8]. In blockchain-enabled UC-MEC system, each node maintains an identical distributed ledger, which records network resources trading information between users and APs. The ledger is totally transparent and tamper-free. After the offloading completed, the network node executes the consensus mechanism, and packages the transaction information into blocks to record on the blockchain,

- *L.Qin, H.Lu, Y.Chen, Z.Gu, D.Zhao and F.Wu are with the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei 230027, China.*
E-mail: qlt315@mail.ustc.edu.cn; hclu@ustc.edu.cn; {yuangchen21, guzj, zd2019@mail.ustc.edu.cn}; fengwu@ustc.edu.cn

which can prevent data loss and tampering caused by node failure. In addition, the advent of the blockchain-based smart contract [9], [10] can establish the access control and authorization mechanism of the AP cluster in UC-MEC conveniently and efficiently.

To exploit the benefits of blockchain-enabled UC-MEC, there still remain the following research gaps to be addressed. Firstly, in UC-MEC systems, multiple APs are dynamically formed into an AP cluster to realize a user-centric service. It means that all APs are potential to allocate transmission and MEC resources for users. In this case, a decentralized mechanism is required to ensure secure and reliable resource trading between APs and users. Secondly, blockchain-enabled UC-MEC can significantly improve the quality of service (QoS) and ensure the security and privacy of users, however, at the expense of surge in system energy consumption [4]. To minimize the total energy consumption of blockchain-enabled UC-MEC, it is necessary to jointly consider the energy consumption during task offloading and resource trading. These two issues are coupled with APs clustering decision and resource allocation in blockchain-enabled UC-MEC systems, which are more challenging than that in traditional blockchain-enabled MEC systems.

1.1 Related Work

Blockchain has been viewed as a promising technology to enhance security and reliability of MEC systems in a distributed manner [11]. There exist many studies on the combination of blockchain and MEC. Some studies considered the consensus process as a task offloading to the MEC server to reduce consensus delay [12]–[14]. The authors of [12] used consortium blockchain and smart contract to achieve trusted resource allocation in edge computing of vehicle network. An edge computing resource management framework based on optimal pricing is proposed in [13], where the mining process can be offloaded to the edge computing service provider. In [14], authors considered that mining tasks can be offloaded to nearby edge computing nodes and encrypted hashes of blocks can be cached in MEC servers. There also exist some studies on consideration of both computing task offloading and block mining in the consensus process [15], [16]. In [15], authors proposed a new collaborative task offloading and block mining algorithm for blockchain-based MEC systems. In [16], joint optimization was performed on offloading decisions, power allocation, block size, and block interval to maximize the computing speed of the MEC system and the transaction throughput of the blockchain system. With the rise of machine learning in MEC systems [17], blcokchain was used to ensure the privacy of distributed learning [11]. However, most of these studies considered resource allocation in task offloading and blockchain operations separately, which might lead to low resource utilization efficiency. Moreover, existing studies focused on blockchain operations in a server-centric manner. For UC-MEC where all APs with MEC capability are potential service providers for users, new consensus processes involving all APs should be designed.

Energy consumption has also been focused on in traditional MEC systems. In [18], the weighted sum of the mobile energy consumption under the constraint on computing latency is minimized for a multi-user MEC offloading system. The author of [19] studied task offloading in multi-user MEC systems with heterogeneous clouds, and optimized the energy consumption of multiple mobile devices by jointly allocating bandwidth and computing resources to mobile devices. In [20], the energy consumption of multiple mobile devices was further optimized by jointly consideration of offloading selection, radio resource allocation, and computing resource allocation. However, in these studies, the energy consumption for wireless transmission has not been well considered, which is nontrivial in UC-MEC systems supporting multiple wireless connections for users. Additionally, APs in UC-MEC are connected via wireless links to ensure the flexibility of deployment [21]. In this case, the energy consumption for wireless communication between APs cannot be ignored. In [22], the authors considered usersrisk-seeking or loss-aversion behavior in their offloading process. A non-cooperative game among the users is formulated and a distributed low-complexity algorithm is proposed to obtain the corresponding Pure Nash Equilibrium (PNE), i.e., optimal data offloading strategy. By adopting the satisfaction games and approximate computing, the authors of [23] introduced an energy efficient solution in MEC-enabled fully autonomous aerial systems (FAAS) to obtain the optimal partial offloading decisions under minimum QoS prerequisites. The authors in [24] modeled joint cost and energy-efficient task offloading in the MEC-enabled healthcare system by Stackelberg game. The optimal task offloading decision can be derived in a distributed manner by using alternating direction method of multipliers (ADMM) method. In [25], the authors modeled the economic interaction between the MEC server and users using the Nash bargaining theory to minimizes the energy consumption of the MEC server without compromising on the quality-of-experience (QoE) of the users. The authors of [26] proposed a software-defined networking (SDN) based framework for edge and cloud computing system. An evolutionary stackelberg differential game based dynamic pricing and computing resource allocation mechanism is proposed to optimize computing resource utilization and satisfy the time-varying computational tasks. In [27], the task offloading in MEC is modeled as a constrained multi-objective optimization problem (CMOP). The authors proposed an evolutionary algorithm to find the best trade-offs between energy consumption and task processing delay.

1.2 Contributions

To address aforementioned issues, we propose an energy-efficient blockchain-enabled UC-MEC system. Different from traditional MEC, in the proposed system, we provide users with user-centric computing services, and then perform joint optimization of APs clustering decision, computing resource and bandwidth allocation to achieve secure, reliable and efficient resource utilization. The main contributions of this paper are summarized as follows.

- To implement secure and reliable resource trading in UC-MEC, we design a resource-aware RAFT consensus mechanism based on blockchain, i.e., R-RAFT. In R-RAFT, we propose resource-aware election of *Leader* replacing random selection of *Leader* in RAFT. With R-RAFT, the efficiency of the consensus process

is significantly improved. Based on R-RAFT, we design a smart contract to ensure security and privacy of resource trading between APs and users.

- We analyze the energy consumption of the proposed blockchain-enabled UC-MEC. The energy consumption for wireless transmission and consensus process is jointly considered with that for task computing. Based on our analytical work, we formulate the energy consumption minimization problem by jointly optimizing APs clustering, computing resource allocation and bandwidth allocation.
- To solve the formulated problem, we propose a parallel optimization framework based on alternating direction method of multipliers (ADMM). Particularly, we firstly decompose the problem into some sub-problems. Then we convert the sub-problem into a D.C programming problem and propose a sub-gradients iterative algorithm to solve the converted problem in parallel by all APs.

The simulation results show that the proposed ADMM-based scheme outperforms reference schemes in terms of total delay and energy consumption. Compared with traditional MEC, the proposed blockchain-enabled UC-MEC system can reduce the total delay .

1.3 Organization

The rest of the paper is organized as follows. The blockchain-enabled UC-MEC system is modeled in Section II. In Section III, the energy consumption of the proposed UC-MEC system is analyzed and then the energy consumption minimization problem is formulated. To solve the formulated problem, joint APs clustering and resource allocation optimization based on ADMM is performed in Section IV. Simulation results are present in Section V. Finally, conclusion and future works are given in Section VI.

Notations: In this paper, we use $\boldsymbol{A}$ to represent a matrix. a_{ij} denotes the element in i-th row and j-th column of $\boldsymbol{A}$, $\boldsymbol{A}^{\dagger}$ and $\|\boldsymbol{A}\|_2$ represents the Pseudo inverse and ℓ_2 norm of $\boldsymbol{A}$, respectively. Specifically, we use $\boldsymbol{I}_N$ to denote a identity matrix with dimension N. We use $\boldsymbol{x}$ to denote a vector, and x_i denotes the i-th element of vector $\boldsymbol{x}$. $\boldsymbol{x}^T$ and $\boldsymbol{x}^H$ represent the transpose and Hermitian of vector $\boldsymbol{x}$, respectively. $\mathbb{R}^{M\times N}$ and $\mathbb{C}^{M\times N}$ represents the space of $M \times N$ real and complex number matrices. $\mathbb{E}[x]$ denotes the mathematical expectation of x. We use calligraphy uppercase letter such as $\mathcal{M}$ to represent a set.

2 System Model

In this section, we first introduce the UC-MEC framework, including the network and the task offloading model of UCMEC. Then we describe the blockchain based resource trading process in UC-MEC, including our proposed R-RAFT consensus mechanism and resource trading smart contract. Some key notations are shown in Table I.

2.1 User-Centric Mobile Edge Computing

2.1.1 Network Model

As shown in Fig. 1, we consider a blockchain-enabled UC-MEC system with M APs and N users. There are two types of wireless links in UC-MEC, i.e., the wireless data transmission links between users and APs, and the wireless backhaul links among APs. We assume that the set of APs is represented as $\mathcal{M} = \{1, 2, 3, \cdots, M\}$. Each AP is equipped with $X > N$ antennas and is integrated with a MEC server which contains certain bandwidth and computing capability. Considering the heterogeneity of APs, the bandwidth and the computing capability of each AP are different, which are indicated by B_m (in MHz) and C_m (in CPU-cycle frequency), respectively. We denote the set of users in the network as $\mathcal{N} = \{1, 2, \cdots, N\}$, and each user will be served by a specific AP cluster (i.e., a collection of several APs).

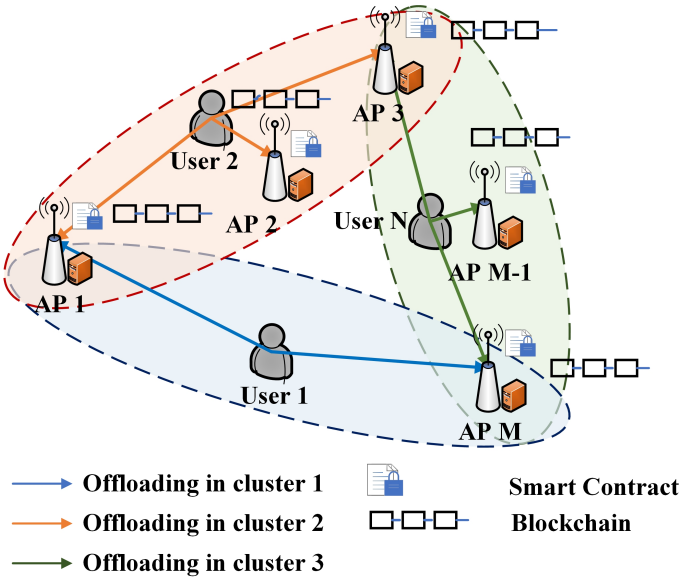


Fig. 1. The framework of blockchain-enabled UC-MEC

2.1.2 Task Offloading Model

In a certain time slot, each user generates a task that need to offload to the MEC-enabled APs. We use a triple $Task_n = \{L_n, \rho_n, D_n^t\}$ to express the task that user n ($n \in \mathcal{N}$) needs to offload, where L_n, ρ_n and D_n^t stand for the task data size (in bit), computing density (in CPU cycles per bit) and maximum tolerated delay (in second) [28]. When user n generates a task and sends the offloading request to the network, APs in the network will decide whether to join the cluster of user n or not. Similar to [14], [29], we use a continuous variable $a_{m,n} \in [0, 1]$ to indicate the cluster decision of AP m for user n. If $a_{m,n} = 0$, AP m will not join the AP cluster of user n; if $a_{m,n} \in (0, 1]$, AP m will join the AP cluster of user n, and responsible for the corresponding proportion of offloading task. We define Φ_n as the AP cluster of user n and the users served by Φ_n is indicated by Ω_n. The users who share the same AP cluster with user n are called intra-cluster users, while the remaining users are called inter-cluster users. In this case, the total signal received by AP m when providing edge services for user n includes the useful signal of user n and the interference signals of inter-cluster users and intra-cluster users, which is calculated as

TABLE 1
LIST OF KEY NOTATIONS

Notation	Definition
$\mathcal{M}=\{1,...,M\}$	The set of APs
$\mathcal{N}=\{1,...,N\}$	The set of users
X	The antenna number of an AP
B_m / $\overline{B}_m$	The bandwidth / remaining bandwidth of AP m
C_m / $\overline{C}_m$	The computing capability / remaining computing capability of AP m
L_n / ρ_n / D_n^t	The task data size / computing density / delay threshold of user n
p_n^u / p_m^a / p_m^i	The transmit power of user n / transmit power of AP m / interference signal power of AP m
$a_{m,n}$ / $b_{m,n}$ / $c_{m,n}$	The clustering / bandwidth allocation variable / computing resource variable between user n and AP m
$r_{m,n}$	The uplink transmission rate of user n
R_m / R	The reputation of AP m / Maximum reputation of APs
φ_{m_1,m_2} / $\overline{\varphi}$	The signal to Interference plus Noise Ratio (SINR) between AP m_1 and AP m_2 / average SINR among APs
L^s / L^b	The data size of state confirmation signal / block
B_m^s / B_m^b	The bandwidth when an AP transmits state confirmation information / transmits block
$D_{m,n}^u$ /$D_{m,n}^e$	The transmission / computing delay of user n when offlaoding to AP m
D_l^g / D_f^s	The delay required by the *Leader* to generate block / delay required by a *Follower* to transmit state transmission signal
D_n^o / D^o / D^c	The offloading delay of user n / average offloading delay of all users / average consensus delay of all APs
$E_{m,n}^u$ / $E_{m,n}^e$	The energy consumption of uplink transmission / task processing when AP m serve user n
E_n^o / E_m^o / E_m^c	The offloading energy consumption of user n / offloading energy consumption of AP m / consensus energy consumption of AP m
E_l^s / E_l^b / E_l^t / E_l^g	The energy consumption for state confirmation transmission / block duplication / data transmission / block generation of the *Leader*
E_f^s / E_f^b / E_f^t	The energy consumption for state confirmation transmission / block duplication / data transmission of a *Follower*
D^a / E^a	The total delay / energy consumption of UC-MEC system

follows

$$\begin{aligned}\boldsymbol{s}_{m,n} =&\sqrt{p_n^u}\boldsymbol{g}_{m,n}x_n+\sum_{\substack{v\neq n,\\ v\in\Omega_n}}\sqrt{p_v^u}\boldsymbol{g}_{m,v}x_v\\ &+\sum_{\substack{w\neq n,\\ w\notin\Omega_n}}\sqrt{p_w^u}\boldsymbol{g}_{m,w}x_w+z,\end{aligned}\tag{1}$$

where p_n^u is the transmit signal power of user n, $\boldsymbol{g}_{m,n}\in\mathbb{C}^{X\times 1}$ is the complex channel coefficient between user n and AP m, z is the Gaussian white noise with mean 0 and variance σ^2, $x_n\sim\mathcal{CN}(0,1)$ is the complex signal send by user n.

Through beamforming technology, the interference of intra-cluster users can be completely eliminated [30]. The beamforming vector of user n be calculated as

$$\boldsymbol{w}_n=\frac{(\boldsymbol{I}_{A|\Phi_n|}-\boldsymbol{G}_{-n}\boldsymbol{G}_{-n}^{\dagger})\boldsymbol{g}_n^n}{\left\|(\boldsymbol{I}_{A|\Phi_n|}-\boldsymbol{G}_{-n}\boldsymbol{G}_{-n}^{\dagger})\boldsymbol{g}_n^n\right\|_2},\tag{2}$$

where $\boldsymbol{g}_n^n=[\cdots,\boldsymbol{g}_{m,n},\cdots]_{m\in\Phi_n}^T$, $\boldsymbol{G}_{-n}=[\cdots,(\boldsymbol{g}_v^n),\cdots]_{v\neq n,v\in\Omega_n}$ and $\boldsymbol{g}_v^n=[\cdots,\boldsymbol{g}_{m,v},\cdots]_{m\in\Phi_n}^T$.

Similar to [3], we introduce the "service center" in the network, which is responsible for managing the AP clusters. When a new user accesses the network, it will send the summary information of its task (data size, computing density and tolerant delay) request to the nearest AP. Then the AP will send the request to the service center in the network. After that, the service center will cooperate with all APs to execute the distributed ADMM algorithm to obtain the optimal clustering decision and resource allocation decision (the details will be explained in Section IV). Then, the resource allocation decision will be sent to all APs and the clustering decision will be sent to all APs and users. When a user moves to a new location and needs to offload, the service center will dynamically allocate AP clusters according to the user's real-time location and communication conditions. Since the focus of this paper is on the wireless transmission process of task offloading, so we ignore the wireless transmission overhead of cluster initialization.

After the clustering is completed, users will obtain the clustering (i.e. the task partition information) profile from the network. Then the user divides the task into several parts in proportion according to the clustering profile, and transmit them to the corresponding AP in the cluster through the uplink. During the offloading process, each AP may exist in multiple AP clusters and serve multiple users at the same time. Each AP will allocate its resources for all users it serves. Specifically, an AP will allocate computing resources to process the tasks and bandwidth to transmit the data. Assume the computing resources and the bandwidth AP m allocates to user n are $c_{m,n}$ and $b_{m,n}$, respectively. Both $c_{m,n}$ and $b_{m,n}$ are continuous variables, and cannot exceed the computing capability and bandwidth of AP m. AP m will not allocate resources to the user who are not within the service range. To allocate bandwidth and ensure that users can send signals to different APs in the AP cluster, the frequency division multiple access (FDMA) method is adopt in this paper [31], [32]. When AP m provides bandwidth for user n to transmit task data, the uplink transmission rate of user n can be expressed by

$$r_{m,n}=b_{m,n}\log_2(1+\frac{p_n^u\left|\boldsymbol{w}_n^H\boldsymbol{g}_n^n\right|^2}{\sum\limits_{w\notin\Omega_n}p_w^u\left|\boldsymbol{w}_n^H\boldsymbol{g}_w^n\right|^2+\left|\boldsymbol{w}_n\right|^2\sigma^2}).\tag{3}$$

After the AP cluster finishes processing the user's task, the results will be integrated and return to the user. After verifying the returned data, the user will pay the AP cluster remuneration according to the amount of resources provided by the AP cluster. Since the size of downlink data is relatively small, we ignore the overhead of downlink data in this paper.

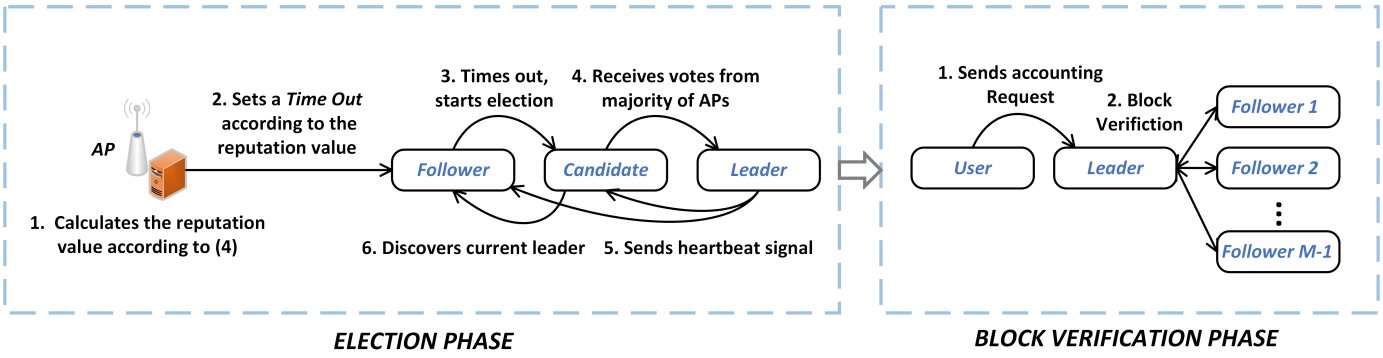


Fig. 2. Illustration of R-RAFT consensus mechanism

2.2 Blockchain-based Resource Trading

To establish a multi-party trusted resource transaction environment and ensure the data safety and privacy of users, blockchain is deployed in our UC-MEC system. Due to the limited computing and storage capability of users, we assume users as the light weight nodes, who are responsible for generating transactions and requests, and only keep part of the ledger. As the full nodes, APs own the complete ledger and are responsible for the consensus process. After the resource trading is completed, the blockchain system will record the trading information (i.e., timestamp, AP ID, user ID, amount of computing resources and bandwidth provided by the AP cluster, amount of remuneration paid by user, information of computing task, et.al. [33]) by executing the consensus mechanism. Furthermore, a smart contract with a series of security mechanisms is deployed to manage the process of resource trading, which can ensure the credibility of resource trading and the data security of users. In this subsection, we first introduce our R-RAFT consensus mechanism, then describe the workflow of our proposed smart contract.

2.2.1 Resource-aware RAFT Consensus Mechanism

RAFT is an algorithm implements in distributed system for reaching consensus [34] among nodes, which has been widely used in Internet of Things (IoT) due to its simplicity of process and high speed of consensus [35]. In RAFT, all nodes are divided into three roles:1) *Follower*; 2) *Candidate*; and 3) *Leader*. The process of RAFT consists of two phases, i.e., the election phase and the block verification phase. In the election phase, all nodes become *Followers* by default and decide to elect the *Leader*. RAFT uses a heartbeat mechanism to trigger the *Leader* election. At first, each node will randomly initialize a *time out* value, and if a node does not receive the heartbeat signal from the *Leader* before the end of the *time out*, it will convert to a *Candidate*. Each *Candidate* will broadcast to all other nodes for canvassing. After receiving the canvass request, each *Follower* will vote for the *Candidate* who send the request, and the *Candidate* with the largest number of votes will become the *Leader*. After the *Leader* election, all other non-*Leader* nodes will become *Followers* and assist the *Leader* to verify the block. In the block verification phase, the user first sends an accounting request to the *Leader*. Then the *Leader* generates and copies the block to other *Followers*. After that, Other *Followers* verify the block and return the confirmation signal to the *Leader*.

As described above, since the *Leader* is responsible for the generation and duplicate of the block, it needs to consume much more resources compared to other *Followers*, thus the *Leader* should be selected carefully. However, the *Leader* is randomly assigned in RAFT which is far from optimal. Besides, RAFT is prone to multiple *Candidates* with the same number of votes during the election phase, which will lead to the failure of the election, and all the nodes need to wait for a *time out* long and restart the election process again, thus greatly increase the consensus overhead. Therefore, we propose a resource-aware RAFT algorithm named R-RAFT to improve the performance of the original RAFT. The illustration of R-RAFT is shown in Fig. 2.

In R-RAFT, after APs finish processing the computing tasks, the remaining computing resources and bandwidth are denoted as $\overline{C}_m$ and $\overline{B}_m$, respectively. During the consensus, the *Leader* and *Followers* need to send state confirmation signals (such as heartbeat packets, vote requests, confirm replies, et.al.) and blocks via the wireless channel. To simplify the communication model of R-RAFT, we assume the data size of state confirmation as L^s. Let L^b and D^i denote the size and the generation interval of blocks, respectively. The bandwidth and the computing resources that the *Leader* needs to consume in the consensus are far more than those of *Followers*. Therefore, we can choose the AP with the most abundant resources as the *Leader*. We use a variable called reputation to describe the communication and the computing capability of APs in the consensus process. Assume all remaining resources of APs are used for consensus, we can define the reputation of AP m as

$$R_m = \frac{\overline{C}_m}{L^b} + \frac{\overline{B}_m}{L^s + L^b}. \tag{4}$$

The larger amount of remaining resources means that the communication and the computing capability of an AP in the consensus process will be stronger, and its reputation will be higher.

To prevent the uncertainty of *Leader* election in the original RAFT, we propose a reputation-based *time out* initialization strategy where *time out* is proportional to $\frac{1}{R_m}$. The node with shorter *time out* can become *Candidate* faster and have a first-mover advantage of becoming *Leader*. In this paper, we use a probability-based model to describe the behavior of

APs in the R-RAFT mechanism. We define the probability of AP m becoming the *Leader* in the election as $P(m = Leader)$, $\forall m \in \mathcal{M}$ which is positively correlated with R_m as

$$P(m = Leader) = \frac{R_m}{R}, \tag{5}$$

where R is the maximum reputation value of all APs, which can be calculated by

$$R = \max_{m \in \mathcal{M}} \{R_m\}. \tag{6}$$

Since all APs will become a *Leader* or a *Follower* after the election, thus $P(m = Follower) = 1 - P(m = Leader)$. Next, we model the transmission process among the *Leader* and the *Followers*. Assume the backhaul channel among APs and the access channel between APs and users adopt different spectrum. Therefore, an AP only needs to consider the interference of other APs. Suppose p_m^a is the signal power sent by AP m, p_m^i is the interference signal power of other APs, and h_{m_1,m_2} is the channel coefficient between AP m_1 and AP m_2. h_{m_1,m_2} can be generated by $h_{m_1,m_2} = \frac{h}{\sqrt{d_{m_1,m_2}^{\gamma}}}$, $h \sim \mathcal{CN}(0,1)$, where γ is the path loss, d_{m_1,m_2} is the distance (in kilometers) between AP m_1 and AP m_2. Similar to [21] and [36], we consider the single hop communication among APs. The signal to interference plus noise ratio (SINR) between AP m_1 and AP m_2 is calculated by

$$\varphi_{m_1,m_2} = \frac{p_m^a |h_{m_1,m_2}|^2}{d_{m_1,m_2} \times \left(\sum_{m \neq m_1,m_2}^{\mathcal{M}} \frac{p_m^i |h_{m,m_2}|^2}{d_{m,m_2}} + \sigma^2 \right)}. \tag{7}$$

In the probability-based model, to calculate the transmission overhead of a *Follower*, we adopt the mean of SINR between *Follower* and all other consensus nodes as the SINR when a specific *Follower* sends signals to the *Leader*

$$\overline{\varphi} = \frac{1}{M-1} \sum_{i \in \mathcal{M}, i \neq m} \varphi_{m,i}. \tag{8}$$

It is worth noting that the data size of status confirmation is much smaller than that of the block [21]. In order to make efficient use of limited bandwidth as much as possible, we assume that APs transmit two different data with fixed bandwidth according to the data size. Thus the bandwidth allocation strategy can be formulated as

$$\frac{L^s}{B_m^s} = \frac{L^b}{B_m^b}, \tag{9a}$$

$$s.t. \quad B_m^s + B_m^b = \overline{B}_m \tag{9b}$$

where B_m^s, B_m^b are the fixed bandwidth used by AP m to transmit status confirmation signal and blocks, respectively. When the *Leader* receives the block confirmation signals returned by the *Followers*, the block will be admitted and attached to the blockchain. All the transaction information in this block will be completely transparent and cannot be changed.

2.2.2 Smart Contract Design for Resource Trading

As mentioned above, the resource allocation of task offloading can be regarded as a transaction process in a resource trading market. A user requests computing resources and bandwidth from APs as a buyer in the market to run the computing-intensive applications and pays APs remuneration. As a seller, an AP provides its own resources and makes profits from users. To further ensure the efficiency and the security of resource transactions in UC-MEC, we design a smart trading contract with multiple security mechanisms.

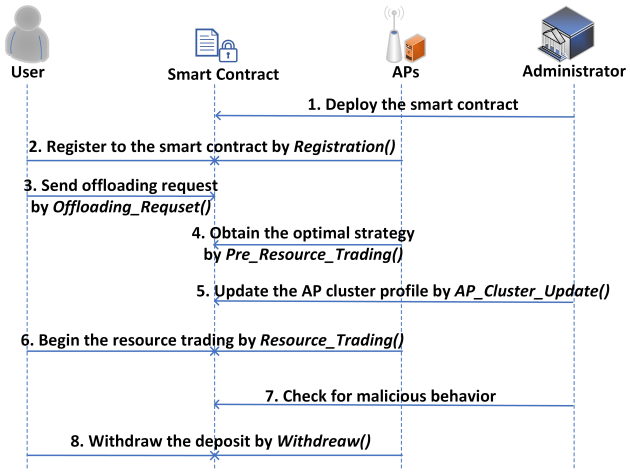


Fig. 3. Flow of the designed smart contract.

The deployment of smart contract needs to cost large gas and selfish APs are unwilling to pay for it. In addition, RAFT can only applies to non-Byzantine problems, thus there needs to be an entity in the blockchain network to supervise transactions. To better encourage the resource trading among APs and users, as well as to further ensure the security of the blockchain network, we introduce an entity called **administrator** into the contract. In this paper, the service center can be the administrator, which is not only responsible for the AP cluster management, but also responsible for the deployment of smart contract, the certificate authentication and the supervision of transactions [37]. It should be noted that the administrator is an independent entity, which is not elected from other APs. Since the administrator does not participate in the resource transaction process, the system is still distributed [38]. The flow of our proposed smart contract is shown in Fig. 3, and the description of smart contract functions is as follows.

Registration(): After the blockchain network is initialized, the smart contract is deployed by the administrator. All nodes in the network choose to join the resource trading market as a user (buyer) or an AP (seller) by calling this method and pay a certain amount of deposit to the smart contract account.

Offloading_Request(): A user sends task offloading request and task information $Task_n = \{L_n, \rho_n, D_n^t\}, n \in \mathcal{N}$ to the contract. This method can only be called by a user.

Pre_Resource_Trading(): An AP request the task information required the ADMM-based clustering and resource allocation optimization algorithm from the contract and runs the algorithm (the details will be explained in Section IV) cooperatively. This method can only be called by an AP.

AP_Cluster_Update(): The smart contract extracts optimal clustering decision from the strategy profiles, then records the current AP clusters of all users in the form of transactions. When the AP cluster of any user is changed,

the smart contract will verify the security of the newly added AP. This method can only be called by the administrator.

Resource_Trading(): APs and users start the resource transaction process. Users first send a request to the contract to obtain the AP cluster information of their own. After that, each user divides the task into several parts in proportion, then transmits them to the corresponding AP in the cluster. During the offloading process, when an AP requests the private information of a user, the smart contract will verify whether the AP belongs to the user's AP cluster. If not, the smart contract will refuse the request of the AP. After finishing the task, the AP cluster will integrate the result and send back to the user through the smart contract. The user will make payment to the AP cluster after verifying the result. After the transaction is completed, APs will perform the R-RAFT to generate the block. This method can only be called by the administrator.

Check for malicious behavior: During the transaction, the administrator will detect the behavior of each APs. If there exists malicious behaviors, such as sudden change of access location, lazy tips (refusing to verify the latest transactions but always verifying the previous fixed transactions), or double spending (spend the same tokens for transactions two or more times), etc, the administrator will record the malicious behaviors in the smart contract. After the transaction is completed, the administrator will check whether each APs has malicious behavior during the transaction. In addition, we can introduce a reputation mechanism into the smart contract. For nodes that comply with the system rules for a period of time, the reputation value will increase. On the contrary, for nodes with abnormal behavior, the reputation will decrease. APs with higher reputation can have more votes in the consensus mechanism, and can get more token rewards.

Withdraw(): All the nodes can call this method to get back the remaining deposit. The administrator will evaluate the behavior of this node in resource trading. If there exists illegal behavior, the deposit withdrawal will be refused. Otherwise, the remaining deposit will be returned normally. For the remaining deposit, part of it can be left to the administrator for deploying smart contract or calling functions. Other parts can be distributed to the APs that comply with the rules according to their reputation.

2.2.3 Safety Analysis

Although RAFT can only applies to non-Byzantine problems, by introducing the administrator, the proposed smart contract can still ensure network security. First, when a user's AP cluster changes, the smart contract will verify the security of the newly added AP, which can prevent unauthenticated or hijacked AP from stealing user's private information. Second, a user's information can only be obtained by the APs within the cluster of the user, thus the APs outside the cluster will not be able to access the user's privacy information, which further protects user's data privacy. Third, users can only get the APs clustering information of their own, thus avoiding the attack from other malicious users.

3 Problem Formulation and Analysis

3.1 Energy Consumption Analysis

3.1.1 Energy Consumption of Task Offloading

Since users will divide their tasks in proportion and transmit them to the corresponding AP in the cluster for processing, for user n, the data transmission delay when offloading computing task to AP m can be given as

$$D^u_{m,n} = \frac{a_{m,n} L_n}{r_{m,n}}. \tag{10}$$

When AP receives the task, it will consume its own computing resources for task processing. The data processing delay when user n offloading its task to AP m can be given as

$$D^e_{m,n} = \frac{a_{m,n} L_n \rho_n}{c_{m,n}}. \tag{11}$$

According to [20] and [39], the energy consumption of a device is positively correlated with the running time. When user n transmit data to AP m, the energy consumption for uplink transmission can be formulated as

$$E^u_{m,n} = \epsilon_c D^u_{m,n}, \tag{12}$$

where ϵ_c is the transmission energy consumption coefficient (in J/ms).

Similarly, the energy consumption of AP m when processing task of user n can be formulated as

$$E^e_{m,n} = \epsilon_p D^e_{m,n}, \tag{13}$$

where ϵ_p represents the computing energy consumption coefficient.

Thus, the total offloading energy consumption of AP m and user n are expressed as

$$E^o_m = \sum_{n \in \mathcal{N}} E^e_{m,n}, \tag{14}$$

$$E^o_n = \sum_{m \in \mathcal{M}} E^u_{m,n}. \tag{15}$$

respectively.

3.1.2 Energy Consumption of Consensus Mechanism

In the election phase of R-RAFT, the *Leader* needs to perform $(2 + D^i)(M - 1)$ times state confirmation communication, including $(M - 1)$ times voting requests, $(M - 1)$ times election confirmations and $D^i(M - 1)$ times heartbeats. In addition, the *Leader* needs to perform $(M - 1)$ times block duplication in the block verification phase.

It's worth noting that APs need to occupy bandwidth and consume energy in each data transmission. Hence, the total energy consumption for data transmission should be equal to the sum of the energy consumed by each transmission. In addition, since APs are densely deployed and transmit data through wireless links in UC-MEC, the transmission rate of each AP can still be obtained through the Shannon formula. According to the bandwidth allocation strategy mentioned in part 2.2.1, the bandwidth of each state confirmation signal transmission is equal to $B^s_l/(2 + D^i)$. Thus the energy consumption for the *Leader* to transmit state confirmation signal can be expressed as

$$E_l^s = \sum_{i \in \mathcal{M}, i \neq Leader} \epsilon_c \frac{(2+D^i)(M-1) \times L^s}{\frac{B_L^s}{2+D^i} \times log_2(1+\varphi_{l,i})}. \quad (16)$$

Similarity, the energy consumption for the *Leader* to duplicate block can be expressed as

$$E_l^b = \sum_{i \in \mathcal{M}, i \neq Leader} \epsilon_c \frac{(M-1) \times L^b}{\frac{B_l^b}{M-1} \times log_2(1+\varphi_{l,i})}. \quad (17)$$

Then the energy consumption of the *Leader* for data transmission is formulated as

$$E_l^t = E_l^s + E_l^b, \quad (18)$$

In addition, the *Leader* needs to consume the computing resources to generate blocks. The computing delay required by the *Leader* to perform block generation is represented by

$$D_l^g = \frac{L^b}{\overline{C}_l}, \quad (19)$$

Then, energy consumption for *Leader* to generate the block is

$$E_l^g = \epsilon_p D_l^g, \quad (20)$$

During the consensus, each *Follower* needs send a vote reply, D^i heartbeat packet replies, and a block confirmation reply, thus total $\left(D^i+2\right)$ times state confirmation transmission. We can get the transmission delay of a *Follower* as

$$D_f^s = \frac{L^s\left(D^i+2\right)}{\frac{\overline{B}_m}{D^i+2} log_2(1+\overline{\varphi})}. \quad (21)$$

Thus, the energy consumption of a *Follower* for data transmission is

$$E_f^t = \epsilon_c D_f^s. \quad (22)$$

Therefore, the energy consumption of AP m in the consensus is expressed as

$$\begin{aligned} E_m^c &= \mathbb{E}[E_l] + \mathbb{E}[E_f] \\ &= \left(E_l^g + E_l^t\right) \times P(m = Leader) \\ &+ \left(E_f^t\right) \times P(m = Follower). \end{aligned} \quad (23)$$

3.1.3 Total Energy Consumption

In our system, total energy consumption includes offloading energy consumption of all users and offloading and consensus energy consumption of all APs. Therefore, the energy consumption of UC-MEC can be formulated as

$$E^a = \sum_{m \in \mathcal{M}} \left(E_m^o + E_m^c\right) + \sum_{n \in \mathcal{N}} E_n^o. \quad (24)$$

3.2 Problem Formulation

During the offloading process, for user $n, n \in \mathcal{N}$, its tasks will be processed in parallel by all APs in the AP cluster. Hence, offloading delay of user n is the maximum value of the service delay of all APs in the AP cluster, which is given by

$$D_n^o = \max\{D_{1,n}^o, D_{2,n}^o, ..., D_{m,n}^o\}, \forall m \in \mathcal{M}, \quad (25)$$

where $D_{m,n}^o = D_{m,n}^u + D_{m,n}^e$.

We assume the clustering profiles of all APs is $\boldsymbol{A} = \{a_{m,n}, \forall m \in \mathcal{M}, n \in \mathcal{N}\}$. The computing resource allocation and bandwidth allocation profiles of all APs are denoted as $\boldsymbol{C} = \{c_{m,n}, \forall m \in \mathcal{M}, n \in \mathcal{N}\}$ and $\boldsymbol{B} = \{b_{m,n}, \forall m \in \mathcal{M}, n \in \mathcal{N}\}$, respectively. Formally, the energy consumption minimization problem in offloading and consensus is formulated as follows:

$$\mathcal{P}1: \quad \min_{\boldsymbol{A},\boldsymbol{C},\boldsymbol{B}} E^a, \quad (26a)$$

$$s.t. \quad D_n^o \leq D_n^t, \forall n \in \mathcal{N}, \quad (26b)$$

$$\sum_{n=1}^{N} c_{m,n} \leq C_m, \forall m \in \mathcal{M}, \quad (26c)$$

$$\sum_{n=1}^{N} b_{m,n} \leq B_m, \forall m \in \mathcal{M}, \quad (26d)$$

$$a_{m,n} \in [0,1], \forall n \in \mathcal{N}, \forall m \in \mathcal{M}, \quad (26e)$$

$$\sum_{m=1}^{M} a_{m,n} = 1, \forall n \in \mathcal{N}. \quad (26f)$$

where (26b) represents that the offloading delay cannot exceed the task delay threshold; (26c) and (26d) represent that the computing resources and the bandwidth allocated by an AP cannot exceed its computing capability and bandwidth; (26e) represents that the clustering variable is a continuous variable in $[0,1]$; (26f) represents that the task offloaded by each user will be fully processed by APs.

As far as the problem $\mathcal{P}1$ is concerned, it is far from easy to handle due to the following challenges. First, There exist various coupling between optimization variables, which including product forms and division forms. In addition, there are coupling of $\boldsymbol{A}$ and $\boldsymbol{B}$, $\boldsymbol{A}$ and $\boldsymbol{C}$, $\boldsymbol{B}$ and $\boldsymbol{A}$, making the original problem non-convex and NP-hard. Second, since the number of variables could reach M^3N^3, the solution space of the original problem is very large. When the network scale increases, the complexity of adopting a centralized algorithm will increase sharply.

4 Joint Optimization for APs Clustering and Resource Allocation

4.1 ADMM-based Parallel Optimization Framework

4.1.1 Problem Transformation

Since optimization variables in $\mathcal{P}1$ are coupled with each other, making $\mathcal{P}1$ a non-convex problem, it is necessary to introduce new auxiliary variables and constraints to decouple and simplify the problem. Expanding Eq. (24) and substituting it into $\mathcal{P}1$, the optimization problem can be reformulated as

$$\min_{\boldsymbol{A},\boldsymbol{C},\boldsymbol{B}} \sum_{m}^{\mathcal{M}} \left(\frac{1}{M} \sum_{n}^{\mathcal{N}} \epsilon_c D_{m,n}^u + \frac{1}{M} \sum_{n}^{\mathcal{N}} \epsilon_p D_{m,n}^e + \left(1 - \frac{R_m}{R}\right) \times \right. \quad (27a)$$

$$\left. \epsilon_c D_f^s + \frac{R_m}{R}\left(\epsilon_p D_l^g + E_l^s + E_l^b\right) \right)$$

$$s.t. \, (26b) - (26f). \quad (27b)$$

Observing the above optimization problem carefully, we can find the following coupling items:

- $D_{m,n}^u$ and $D_{m,n}^e$ contain $\frac{a_{m,n}}{b_{m,n}}$ and $\frac{a_{m,n}}{c_{m,n}}$, respectively;

- $\frac{\overline{C}_m+\overline{B}_m}{\overline{B}_m}$ which is equal to $\frac{C_m-\sum_n^{\mathcal{N}} c_{m,n}}{B_m-\sum_n^{\mathcal{N}} b_{m,n}}$ is included in $\frac{R_m}{R}\left(E_l^s+E_l^b\right)$ and $\frac{\epsilon_c R_m}{R}D_f^s$;
- $\frac{\overline{C}_m+\overline{B}_m}{\overline{C}_m}$ is contained in $\frac{\epsilon_p R_m}{R}D_l^g$;
- There are coupling items $\frac{a_{m,n}}{c_{m,n}}$ and $\frac{a_{m,n}}{b_{m,n}}$ in constraint $(26b)$.

Let $b_{m,n}^{'}=\frac{1}{b_{m,n}}$ and $c_{m,n}^{'}=\frac{1}{c_{m,n}}$, $\forall m\in\mathcal{M},\forall n\in\mathcal{N}$, we can introduce the following auxiliary variables:

$$\chi_{m,n}=a_{m,n}b_{m,n}^{'},\forall m\in\mathcal{M},\forall n\in\mathcal{N}, \tag{28}$$

$$\psi_{m,n}=a_{m,n}c_{m,n}^{'},\forall m\in\mathcal{M},\forall n\in\mathcal{N}, \tag{29}$$

$$\kappa_{m,n}=\frac{C_m-\sum_n^{\mathcal{N}}\frac{1}{c_{m,n}^{'}}}{B_m-\sum_n^{\mathcal{N}}\frac{1}{b_{m,n}^{'}}},\forall m\in\mathcal{M}. \tag{30}$$

According to reformulation linearization technology (RLT) [40], the above three equations can be replaced by the following three constraint sets:

$$\begin{cases}\chi_{m,n}\geq\dfrac{a_{m,n}}{B_m}, & \forall m\in\mathcal{M},\forall n\in\mathcal{N};\\ \chi_{m,n}\leq b_{m,n}^{'}+\dfrac{a_{m,n}}{B_m}-\dfrac{1}{B_m}, & \forall m\in\mathcal{M},\forall n\in\mathcal{N};\\ \chi_{m,n}\leq\dfrac{a_{m,n}}{\beta}, & \forall m\in\mathcal{M},\forall n\in\mathcal{N};\\ \chi_{m,n}\geq\dfrac{a_{m,n}}{\beta}-\dfrac{1}{\beta}+b_{m,n}^{'}, & \forall m\in\mathcal{M},\forall n\in\mathcal{N}.\end{cases} \tag{31}$$

$$\begin{cases}\psi_{m,n}\geq\dfrac{a_{m,n}}{C_m}, & \forall m\in\mathcal{M},\forall n\in\mathcal{N};\\ \psi_{m,n}\leq c_{m,n}^{'}+\dfrac{a_{m,n}}{C_m}-\dfrac{1}{C_m}, & \forall m\in\mathcal{M},\forall n\in\mathcal{N};\\ \psi_{m,n}\leq\dfrac{a_{m,n}}{\beta}, & \forall m\in\mathcal{M},\forall n\in\mathcal{N};\\ \psi_{m,n}\geq\dfrac{a_{m,n}}{\beta}-\dfrac{1}{\beta}+c_{m,n}^{'}, & \forall m\in\mathcal{M},\forall n\in\mathcal{N}.\end{cases} \tag{32}$$

$$\begin{cases}\kappa_m\geq\dfrac{C_m-\sum_n^{\mathcal{N}}\frac{1}{c_{m,n}^{'}}}{B_m}, & \forall m\in\mathcal{M};\\ \kappa_m\leq\dfrac{C_m}{B_m-\sum_n^{\mathcal{N}}\frac{1}{b_{m,n}^{'}}}-\dfrac{C_m}{B_m}, & \forall m\in\mathcal{M};\\ \kappa_m\leq\dfrac{C_m-\sum_n^{\mathcal{N}}\frac{1}{c_{m,n}^{'}}}{\beta}, & \forall m\in\mathcal{M};\\ \kappa_m\geq\dfrac{C_m}{B_m-\sum_n^{\mathcal{N}}\frac{1}{b_{m,n}^{'}}}-\dfrac{\sum_n^{\mathcal{N}}\frac{1}{c_{m,n}^{'}}}{\beta}, & \forall m\in\mathcal{M}.\end{cases} \tag{33}$$

where β is an infinitesimal positive number.

It's easy to prove that the above three constraint sets are equivalent to equations (28) - (30) according to [20] and [41]. In addition, constraint (26b) can also be transformed into the following form:

$$\frac{L_n\chi_{m,n}}{log_2(1+r_{m,n})}+L_n\rho_n\psi_{m,n}\leq D_n^t,\forall m\in\mathcal{M},\forall n\in\mathcal{N} \tag{34}$$

Constraints (26c), (26d) can be transformed to the following forms:

$$\sum_n^{\mathcal{N}}\frac{1}{c_{m,n}^{'}}\leq C_m,\forall m\in\mathcal{M} \tag{35}$$

$$\sum_n^{\mathcal{N}}\frac{1}{b_{m,n}^{'}}\leq B_m,\forall m\in\mathcal{M} \tag{36}$$

Substituting (28) - (36) into (27), the optimization objective can be expressed as the sum of M functions in the same form. Therefore, the optimization problem $\mathcal{P}1$ can be reformulated as

$$\mathcal{P}2: \min_{\boldsymbol{A},\boldsymbol{C}',\boldsymbol{B}',\boldsymbol{X},\boldsymbol{\Psi},\boldsymbol{K}}\sum_m^{\mathcal{M}}V_m(\boldsymbol{B}',\boldsymbol{X},\boldsymbol{\Psi},\boldsymbol{K}) \tag{37a}$$

$$s.t.\quad (26e),(26f),(31)-(36). \tag{37b}$$

As far as *P2* is concerned, the numbers of variables and constraints reach $3MN+M$ and $9MN+6M$, respectively. When the number of users and APs increases, the worse-case complexity will increase exponentially, and the time to obtain the optimal solution will be unbearable. As a mature optimization framework, ADMM [42] can be the reliable solution for solving large-scale convex or non-convex problems. Through the transformation of ADMM method, the originally complex problem can be decomposed into the sum of several simpler sub-problems, each sub-problem can be allocated to a node for parallel processing. Compared with other classical centralized algorithms, ADMM can significantly reduce the computational complexity and running time. In this paper, we develop a novel ADMM-based parallel optimization framework to solve *P2*. To reduce complexity, The proposed framework decompose the original problem into M same sub-problems and each AP can solve a sub-problem in parallel.

4.1.2 The Update of Global Variables

Clearly, the existence of constraint (26f) makes each AP need to know the global clustering information (clustering decision of other APs), so that the problem cannot be decomposed. Thus, we introduce a local copy of a, denoted as $\hat{a}$, which satisfies the following formula

$$\hat{a}_{m,n}=a_{m,n},\forall m\in\mathcal{M},\forall n\in\mathcal{N}. \tag{38}$$

Then the constraint (26f) is transformed into the following form

$$\sum_m^{\mathcal{M}}\hat{a}_{m,n}=1,\forall n\in\mathcal{N}. \tag{39}$$

Thus, the optimization problem can be further reformulated as

$$\mathcal{P}3: \min_{\hat{\boldsymbol{A}},\boldsymbol{C}',\boldsymbol{B}',\boldsymbol{X},\boldsymbol{\Psi},\boldsymbol{K}}\sum_m^{\mathcal{M}}V_m(\boldsymbol{A},\boldsymbol{B}',\boldsymbol{X},\boldsymbol{\Psi},\boldsymbol{K}) \tag{40a}$$

$$s.t.\quad (26e),(31)-(36),(38),(39). \tag{40b}$$

Let $\boldsymbol{\Pi}=\{\boldsymbol{A},\boldsymbol{C}',\boldsymbol{B}',\boldsymbol{X},\boldsymbol{\Psi},\boldsymbol{K}\}$ and $\ddot{\boldsymbol{\Pi}}=\{\boldsymbol{A},\boldsymbol{B}',\boldsymbol{X},\boldsymbol{\Psi},\boldsymbol{K}\}$. Then, the augmented Lagrangian function of $\mathcal{P}3$ is represented as

$$\begin{aligned}\mathcal{L}(\ddot{\boldsymbol{\Pi}},\hat{\boldsymbol{A}},\boldsymbol{\lambda})=&\sum_m^{\mathcal{M}}V_m(\ddot{\boldsymbol{\Pi}})+\sum_m^{\mathcal{M}}\sum_n^{\mathcal{N}}\lambda_{m,n}(a_{m,n}-\hat{a}_{m,n})\\&+\frac{q}{2}\sum_m^{\mathcal{M}}\sum_n^{\mathcal{N}}(a_{m,n}-\hat{a}_{m,n})^2\end{aligned} \tag{41}$$

where $\boldsymbol{\lambda} = \{\lambda_{m,n}, \forall m \in \mathcal{M}, \forall n \in \mathcal{N}\}$ is the Lagrange multiplier, and $q \geq 0$ is the augmented Lagrangian parameter, which affects the convergence of ADMM algorithm.

According to [14], [43], the global variable $\mathbf{\Pi}$ in $\mathcal{P}3$ is updated at the $(t+1)$-th iteration by solving the following optimization problem:

$$\mathcal{P}4: \quad \min_{\mathbf{\Pi}} \mathcal{L}(\ddot{\mathbf{\Pi}}, \hat{\boldsymbol{A}}^{(t)}, \boldsymbol{\lambda}^{(t)}) \tag{42a}$$

$$s.t. \quad (26e), (31)-(36). \tag{42b}$$

where (t) represents the index of the number of iterations.

For each AP m, we define $\boldsymbol{a}_m = \{a_{m,n}, \forall n \in \mathcal{N}\}$ as the clustering decision vector, $\boldsymbol{b_m^{'}} = \{b_{m,n}^{'}, \forall n \in \mathcal{N}\}$ as the bandwidth allocation vector and $\boldsymbol{c_m^{'}} = \{c_{m,n}^{'}, \forall n \in \mathcal{N}\}$ as the computing resource allocation vector, respectively. Similarly, we define $\boldsymbol{\chi}_m = \{\chi_{m,n}, \forall n \in \mathcal{N}\}$, $\boldsymbol{\psi}_m = \{\psi_{m,n}, \forall n \in \mathcal{N}\}$, $\boldsymbol{\pi_m} = \{\boldsymbol{a}_m, \boldsymbol{b}_m^{'}, \boldsymbol{c}_m^{'}, \boldsymbol{\chi}_m, \boldsymbol{\psi}_m, \boldsymbol{\kappa}_m\}$, and $\ddot{\boldsymbol{\pi}}_m = \{\boldsymbol{a}_m, \boldsymbol{b}_m^{'}, \boldsymbol{\chi}_m, \boldsymbol{\psi}_m, \boldsymbol{\kappa}_m\}$. Then, we can rewrite the augmented Lagrangian function of $\mathcal{P}4$ as

$$\mathcal{L}(\ddot{\mathbf{\Pi}}, \hat{\boldsymbol{A}}^{(t)}, \boldsymbol{\lambda}^{(t)}) = \sum_{m}^{\mathcal{M}} \mathcal{F}_m(\ddot{\boldsymbol{\pi}_m}). \tag{43}$$

Secondly, by analyzing the problem $\mathcal{P}4$, we can get that the objective function of $\mathcal{P}4$ is convex for all variables. However, the second and the fourth item of constraint (33) are non-convex, making the problem still intractable. Therefore, we take a relaxation for (33) by adding it into the objective function of $\mathcal{P}4$ in the form of a penalty term, which can be expressed as

$$\begin{aligned}\Upsilon(\boldsymbol{K}, \boldsymbol{B}^{'}, \boldsymbol{C}^{'}) = \zeta_1 \left(\frac{C_m - \sum_n^{\mathcal{N}} \frac{1}{c_{m,n}^{'}}}{B_m} - \kappa_m \right) \\ + \zeta_2 \left(\kappa_m - \frac{C_m}{B_m - \sum_n^{\mathcal{N}} \frac{1}{b_{m,n}^{'}}} + \frac{C_m}{B_m} \right).\end{aligned} \tag{44}$$

where ζ_1 and ζ_2 are penalty coefficients. It's easy to prove that when the values of ζ_1 and ζ_2 are large enough, the global optimization problem $\mathcal{P}4$ can be transformed into the following form

$$\mathcal{P}4^{'}: \quad \min_{\mathbf{\Pi}} \sum_{m}^{\mathcal{M}} \mathcal{F}_m(\ddot{\boldsymbol{\pi}}_m) - \tilde{\Upsilon}\left(\boldsymbol{K}, \boldsymbol{B}^{'}, \boldsymbol{C}^{'}\right) \tag{45a}$$

$$s.t. \quad (26e), (31)-(36), \tag{45b}$$

where $\tilde{\Upsilon}(\boldsymbol{K}, \boldsymbol{B}^{'}, \boldsymbol{C}^{'}) = -\Upsilon(\boldsymbol{K}, \boldsymbol{B}^{'}, \boldsymbol{C}^{'})$. Obviously, $\tilde{\Upsilon}(\boldsymbol{K}, \boldsymbol{B}^{'}, \boldsymbol{C}^{'})$ is a convex function. To sum up, the optimization problem of updating the global variable $\boldsymbol{\pi_m}$ independently and in parallel for each AP m is expressed as

$$\mathcal{P}5: \quad \min_{\boldsymbol{\pi}_m} \mathcal{F}_m(\ddot{\boldsymbol{\pi}}_m) - \tilde{\Upsilon_m}(\boldsymbol{\kappa}_m, \boldsymbol{b}_m^{'}, \boldsymbol{c}_m^{'}) \tag{46a}$$

$$s.t. \quad (26e), (31)-(36). \tag{46b}$$

The optimization objective of $\mathcal{P}5$ is the difference of two convex functions, which is a D.C. Programming Problem [44]. Similar to [41], we propose a sub-gradient iteration algorithm to solve $\mathcal{P}5$. The details of the proposed algorithm are summarized in **Algorithm 1**.

Algorithm 1 Sub-gradient Based Iteration Algorithm

Input: Initial a feasible value $\left\{\boldsymbol{k}_m^{(n)}, \boldsymbol{b}_m^{'(n)}, \boldsymbol{c}_m^{'(n)}\right\}$for AP m where $m \in \mathcal{M}$ and $n = 0$.

Output: The global variable $\boldsymbol{\pi}_m$ for AP m where $m \in \mathcal{M}$.

1: **repeat**

2: Solve the following convex problem named $\mathcal{P}5^{'}$:

$$\begin{aligned}\min_{\boldsymbol{\pi}_m} \Big\{ & \mathcal{F}_m(\ddot{\boldsymbol{\pi}}_m) - \nabla_{b_m^{'}} \tilde{\Upsilon}^{(n)}(b_m^{'} - b_m^{'(n)}) \\ & - \nabla_{c_m^{'}} \tilde{\Upsilon}^{(n)}(c_m^{'} - c_m^{'(n)}) - \nabla_{\kappa_m} \tilde{\Upsilon}^{(n)} \\ & \times (\kappa_m - \kappa_m^{(n)}) - \tilde{\Upsilon}^{(n)} \Big\} \\ s.t. \quad & (26e), (31)-(36).\end{aligned}$$

3: Obtain the optimal solution for the $(n+1)$-th iteration $\{\boldsymbol{\pi}_m\}^{(t+1)}$.

4: $n = n + 1$.

5: **until** Global variable $\boldsymbol{\pi}_m$ converges.

In **Algorithm 1**, ∇ denotes the sub-gradient of the corresponding variable of the objective function, and $\tilde{\Upsilon}^{(n)} = \tilde{\Upsilon}(\boldsymbol{\kappa}_m^{(n)}, \boldsymbol{b}_m^{'(n)}, \boldsymbol{c}_m^{'(n)})$.Since $\mathcal{P}5^{'}$ is convex, and all constraints of it are linear, we can use convex optimization tools such as CVX [45] to solve it.

4.1.3 The Update of Local Variables

After each AP independently solves the global variables in parallel, the results will be aggregated to a central node. According to [46], we can set up an anchor AP, and the integrated MEC on the anchor AP can be the central node. After the central node collects and integrates the global variables, it will be responsible for updating the local variables in the $(t+1)$-th iteration. The updating of local variables $\hat{\boldsymbol{A}}$ depends on solving the following optimization problem

$$\mathcal{P}6: \quad \min_{\hat{\boldsymbol{A}}} \mathcal{L}(\hat{\boldsymbol{A}}, \hat{\mathbf{\Pi}}^{(t+1)}, \boldsymbol{\lambda}^{(t)}) \tag{47a}$$

$$s.t. \quad (39), \hat{a}_{m,n} \in [0,1], \forall m \in \mathcal{M}, \forall n \in \mathcal{N}. \tag{47b}$$

After removing some constants, $\mathcal{P}6$ is equivalent to solving the following problem

$$\mathcal{P}6^{'}: \quad \min_{\hat{\boldsymbol{A}}} \hat{\mathcal{J}}(\hat{\boldsymbol{A}}) \tag{48a}$$

$$s.t. \quad (39), \hat{a}_{m,n} \in [0,1], \forall m \in \mathcal{M}, \forall n \in \mathcal{N}. \tag{48b}$$

where

$$\begin{aligned}\hat{\mathcal{J}}(\hat{\boldsymbol{A}}) = & \sum_{m}^{\mathcal{M}} \sum_{n}^{\mathcal{N}} \lambda_{m,n}^{(t)} \left(a_{m,n}^{(t+1)} - \hat{a}_{m,n}\right) \\ & + \frac{q}{2} \sum_{m}^{\mathcal{M}} \sum_{n}^{\mathcal{N}} \left(a_{m,n}^{(t+1)} - \hat{a}_{m,n}\right)^2.\end{aligned} \tag{49}$$

Obviously, the problem $\mathcal{P}6^{'}$ is also convex. Therefore, the local variables $\hat{\boldsymbol{A}}$ can be solved with some mature convex optimization tools.

4.1.4 The Update of Dual Variables

After the global variables and local variables are obtained, the update of dual variables in the $(t+1)$-th iteration can be expressed as

$$\lambda_{m,n}^{(t+1)} = \lambda_{m,n}^{(t)} + q\left(a_{m,n}^{(t+1)} - \hat{a}_{m,n}^{(t+1)}\right), \forall m \in \mathcal{M}, \forall n \in \mathcal{N}. \tag{51}$$

In a word, the efficient APs clustering and resource allocation strategy are obtained through the sequential iteration of global variables, local variables, and dual variables. The ADMM-based parallel optimization algorithm is summarized in **Algorithm 2**.

4.2 Performance Analysis

4.2.1 Complexity Analysis

In our proposed ADMM-based parallel iteration optimization algorithm, i.e. **Algorithm 2**, each AP is able to solve $\mathcal{P}5$ independently and in parallel, which greatly reduces the computational complexity. Moreover, for each AP, $\mathcal{P}5$ can be viewed as a D.C Programming Problem and solved by a sub-gradient iterative algorithm, as described in **Algorithm 1**. In $\mathcal{P}5^{'}$, there are $5N+1$ optimization variables and $10N+2$ convex constraints for each AP. We assume that $\mathcal{P}5^{'}$ requires n^d iterations to converge. Then, the complexity of $\mathcal{P}5^{'}$ is $\mathcal{O}(\max_{m\in\mathcal{M}}(5N+1)(10N+2)n^D) = \mathcal{O}(\max_{m\in\mathcal{M}} N^2)$. Similarly, for the optimization problem $\mathcal{P}6$, the number of optimization variables is MN, and the number of constraints is $N+MN$. Assuming that n^l steps are needed to solve the optimization problem, then its complexity is $\mathcal{O}(MN(N+MN)n^l) = \mathcal{O}(n^l(M^2N^2+MN^2))$. For the updating of dual variables, we assume that n^d steps are required to solve (51), then its computational complexity is $\mathcal{O}(n^dMN)$. Assuming that **Algorithm 2** requires t^a times to converge, then the complexity of overall algorithm is $\mathcal{O}(N^2+n^l(M^2N^2+MN^2)+n^dMN) = \mathcal{O}(M^2N^2)t^a$, which is greatly reduced compared using centralized algorithm to slove $\mathcal{P}1$ directly.

4.2.2 System delay Analysis

For delay-sensitive applications, the tasks offload to the edge server needs to be completed within a limited time, i.e., the offloading delay cannot be greater than the tolerance delay of the tasks. Furthermore, all APs need to complete the consensus process before the next round of offloading. Therefore, the system delay is also need to be considered in the blockchain-enabled UC-MEC system. As mentioned above, the offloading delay of user n has given by Eq. (25). For the consensus delay, the data transmission between *Leader* and *Follower* are also in parallel, thus the consensus delay of the proposed system is

$$\begin{aligned} D^c = \max_{m\in\mathcal{M}} \Bigg(& \max_{j\in\mathcal{M},j\neq m} \frac{(M-1)\,L^s\left(D^i+2\right)^2}{B_m^s \log_2\left(1+\varphi_{m,j}\right)} \\ & + \max_{j\in\mathcal{M},j\neq m} \frac{\left(D^i+2\right)^2 L^s}{\overline{B}_j \log_2\left(1+\overline{\varphi}\right)} \\ & + \max_{j\in\mathcal{M},j\neq m} \frac{(M-1)\,L^b}{B_m^b \log_2\left(1+\varphi_{m,j}\right)} + D_m^g \Bigg). \end{aligned} \tag{52}$$

Therefore. the total delay of user n is equal to the sum of the offloading delay and the consensus delay.

Algorithm 2 ADMM-Based Parallel Iteration Optimization Algorithm

Initialization :
1. $t=1$, $\lambda^{(t)} = \mathbf{0}$ and $t_{max} = 100$.
2. Select a suitable iterative stopping conditions γ of the MEC anchor. And choose a penalty factor q of the ADMM algorithm and send it to all APs.
3. Each AP m determines an initial feasible solution $\hat{\boldsymbol{A}}^{[1]}$ satisfying (26f)

while $|\boldsymbol{A}-\hat{\boldsymbol{A}}|_2 \geq \gamma$ and $t \leq t_{max}$ **do**
1. Update global variables: each AP m ($m \in \mathcal{M}$) executes **Algorithm 1** to obtain the global variable $\boldsymbol{\Pi}^{(t+1)}$ and uploads it to the anchor node, respectively.
2. Update local variables: the anchor node updates the local variable $\hat{\boldsymbol{A}}^{(t+1)}$ based on (48).
3. Update dual variables: the dual multiplier $\lambda^{(t+1)}$ is updated based on (51).
4. Update $t = t+1$

end while

Output : the optimal clustering and resource allocation solution $\{\boldsymbol{\Pi}\}^*$.

5 Simulation Results and Discussion

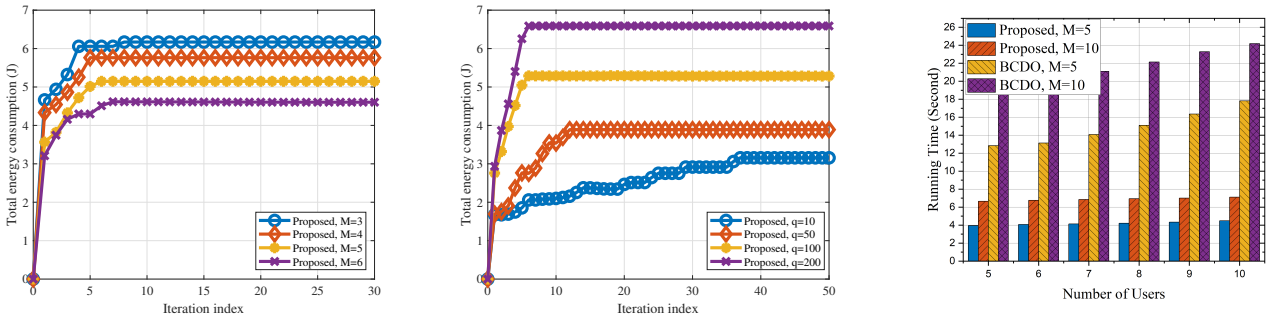
In this section, the simulation results and discussion are presented to evaluated the performance of our algorithm. First, the simulation settings and comparison schemes are presented. Then it follows the convergence, delay, and energy consumption performance with various parameters. The simulation code is available on https://github.com/qlt315/Blockchain-enabled-UCMEC.

5.1 Simulation Settings

TABLE 2
SIMULATION PARAMETERS

Parameter	Value
bandwidth of AP m B_m	[10,50] MHz
computing resource of AP m C_m	[5,20] GHz
Task data size L_n	[100,200] kbits
computing density ρ_n	[1000,2000]
Maximum tolerated delay D_n^t	[100,150] ms
Transmission power of user n p_n^u	100 mW
Transmission power of AP m p_m^a	200 mW
Interference power of AP m p_m^I	20 mW
Transmission energy consumption coefficient ϵ_c	1.5 J/s
computing energy consumption coefficient ϵ_p	1 J/s
Gaussian white noise power z	-174 dBm/Hz
Size of state confirmation data L^s	10 kbits
Block size L^b	500 kbits
Block interval D^i	1 block/s
Maximum tolerance error γ	0.01
Maximum iterations t_{max}	100

We implement the simulations using MATLAB R2021b on a computer with a AMD Ryzen 3600 CPU running on a processor speed of 3.6 GHz, and 16 GB RAM. Assume that all APs and users are uniformly distributed in a 200 m $\times$ 200 m circle and the path loss model is $121.8 + 37.6\log d$



(a) Convergence performance under different AP numbers ($q = 100$).

(b) Convergence performance under different Lagrangian penalty factor q ($M = 5$).

(c) Running time of different schemes

Fig. 4. Convergence and running time performance comparison.

(d in kilometers) dB. To better simulate the heterogeneity of APs and the diversity of offloading tasks, the bandwidth of APs are evenly distributed in [10,50] MHz [41], and the computing resources are evenly distributed in $[5, 20]$ GHz [39]. The size of user's task, the computing density, and the maximum tolerated delay are uniformly distributed in $[100, 200]$ kbits, $[1000, 2000]$ CPU cycles/bit, and $[100, 150]$ milliseconds, respectively. During the offloading and the consensus process, the transmission power of user, the transmission power of AP, and the interference signal power of other APs are 100 mW, 200 mW, 20 mW [47], respectively. We assume the energy consumption coefficient of unit computing delay and transmission delay are 1 J/s and 1.5 J/s respectively, the power of Gaussian white noise is -174 dBm/Hz [14], and the size of state confirmation data is 10 kbits. In the blockchain system, the block size is generally no more than 1 kbits. We assume that the block size is 500 kbits and the block interval is 1 second. In the proposed ADMM based algorithm, we assume the maximum tolerance error is 0.01 [41] and the maximum number of iterations is 100. In order to ensure the accuracy of the results, we take the average value of 10 simulations. The parameters used in the simulation are summarized in TABLE II.

5.2 Comparison Schemes

In order to better reflect the effectiveness of our proposed scheme, we also simulate the following three schemes for comparison. It should be noted that all the comparison schemes are deployed with blockchain to ensure the fairness of comparison.

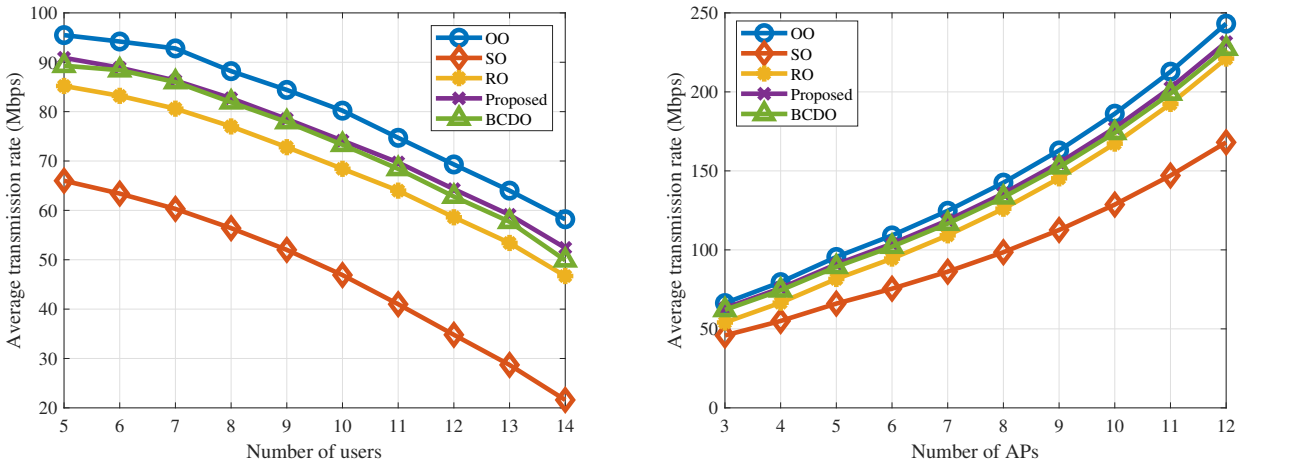
- **Single Offloading (SO)**: To verify the effectiveness of the user-centric cooperative offloading method compared with traditional cellular-based MEC, each user will offload computing task to a single AP in this scheme[33]. Hence, the clustering decision variable $\mathcal{A}$ satisfies $\sum_{m}^{\mathcal{M}} a_{m,n} = 1, \forall m \in \mathcal{M}, \forall n \in \mathcal{N}$, $a_{m,n} \in \{0, 1\}$.
- **Block Coordinate Descent-based Offloading (BCDO)**: In this scheme, to verify the effectiveness of the proposed distributed ADMM optimization framework, we use centralized block coordinate descent (BCD) [48], [49], to solve $\mathcal{P}2$. Relying on the BCD technique, the original problem is decoupled into three sub-problems for alternatively optimizing the AP clustering, computing resource and bandwidth allocation decisions.
- **Optimal Offloading (OO)**[35]: To verify the necessity of joint optimization of the offloading process and the consensus process, the optimization objective in this scheme only includes offloading energy consumption of all APs and users, i.e., $\sum_{m\in\mathcal{M}} E_m^o + \sum_{n\in\mathcal{N}} E_n^o$
- **RAFT-based Offloading (RO)**: To verify the effectiveness of our proposed R-RAFT consensus mechanism, we select RAFT consensus algorithm for comparison, thus the probability of each node becoming the *Leader* is completely stochastic. We assume that p_m = *Leader* is a random number uniformly distributed in $(0, 1]$.

5.3 Simulation Results

5.3.1 Convergence and Running Time Performance

We assume that the number of users $N = 5$. The convergence processes of our proposed joint optimization scheme based on ADMM are shown in Fig. 4 (a) and Fig. 4 (b). We separately validate the effects of the Lagrangian penalty factor q and the number of APs M on the convergence performance. In addition, We compare the actual running times of BCD-based scheme and the propsoed ADMM-based scheme under different parameter settings in Fig. 4 (c).

As shown in Fig. 4 (a), the proposed scheme can rapidly converge within 10 steps. As the number of APs increases, the total energy consumption will decrease, but the downward trend will slow down. This is because the increase of APs will enlarge the available computing and bandwidth resources in the network, thus reducing the computing and transmission delay and energy consumption. However, the number of communications among APs in consensus will also increase accordingly, so the consensus energy consumption will increase and gradually offset the gain of the reduction of the other parts of the energy consumption. Therefore, MEC service providers and communication



(a) Average transmission rate under different user numbers.

(b) Average transmission rate under different AP numbers.

Fig. 5. Average transmission rate performance under different user and AP numbers.

service operators need to carefully decide the number or deploy density of AP.

It can be seen from Fig. 4 (b) that the value of the Lagrangian penalty factor q will affect the convergence performance of the ADMM algorithm. When q increases, the speed of convergence will increase, however, the total energy consumption will also increase. This is because the local copy of the clustering decision variable $\hat{\boldsymbol{A}}$ will approach the global variable at a faster speed when q increases, thereby approximating the stopping condition of the algorithm faster. However, the increase in the convergence speed will sacrifice the quality of the solution, making the total energy consumption becomes relatively large. Therefore, the value of q should be chosen carefully to trade off the convergence speed and the performance of the algorithm.

The average running times of the proposed scheme and BCD-based scheme for the different numbers of users or APs are given in Fig. 4 (c). It can be observed that the running times of the proposed scheme are much less than that of the BCD under the same parameter setting. This is because the proposed scheme can distribute the sub-problems to all APs for parallel optimization, while in BCD the problem can only be centrally calculated by one node. Moreover, with the increase in the number of users and APs, the size of the optimization variables will also increase, and the larger search spaces will increase the running time of the schemes.

5.3.2 Uplink Transmission Rate Performance

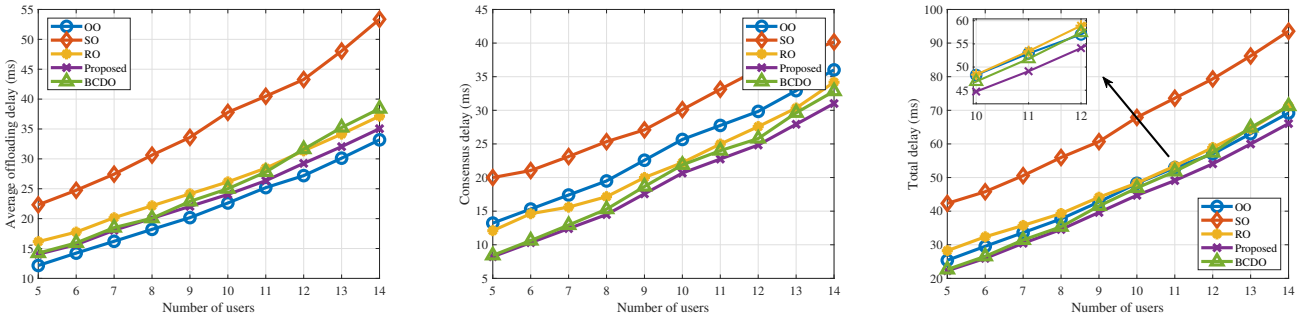
We now focus on the evaluation of the average uplink transmission rate of users. If not specifically mentioned, the number of users and APs are set to 5, and the settings of other parameters are consistent with TABLE 2. Fig. 5 shows the average uplink transmission rate versus the number of users and APs. From Fig. 5 (a), we can see that when the number of users increases, the average transmission rate of all users will decrease. This is because the bandwidth resources allocated to each user will decrease and the inter-cluster interference will increase. As can be seen from Fig. 5 (b), when the number of APs increases, the available bandwidth resources of the system will increase, and the average transmission rate of users will be greatly improved. In addition, it can be seen that **SO** has a lower transmission rate than other schemes, which shows that the user-centric cooperative transmission mode is superior to the traditional transmission mode. The performance of our scheme is second only to **OO**, and **BCDO** has the same performance as the proposed scheme.

5.3.3 Delay Performance

Next, we analyse the delay performance when the number of users changes. We compare the average offloading delay $D^o = \frac{1}{N}\sum_{n\in\mathcal{N}} D_n^o$, the consensus delay D^c and the total delay $D^a = D^o + D^c$, respectively.

It can be observed from Fig. 6 that when the number of users increases, the total delay, the average offloading delay of users, and the consensus delay will also increase. This is because the increase in offloading tasks causes resource-constrained APs to spend more time processing them. Hence, the offloading delay will increase. Similarly, since APs spend more resources in the offloading process, the available computing resources and bandwidth in the consensus process will reduce, resulting the increase of consensus delay. Moreover, the total delay in **SO** is significantly higher than that in the other three schemes. The reason is that although transmission delay will reduce when users select the closest AP, the service quality is not optimal. In contrast, by introducing a cooperative APs clustering mechanism, users can be served by multiple APs, thereby significantly improving the efficiency of resource utilization and reducing the total delay.

As we can see, **OO** can achieve the lowest offloading delay. However, it does not consider the consensus delay and focuses too much on task offloading, making the available resources in the consensus process insufficient, resulting in a relatively large total delay. In **RO**, the optimization of task offloading and consensus is jointly considered, thus the

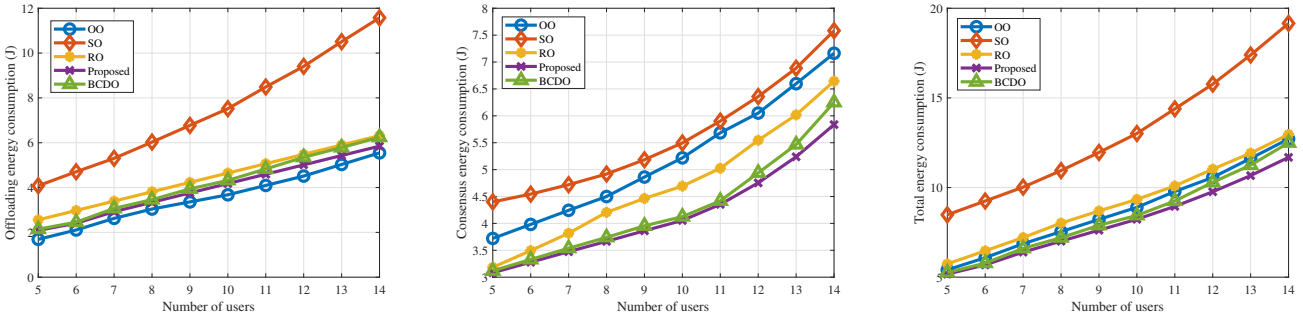


(a) Offloading delay under different user numbers.

(b) Consensus delay under different user numbers.

(c) Total delay under different user numbers.

Fig. 6. Delay performance under different user numbers.



(a) Offloading energy consumption under different user numbers.

(b) Consensus energy consumption under different user numbers.

(c) Total energy consumption under different user numbers.

Fig. 7. Energy consumption under different user numbers.

offloading delay is less than that in **SO**, and the consensus delay is also less than that in **SO** and **OO**. However, RAFT does not consider the resource usage of APs when selecting *Leader*, thereby election conflict is prone to happen, resulting a relatively large consensus delay.

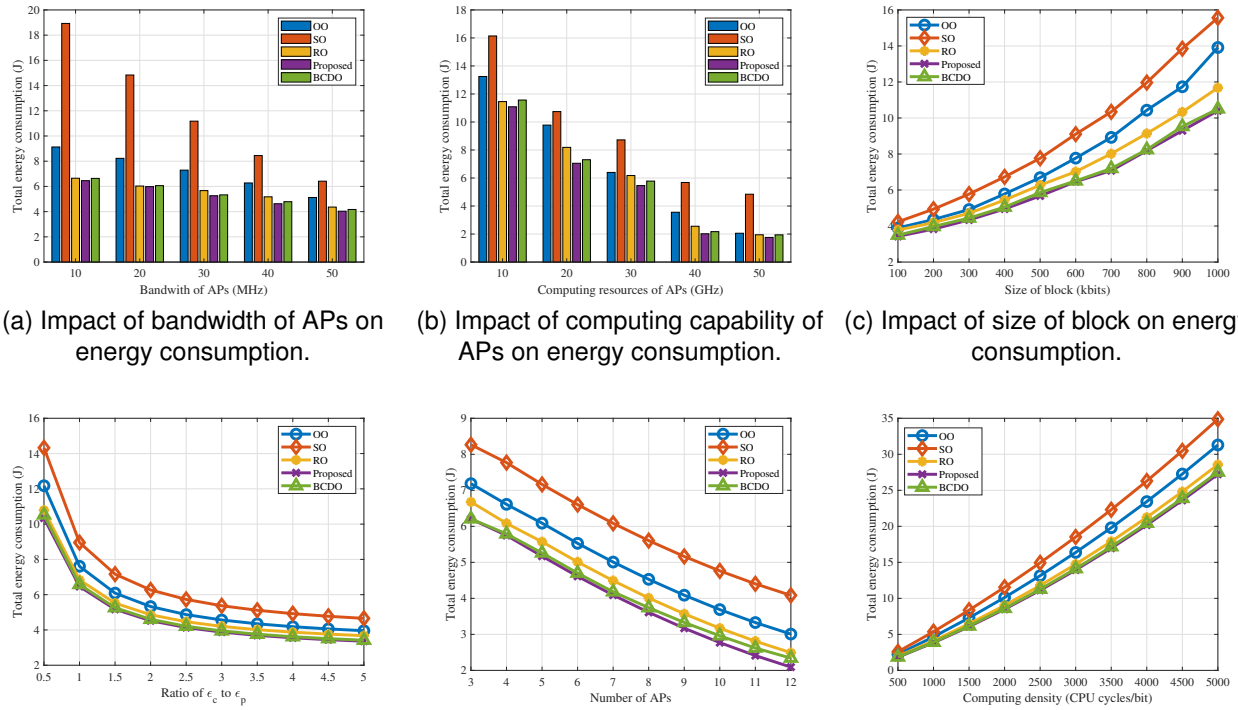
In contrast, our proposed scheme uses an improved resource-aware RAFT consensus mechanism while jointly considering the optimization of task offloading and consensus. When $N = 14$, the offloading delay of our proposed scheme is only 5.67% more than that of **OO**, but 33.96% less than that of **SO**, which can still be maintained within 35 ms. Consensus delay of our proposed scheme is the smallest of the four schemes, which can keep at 31.01 ms, 11.43% less than that of **SO**. The total delay is also the smallest compared to reference schemes, which is about 66 ms and is 21.4% less than the total delay of **SO**. In addition, **BCDO** is basically consistent with the proposed scheme when the number of users is small, but when the number of users increases, the solution space will become larger and the performance of **BCDO** will deteriorate. Therefore, it can be concluded that our scheme can be applied to delay-sensitive applications and maintain a low-latency performance.

5.3.4 Energy Consumption Performance

In this section, we evaluate the impact of different parameters on the performance of the offloading energy consumption $E^o = \sum_{n\in\mathcal{N}} E_n^o + \sum_{m\in\mathcal{M}} E_m^o$, the consensus energy consumption $E^c = \sum_{m\in\mathcal{M}} E_m^c$ and the total energy consumption E^a.

As shown in Fig. 7, when the number of users increases, the energy consumption in offloading and consensus will increase. The reason is that energy consumption and delay are positively correlated, when the offloading and the consensus delay increase, the corresponding energy consumption will increase too. When the number of users $N = 14$, the offloading energy consumption of our proposed scheme is only 5.42% more than that of **OO**, but 49.52% less than that of **OO**, consuming 5.84 J of energy when $N = 14$. Our proposed scheme can not only consume the least energy in the consensus, 25.54% less than that of **SO**, consume 5.83 J of energy, but also consume the least energy in total, 39.01% less than that of **SO**, consuming 11.68 J of energy in total.

Then we investigate the performance of total energy consumption under different bandwidth and computing resources. As shown in Fig. 8 (a) and Fig. 8 (b), when the bandwidth and the computing resources of APs increase, the total energy consumption will gradually decrease. This is because APs can complete task offloading and consensus in a shorter time, thereby consuming less energy. Furthermore, it can be seen that our proposed scheme can achieve the minimum energy consumption, when the bandwidth of APs is 50 MHz, the energy consumption of our proposed scheme is 37.13% less than that of **SO**, consuming 4.03 J. When computing capability is 50 CPU cycles frequency in



(a) Impact of bandwidth of APs on energy consumption.

(b) Impact of computing capability of APs on energy consumption.

(c) Impact of size of block on energy consumption.

(d) Impact of ratio of ϵ_c to ϵ_p on energy consumption.

(e) Impact of number of APs on energy consumption.

(f) Impact of computing density on energy consumption.

Fig. 8. Total energy consumption under different parameters

GHz, the energy consumption of our proposed scheme is 63.8% less than that of **SO**, consuming 1.75 J of energy.

Fig. 8 (c) shows the energy consumption under different block sizes. When the block size increases, energy consumption of the system will increase. Because the increase in the delay for APs to generate blocks during the consensus will consume more energy. In **SO**, **OO**, and **RO**, the total energy consumption increase significantly with the increase in the block size, however, in the proposed scheme, energy consumption increases slowly since the delay-based RAFT mechanism can optimally select APs with more resources as the *Leader*, thereby minimizing the delay and the energy consumption for communication and block generation. Our scheme maintains the lowest energy consumption when the size of the block increase, which is 33.16% less than that of **SO**, consuming 10.4 J of energy when block size is 1000 kbits.

Furthermore, we compare the energy consumption under different ratios of the transmission energy consumption coefficient to the computing energy consumption coefficient, and the transmission energy consumption coefficient is fixed to 1.5 . As shown in Fig. 8 (d), when the energy consumption coefficient ratio $\frac{\epsilon_c}{\epsilon_p}$ increases, the total energy consumption will gradually decrease, and the downward trend will gradually slow down. This is because the energy consumption of block and task processing still accounts for a larger proportion in total then wireless transmission, and the increase in $\frac{\epsilon_c}{\epsilon_p}$ is equivalent to the decrease in ϵ_p. However, the energy consumption in the transmission process also accounts for a considerable proportion of the total energy consumption, so the downward trend will gradually slow down. It is worth noting that our scheme maintains the lowest energy consumption when the ratio increase, which is 27.74% less than that of **SO**, consuming 3.36 J of energy when $\frac{\epsilon_c}{\epsilon_p}$ becomes 5.

Fig. 8 (e) depicts the impact of number of APs on the total energy consumption. When the number of APs increases, the total energy consumption will decrease, which is consistent with the conclusion in Fig. 4 (c). When the number of APs increases, the computing resources and bandwidth allocated to users will be more abundant, but more energy will be consumed in the process of task processing and consensus, and the gain will be gradually offset. When number of APs $M = 12$, the total energy consumption of the proposed scheme is 48.77% less than that of **SO**, consuming 2.09 J of energy.

Finally, we turn our attention to the impact of computing density of tasks on the total energy consumption. When the computing density of tasks increases, the AP needs to allocate more computing resources to meet the offloading requests of users. Therefore, more energy will be consumed in the consensus process, thus increasing the total energy consumption. It can be concluded form Fig.7 and Fig.8 that under the influence of different parameters, our scheme can always maintain the lowest total energy consumption and the smallest fluctuation. When computing density equals to 5000 CPU cycles/bit, the total energy consumption of the proposed scheme is 21.72% less than that of **SO**, consuming 27.23J of energy.

6 Conclusion and Future Work

In this paper, we propose an energy-efficient blockchain-enabled UC-MEC framework. In the proposed system, each user will be provided with wireless transmission and task offloading services by a specific AP cluster. After the task processing is completed, all APs will run the consensus mechanism to record the resource transaction transactions to the blockchain to ensure the security of the network. A resource-aware RAFT consensus mechanism is proposed which considers the computing resources and bandwidth consumed by the data transmission and block duplication. To minimize the total energy consumption, we jointly optimize the clustering and the resource allocation strategy in the task offloading and the consensus process. To solve the problem, we decompose it into M sub-problems based on ADMM so that each AP can solve the sub-problem in parallel. Simulation results show that our proposed scheme has high effectiveness under different parameters and can significantly reduce the total energy consumption when compared with traditional MEC.

As an emerging technology, artificial intelligence (AI), especially machine learning (ML), can be seen as a fundamental solution to realize fully intelligent orchestration and management in edge/cloud computing [50]. For blockchain-enabled mobile edge computing, deep reinforcement learning (DRL) can be used to obtain the optimal strategy of network resource allocation, block size, block interval or other parameters decisions in blockchain. In addition, the task types and location of users will also affect the AP clustering process. Users with similar task types or geographically co-located can be assigned to the similar AP cluster. In the future, we will study the impact of these factors on AP clustering and system performance.

References

[1] Q. Luo, S. Hu, C. Li, G. Li, and W. Shi, "Resource scheduling in edge computing: A survey," *IEEE Commun. Surv. Tutorials*, vol. 23, no. 4, pp. 2131–2165, Aug. 2021.

[2] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Jun. 2016.

[3] S. Chen, F. Qin, B. Hu, X. Li, and Z. Chen, "User-centric ultra-dense networks for 5g: challenges, methodologies, and directions," *IEEE Wireless Commun.*, vol. 23, no. 2, pp. 78–85, May. 2016.

[4] C. Pan, M. Elkashlan, J. Wang, J. Yuan, and L. Hanzo, "User-centric c-ran architecture for ultra-dense 5g networks: Challenges and methodologies," *IEEE Commun. Mag.*, vol. 56, no. 6, pp. 14–20, Jun. 2018.

[5] G. Caso, . Alay, G. C. Ferrante, L. D. Nardis, M.-G. D. Benedetto, and A. Brunstrom, "User-centric radio access technology selection: A survey of game theory models and multi-agent learning algorithms," *IEEE Access*, vol. 9, pp. 84 417–84 464, Jun. 2021.

[6] G. O. Boateng, D. Ayepah-Mensah, D. M. Doe, A. Mohammed, G. Sun, and G. Liu, "Blockchain-enabled resource trading and deep reinforcement learning-based autonomous ran slicing in 5g," *IEEE Trans. Netw. Serv. Manage.*, vol. 19, no. 1, pp. 216–227, Mar. 2022.

[7] I. R. Fedorov, A. V. Pimenov, G. A. Panin, and S. V. Bezzateev, "Blockchain in 5g networks: Perfomance evaluation of private blockchain," in *Wave Electron. Appl. Inf. Telecommun. Syst., WECONF - Conf. Proc.*, May. 2021, pp. 1–4.

[8] T. M. Hewa, A. Kalla, A. Nag, M. E. Ylianttila, and M. Liyanage, "Blockchain for 5g and iot: Opportunities and challenges," in *IEEE Int. Conf. Commun. Netw., ComNet - Proc.*, Mar. 2020, pp. 1–8.

[9] W. Zou, D. Lo, P. S. Kochhar, X.-B. D. Le, X. Xia, Y. Feng, Z. Chen, and B. Xu, "Smart contract development: Challenges and opportunities," *IEEE Trans. Software Eng.*, vol. 47, no. 10, pp. 2084–2106, Sep. 2021.

[10] M. Aloqaily, A. Boukerche, O. Bouachir, F. Khalid, and S. Jangsher, "An energy trade framework using smart contracts: Overview and challenges," *IEEE Network*, vol. 34, no. 4, pp. 119–125, Apr. 2020.

[11] S. Fan, H. Zhang, Y. Zeng, and W. Cai, "Hybrid blockchain-based resource trading system for federated learning in edge computing," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2252–2264, Nov. 2020.

[12] S. Wang, D. Ye, X. Huang, R. Yu, Y. Wang, and Y. Zhang, "Consortium blockchain for secure resource sharing in vehicular edge computing: A contract-based approach," *IEEE Trans. Network Sci. Eng.*, vol. 8, no. 2, pp. 1189–1201, Jun. 2020.

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

[14] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Computation offloading and content caching in wireless blockchain networks with mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11 008–11 021, Aug. 2018.

[15] D. Nguyen, M. Ding, P. Pathirana, A. Seneviratne, J. Li, and V. Poor, "Cooperative task offloading and block mining in blockchain-based edge computing with multi-agent deep reinforcement learning," *IEEE Trans. Mob. Comput.*, pp. 1–1, Sep. 2021.

[16] J. Feng, F. Richard Yu, Q. Pei, X. Chu, J. Du, and L. Zhu, "Cooperative computation offloading and resource allocation for blockchain-enabled mobile-edge computing: A deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6214–6228, Dec. 2019.

[17] S. Hu, Y.-C. Liang, Z. Xiong, and D. Niyato, "Blockchain and artificial intelligence for dynamic resource sharing in 6g and beyond," *IEEE Wireless Commun.*, vol. 28, no. 4, pp. 145–151, Mar. 2021.

[18] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, May. 2016.

[19] T. Zhao, S. Zhou, L. Song, Z. Jiang, X. Guo, and Z. Niu, "Energy-optimal and delay-bounded computation offloading in mobile edge computing with heterogeneous clouds," *China Commun.*, vol. 17, no. 5, pp. 191–210, May. 2020.

[20] P. Zhao, H. Tian, C. Qin, and G. Nie, "Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing," *IEEE Access*, vol. 5, pp. 11 255–11 268, Jun. 2017.

[21] F. Guo, F. R. Yu, H. Zhang, H. Ji, M. Liu, and V. C. M. Leung, "Adaptive resource allocation in future wireless networks with blockchain and mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 1689–1703, Mar. 2020.

[22] P. A. Apostolopoulos, E. E. Tsiropoulou, and S. Papavassiliou, "Risk-aware data offloading in multi-server multi-access edge computing environment," *IEEE/ACM Trans. Networking*, vol. 28, no. 3, pp. 1405–1418, Jun. 2020.

[23] N. Irtija, I. Anagnostopoulos, G. Zervakis, E. E. Tsiropoulou, H. Amrouch, and J. Henkel, "Energy efficient edge computing enabled by satisfaction games and approximate computing," *IEEE Trans. Green Commun. Networking*, vol. 6, no. 1, pp. 281–294, Mar. 2022.

[24] P. K. Bishoyi and S. Misra, "Enabling green mobile-edge computing for 5g-based healthcare applications," *IEEE Trans. Green Commun. Networking*, vol. 5, no. 3, pp. 1623–1631, Sep. 2021.

[25] P. K. K. Bishoyi and S. Misra, "Towards energy-and cost-efficient sustainable mec-assisted healthcare systems," *IEEE Trans. Sustainable Comput.*, pp. 1–1, Apr. 2022.

[26] J. Du, C. Jiang, A. Benslimane, S. Guo, and Y. Ren, "Sdn-based resource allocation in edge and cloud computing systems: An evolutionary stackelberg differential game approach," *IEEE/ACM Trans. Networking*, vol. 30, no. 4, pp. 1613–1628, Aug. 2022.

[27] A. Bozorgchenani, F. Mashhadi, D. Tarchi, and S. A. Salinas Monroy, "Multi-objective computation sharing in energy and delay constrained mobile edge computing environments," *IEEE Trans. Mob. Comput.*, vol. 20, no. 10, pp. 2992–3005, Oct. 2021.

[28] L. Ying, S. Yang, and R. Srikant, "Optimal delaythroughput tradeoffs in mobile ad hoc networks," *IEEE Trans. Inf. Theory*, vol. 54, no. 9, pp. 4119–4143, Oct. 2008.

[29] A. Bozorgchenani, F. Mashhadi, D. Tarchi, and S. Monroy, "Multi-objective computation sharing in energy and delay constrained

mobile edge computing environments," *IEEE Trans. Mob. Comput.*, vol. 20, pp. 2992–3005, Oct. 2021.
[30] C. Zhu and W. Yu, "Stochastic modeling and analysis of user-centric network mimo systems," *IEEE Trans. Commun.*, vol. 66, no. 12, pp. 6176–6189, Aug. 2018.
[31] J.-B. Wang, J. Zhang, C. Ding, H. Zhang, M. Lin, and J. Wang, "Joint optimization of transmission bandwidth allocation and data compression for mobile-edge computing systems," *IEEE Commun. Lett.*, vol. PP, pp. 1–1, May. 2020.
[32] M. Liu and Y. Liu, "Price-based distributed offloading for mobile-edge computing with computation capacity constraints," *IEEE Wireless Commun. Lett.*, vol. 7, no. 3, pp. 420–423, Dec. 2018.
[33] J. Shi, J. Du, Y. Shen, J. Wang, J. Yuan, and Z. Han, "Drl-based v2v computation offloading for blockchain-enabled vehicular networks," *IEEE Trans. Mob. Comput.*, pp. 1–1, Jan. 2022.
[34] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," *USENIX*, pp. 305–320, Jan. 2014.
[35] L. Hou, X. Xu, K. Zheng, and X. Wang, "An intelligent transaction migration scheme for raft-based private blockchain in internet of things applications," *IEEE Commun. Lett.*, vol. 25, no. 8, pp. 2753–2757, May. 2021.
[36] H. Xu, L. Zhang, Y. Liu, and B. Cao, "Raft based wireless blockchain networks in the presence of malicious jamming," *IEEE Wireless Commun. Lett.*, Jan. 2020.
[37] K. Yu, L. Tan, M. Aloqaily, H. Yang, and Y. Jararweh, "Blockchain-enhanced data sharing with traceable and direct revocation in iiot," *IEEE Trans. Ind. Inf.*, vol. 17, no. 11, pp. 7669–7678, Nov. 2021.
[38] S. Zheng, T. Han, Y. Jiang, and X. Ge, "Smart contract-based spectrum sharing transactions for multi-operators wireless communication networks," *IEEE Access*, vol. 8, pp. 88 547–88 557, 2020.
[39] H. Wu, K. Wolter, P. Jiao, Y. Deng, Y. Zhao, and M. Xu, "Eedto: An energy-efficient dynamic task offloading algorithm for blockchain-enabled iot-edge-cloud orchestrated computing," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2163–2176, Feb. 2021.
[40] Y. Hou, Y. Shi, and H. Sherali, "Applied optimization methods for wireless networks," *Applied Optimization Methods for Wireless Networks*, pp. 1–330, Jan. 2010.
[41] Y. Wang, X. Tao, X. Zhang, P. Zhang, and Y. T. Hou, "Cooperative task offloading in three-tier mobile computing networks: An admm framework," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2763–2776, Jan. 2019.
[42] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, pp. 1–122, Jan. 2011.
[43] J. Park and S. Boyd, "General heuristics for nonconvex quadratically constrained quadratic programming," Mar. 2017.
[44] N. Vucic, S. Shi, and M. Schubert, "Dc programming approach for resource allocation in wireless networks," in *WiOpt - Int. Symp. Model. Optim. Mob., Ad Hoc, Wirel. Networks*, May. 2010, pp. 380–386.
[45] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," http://cvxr.com/cvx, Mar. 2014.
[46] R. Chen, H. Lu, and P. Ma, "User-centric cooperative mec service offloading," in *IEEE Wireless Commun. Networking Conf. WCNC*, Mar. 2021, pp. 1–6.
[47] Y. Xu, H. Zhang, H. Ji, L. Yang, X. Li, and V. C. M. Leung, "Transaction throughput optimization for integrated blockchain and mec system in iot," *IEEE Trans. Wireless Commun.*, vol. 21, no. 2, pp. 1022–1036, Aug. 2021.
[48] X. Chen, Y. Cai, Q. Shi, M. Zhao, B. Champagne, and L. Hanzo, "Efficient resource allocation for relay-assisted computation offloading in mobile-edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 2452–2468, Mar. 2020.
[49] T. Bai, C. Pan, Y. Deng, M. Elkashlan, A. Nallanathan, and L. Hanzo, "Latency minimization for intelligent reflecting surface aided mobile edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 11, pp. 2666–2682, Nov. 2020.
[50] J. , C. Jiang, J. Wang, Y. Ren, and M. Debbah, "Machine learning for 6g wireless networks: Carrying forward enhanced bandwidth, massive access, and ultrareliable/low-latency service," *IEEE Veh. Technol. Mag.*, vol. 15, no. 4, pp. 122–134, Dec. 2020.