Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Blockchain-based 6G task offloading and cooperative computing resource allocation study

Shujie Tian[1†], Yuexia Zhang[2*], Yanxian Bi[3*] and Taifu Yuan[4†]

Abstract

In the upcoming era of 6G, the accelerated development of the Internet of Everything and high-speed communication is poised to provide people with an efficient and intelligent life experience. However, the exponential growth in data traffic is expected to pose substantial task processing challenges. Relying solely on the computational resources of individual devices may struggle to meet the demand for low latency. Additionally, the lack of trust between different devices poses a limitation to the development of 6G networks. In response to this issue, this study proposes a blockchain-based 6G task offloading and collaborative computational resource allocation (CERMTOB) algorithm. The proposed first designs a blockchain-based 6G cloud-network-edge collaborative task offloading model. It incorporates a blockchain network on the edge layer to improve trust between terminals and blockchain nodes. Subsequently, the optimization objective is established to minimize the total latency of offloading, computation, and blockchain consensus. The optimal offloading scheme is determined using the wolf fish collaborative search algorithm(WF-CSA) to minimize the total delay. Simulation results show that the WF-CSA algorithm significantly reduces the total delay by up to 42.58% compared to the fish swarm algorithm, wolf pack algorithm and binary particle swarm optimisation algorithm. Furthermore, the introduction of blockchain to the cloud-side-end offloading system improves the communication success rate by a maximum of 14.93% compared to the blockchain-free system.

Keywords Blockchain, Resource allocation, Cloud-side and end-side collaborative offloading, 6G

Introduction

With the ongoing advancements in communication technology and science, the sixth generation (6G) of mobile communication technology has emerged as a forefront technology, poised to interconnect everything in the future [1–3]. 6G networks are anticipated to enable unprecedented mobile broadband speeds, low-latency communications, and massive terminal connectivity. This positions 6G as a catalyst for technological developments in various fields, including autonomous driving, smart cities, virtual reality, and automated industries [4–6]. However, the huge resource requirements associated with handling large-scale networked devices, real-time applications, and high-speed data transfers present significant obstacles to the rational allocation and efficient use of resources. In this case, traditional resource

†Shujie Tian and Taifu Yuan contributed equally to this work.

*Correspondence:
Yuexia Zhang
zhangyuexia@bistu.edu.cn
Yanxian Bi
biyanxian@cetc.com.cn
[1] Key Laboratory of Information and Communication Systems, Ministry of Information Industry, Beijing Information Science & Technology University, Beijing 100101, China
[2] Key Laboratory of Modern Measurement & Control Technology, Ministry of Education, Beijing Information Science & Technology University, Beijing 100101, China
[3] China Academy of Electronic and Information Technology, CETC Academy of Electronics and Information Technology Group Co., Ltd., Beijing 100041, China
[4] Beijing Academy of Blockchain and Edge Computing, Beijing 100086, China

allocation methods may face the challenge of high transmission latency. Thus the urgent need to reduce latency by improving resource utilisation efficiency and enabling flexible offloading in 6G networks has become apparent. Addressing these core issues has become a key research direction in the field of 6G resource allocation [7–10].

Researchers and scholars have conducted extensive research in the field of 6G computing resource allocation. Prathiba et al. [11] proposed a resource management algorithm, aiming to offload and manage resources with low latency. This algorithm uses a stochastic network algorithm to calculate the upper limit latency of heterogeneous communication systems and determine the probability associated with the offloading mechanism. Computational tasks are then offloaded to the optimal network based on the detected probability values. Lin et al. [12] designed a 6G large-scale IoT architecture that facilitates dynamic resource allocation and introduced a resource allocation algorithm based on artificial intelligence. They have additionally devised a dynamic nested neural network aimed at facilitating online adaptation of the learning model structure to effectively address the evolving demands of dynamic resource allocation. Qin et al. [13] introduced a novel 6G resource allocation framework centered around air-heaven-airspace integration. They addressed the intricate issue of triple matching among equipment, content sources, and users within air-heaven-airspace integrated networks through content-centric and client-focused resource allocation techniques. Goudarzi et al. [14] introduced a computational resource allocation model designed to address the joint optimization challenge of queue-based computational offloading and adaptive computational resource allocation. This method prioritizes maintaining task computation latency for all ground mobile node (MNs) over a defined time frame. Simultaneously meeting the task computation constraints, it seeks to maximize the overall reachability of MNs while minimizing energy consumption for both UAVs and MNs. Gong et al. [15] proposed an innovative framework in the field of communication, employing deep reinforcement learning (DRL) to facilitate task offloading decomposition. Meanwhile the task offloading and resource allocation processes are optimized collaboratively through the Isotonic Action Generation Technique (IAGT) and the dynamic update strategy. Qi et al. [16] employs network duals as central controllers in combination with crowdsourcing techniques to incentivize mobile users to adhere to predefined paths for sharing network resources. The creation of these paths is depicted as an optimization problem in user recruitment with cost constraints. Initially, the study focuses on a scenario where only one mobile user offers network resources, presenting a pseudo-polynomial time algorithm. Subsequently, for the more intricate scenario involving multiple mobile users, a solution based on graph partitioning is proposed. Lastly, the study delves into determining the minimum expected budget needed to maximize utility within the ideal model. However, these studies mentioned above do not address the trust issues resulting from massive user access, where the data being transmitted may be at risk of being eavesdropped, leaked, or falsified. Therefore, ensuring secure, trustworthy, and efficient collaboration in 6G resource allocation and computational offloading has become a key issue that needs urgent attention [17, 18].

Blockchain plays a significant role in achieving security and trustworthiness in resource allocation. As a distributed shared ledger, its tamper-proof, open, and transparent, and decentralized features can effectively ensure the establishment of trust, protect the privacy of information, and achieve reliable authentication of devices [19–21]. In the 6G communication environment, the integration of blockchain technology with resource allocation offers important advantages. Blockchain can effectively store the computational tasks triggered by resource allocation as transaction information in the block. This transaction information is used to generate the Merkle root through a hashing algorithm, while the block's Pre_hash is calculated from the Merkle root, timestamps, random numbers, and other information. Blockchain ensures the data integrity of each block by connecting the blocks to form a chain structure through Pre_hash [22, 23]. Meanwhile, blockchain ensures the legitimacy of end devices' identity and trustworthiness through a consensus mechanism, effectively reducing the threat of unauthorized devices. The consensus mechanism verifies and ensures the legitimate identity of end devices through the consistent cognition of nodes in the network, guards against unauthorized devices entering the system, and enhances the overall trustworthiness of the resource allocation process [24–26]. Therefore, realizing the efficient combination of blockchain and 6G resource allocation has become a major hotspot in current research.

Additionally, both domestic and international research on blockchain-based 6G resource allocation has yielded some results. Yao et al. [27] proposed a blockchain-enabled cloud-edge device (BC-CED) offloading scheme for computational collaboration tasks. These scheme addresses the offloading problem through reinforcement learning and incorporates an incentive mechanism to ensure the honesty of the device. Okegbile et al. [28] investigated collaborative data sharing schemes aimed at facilitating collaboration among multiple data providers and users through the integration of blockchain and cloud-edge computing. The results showed that system performance analysis contributes to the effective deployment of a data sharing system. Li et al. [29] constructed a cloud-side-end collaborative resource allocation framework, addressing the system energy consumption and delay minimization problem. They obtained the optimal policy using collective

reinforcement learning (CRL) by co-optimizing the offloading policies, group intervals, and transmission power. Feng et al. [30] introduced a blockchain-enabled mobile edge computing resource allocation framework aimed at enhancing computation rates and boosting transaction throughput. This framework achieves its goals by concurrently optimizing offloading policies, power allocation, block size, and block interval. Jain et al [31] introduced a novel approach for resource allocation in IoE environments and 6G networks utilizing blockchain technology. They devised a quasi-opposite search and rescue optimization (QO-SRO) algorithm aimed at enhancing the efficiency of resource allocation processes. Although the above literature has yielded promising results in the research area, it has not considered the problem of collaborative offloading of computational tasks to other base stations for processing, nor has it considered the problem of communication interference in practical scenarios, as highlighted in Table 1.

To address these key issues, this study not only extends the offloading range of computational tasks to achieve collaborative task processing among multiple base stations, but also considers the communication interference problem. Through these comprehensive considerations, the overall resource utilisation of the system is improved, and the authenticity and reliability of the system evaluation is enhanced in a way that is closer to the actual communication environment.

The primary contributions of this study are outlined as follows:

1. Designing a blockchain-based collaborative task offloading model for 6G cloud network edges and ends, comprising multiple edge computing servers, communication base stations, user terminals, and a cloud server. Simultaneously, each communication base station and MEC server is regarded as a blockchain node, forming a blockchain system. A blockchain network layer is added to the cloud-side-end cooperative offloading, enabling the offloading of computational tasks to other base stations with available computational resources for collaborative computation. The model also selects the most trustworthy D nodes for consensus by calculating the trust value of end devices to blockchain nodes. This approach ensures sufficient computational resources, reduces task processing time, and greatly enhances the security and trustworthiness of the resource allocation process.
2. Proposing a blockchain-based 6G task offloading and collaborative computational resource allocation algorithm (CERMTOB, Cloud-Edge Resource Management and Task Offloading in Blockchain Networks). The algorithm addresses the minimization delay problem, which involves the total task offloading delay, the total computational task processing delay, and the blockchain network layer computational delay. To solve this issue, WF-CSA(wolf fish collaborative search algorithm) is proposed. WF-CSA divides the fish into head fish, explorer fish, and fierce fish, assigning the seven behaviors of wolf pack and fish pack according to their characteristics. This methodology expedites the convergence rate of the algorithm, successfully accomplishing the goal of minimizing time delay.
3. The simulation results demonstrate that the WF-CSA algorithm achieves a notable reduction in total delay compared to AFSA,WPA and BPSO, with improvements of up to 42.58%, 28.58% and 15.93%, respectively, across varying task sizes. Additionally, WF-CSA demonstrates superior performance with varying numbers of users and computing power of MEC servers. Meanwhile, the cloud-side end offloading system integrated with blockchain improves the communication success rate by up to 14.93% compared to the cloud-side end offloading system without blockchain.

System model

Network model

The blockchain-based 6G cloud network edge-end collaborative task offloading model proposed in this study is shown in Fig. 1. The system model comprises a cloud server layer, a blockchain network layer, an edge layer, and a user terminal layer. The cloud server layer includes a cloud server *CS*.The edge server layer contains M base stations, denoted

Table 1 Differences between this paper and previous studies

Reference	Cloud Edge offloading	Blockchain	Communication interference	Multibase station cooperative computing	Algorithm
[12]	✓	×	×	×	MDP RL
[14]	✓	×	✓	×	Queue-based offloading
[27]	✓	✓	×	×	RL
[29]	✓	✓	×	×	CRL
[30]	×	✓	×	✓	A3C
Our	✓	✓	✓	✓	WF-CSA

by the set BS, $BS = \{BS_1, BS_2, \cdots, BS_m, \cdots, BS_M\}$. Each base station is equipped with an MEC server, and the corresponding MEC server for base station BS_m is denoted as MEC_m. The set M MEC servers is denoted as MEC, $MEC = \{MEC_1, MEC_2, \cdots, MEC_m, \cdots, MEC_M\}$. In the coverage area of each base station, there are N user terminals, and the user terminals covered by base station BS_m are denoted as $UE^m = \{UE_1^m, UE_2^m, \cdots, UE_n^m, \cdots, UE_N^m\}$. Therefore, the user terminal layer comprises a total of $M \times N$ user terminals.

In the blockchain network layer, each base station and its MEC server are treated as distinct blockchain nodes, containing a total of M blockchain nodes. Let $V_{n\to m}^{trust}$ represent the confidence value of user terminal n to blockchain node m. The consensus mechanism selects the most trustworthy $D(D < M)$ blockchain nodes from the total blockchain nodes, while non-selected nodes are only responsible for receiving data and bookkeeping. This design aims to mitigate the threat of unauthorized devices through consensus algorithms, thereby improving the security and trustworthiness of resource allocation and computation offloading. Details of the symbols used in the paper are shown in Table 2 for reference.

Offloading model

In this study, it is assumed that a fine-grained computation task T_n^m is generated at a given time on a user terminal UE_n^m situated in the coverage area of base station BS_m. This task can be partitioned into multiple subtasks.T_n^m can be expressed as a ternary $T_n^m =< d_n^m, s_n^m, \tau_n^m >$,$d_n^m$ denotes the total input data size for computational task T_n^m, unit is expressed in bits;s_n^m represents the computing power required to complete a unit of task data, expressed in cycles/bit;τ_n^m indicates the deadline for task completion in s.

Due to the limited local computing power of the user terminal, completing the computing task within the deadline τ_n^m is not feasible. Therefore, the task needs to be offloaded to the MEC server and cloud server for processing [32]. Simultaneously, if the MEC server is overwhelmed with computational tasks and other base stations have ample computational resources, tasks can be offloaded to those base stations for assistance. In this study, the partial offloading approach aims to optimize the utilization of computing resources at the user terminal layer, edge layer, and cloud server layer.The proportions of sub-tasks for local terminal computing, offloading to MEC server, and cloud server computing, respectively, are expressed by $\alpha_n^m, \beta_n^{m,1}, \beta_n^{m,2}, \cdots, \beta_n^{m,m}, \cdots, \beta_n^{m,M}, \gamma_n^m$,and $\alpha_n^m + \beta_n^{m,1} + \beta_n^{m,2} + \cdots + \beta_n^{m,m} + \cdots + \beta_n^{m,M} + \gamma_n^m = 1$.

Wireless communication links are used between user terminals and base stations, while fiber optics facilitate wired communication between base stations and also between base stations and cloud servers. In this study, Orthogonal Frequency Division Multiple Access

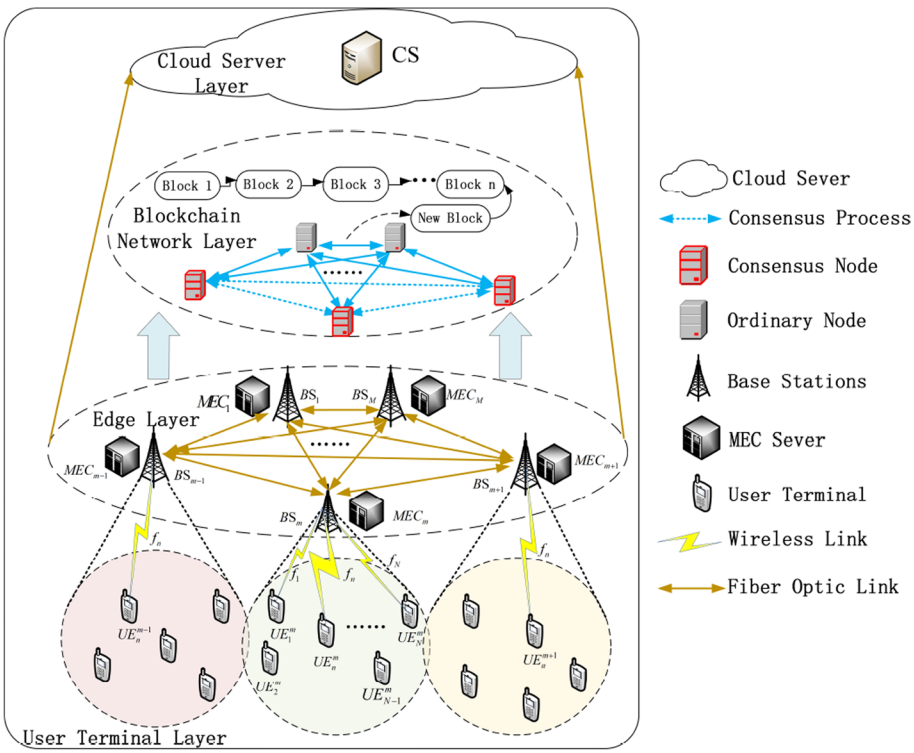

Fig. 1 The blockchain-based 6G cloud network edge-end collaborative task offloading model

Table 2 Summary of notations

Symbols	Description
M	Number of base stations
N	Number of users in each base station
D	Number of consensus nodes
T_n^m	Computational tasks generated by user n in base station m
d_n^m	Total input data size for task T_n^m
s_n^m	Computing power required to complete a unit of task data
τ_n^m	Deadline for task completion
$\alpha_n^m, \beta_n^{m,M}, \gamma_n^m$	Local offload ratio, Mth MEC server offload ratio, cloud server offload ratio
f	Sub-channel bandwidth for users
$g_{n,m}^{f_n}$	Channel gain of user terminal n and MEC server m using subchannel f_n
σ^2	Background noise variance
$f_{n,m}^{local}, f_{n,m}^{MEC}, f_{n,m}^{cloud}$	CPU computation frequency for local, MEC, and cloud tasks
φ, ϕ	CPU cycles required for signature and MAC generation/verification
ϑ	Maximum transaction capacity within a block
θ	weight of correct transactions
f_d^{bt}	CPU cycle frequency of consensus node d

(OFDMA) technique is used for uplink offloading of user terminals within the same base station. The communication bandwidth B of each base station is divided into N mutually orthogonal wireless communication subchannels, each with a bandwidth $f = \frac{B}{N}$,and the set of communication subchannels is defined as $f = \{f_1, f_2, \cdots, f_n, \cdots, f_N\}$.Each subchannel is assigned to a user terminal for communication, ensuring that users within the range of the same base station experience no interference with each other [33]. Assuming that user terminals within the range of different base stations can multiplex the same wireless communication subchannel, $UE_n^1, UE_n^2, \cdots, UE_n^m, \cdots, UE_n^M$ denotes that user terminals n within M base stations multiplex the subchannel segment f_n.Therefore,UE_n^m will experience interference from user terminals from other base stations, denoted by

$$I_{n,m}^{f_n} = \sum\nolimits_{i \in UE^m \setminus \{UE_n^m\}} \sum\nolimits_{j \in BS \setminus \{BS_m\}} a_{ij}^{f_n} p_i g_{i,j}^{f_n} \tag{1}$$

where $a_{ij}^{f_n} \in \{0, 1\}$,assuming that a terminal can only offload the task through one sub-channel, then the computation task T_n^m generated by user terminal i is offloaded to the MEC server j for computation using subchannel f_n when $a_{ij}^{f_n} = 1$,otherwise.$g_{i,j}^{f_n}$ indicates the channel gain between user terminal i and MEC server j,which is calculated from $g_{i,j}^{f_n} = \xi_{i,j}(t) h_0 (d_0/d_{i,j})^{\theta}$,$\xi_{n,m}$ is the Rayleigh fading between user terminal i and MEC server j,h_0 denotes the path loss constant,d_0 is the reference distance, and $d_{i,j}$ represents the distance between user terminal i and MEC server j [34].

Similarly, the signal-to-interference-plus-noise ratio for the offloading task of terminal UE_n^m is obtained as

$$SINR_{n,m} = \frac{p_n^m g_{n.m}^{f_n}}{I_{n,m}^{f_n} + \sigma^2} \tag{2}$$

where σ^2 is the background noise variance. Using Shannon's formula, the transmission rate when terminal UE_n^m offloads the task can be obtained as

$$\begin{aligned} R_{n,m,f_n} &= W \log_2 (1 + SINR_{n,m}) \\ &= W \log_2 \left(1 + \frac{p_n^m g_{n.m}^{f_n}}{I_{n,m}^{f_n} + \sigma^2}\right) \end{aligned} \tag{3}$$

and the communication latency of the offloading task from terminal UE_n^m to MEC server is

$$T_{n,m,f_n} = \frac{(1 - \alpha_n^m) d_n^m}{R_{n,m,f_n}} \tag{4}$$

The total communication latency of the terminal offload task to MEC server is

$$T_{mec}^{trans} = \sum_{m=1}^{M} \sum_{n=1}^{N} \frac{(1 - \alpha_n^m) d_n^m}{R_{n,m,f_n}} \tag{5}$$

In this study, it is assumed that the transmission delay incurred between base stations utilizing fiber optic communication is negligible. The fixed transmission rate for fiber link communication between the MEC server and the cloud server can be denoted by $R_{m,c}$, and the loss in fiber optic transmission is ignored. Therefore, the transmission delay of the task offloading from base station BS_m to the cloud server is

$$T_{m,c} = \frac{\gamma_n^m d_n^m}{R_{m,c}} \tag{6}$$

The total communication latency of the base station offloading tasks to the cloud server is calculated as

$$T_{cloud}^{trans} = \sum_{m=1}^{M} \sum_{n=1}^{N} \frac{\gamma_n^m d_n^m}{R_{m,c}} \tag{7}$$

The total communication latency for task offloading is calculated as

$$T^{trans} = T_{mec}^{trans} + T_{cloud}^{trans} \tag{8}$$

Computation model

Due to the different locations where each molecular task is processed, the computation modes can be categorized into three types: local computation mode, which is processed at local terminals; MEC server computation mode, which is processed on MEC servers; and cloud server computation mode, which is processed on cloud servers. The three computational modes consume computational resources from different endpoints and can, therefore, process tasks in parallel.

Local computation

Assuming that the CPU computation frequency of the nth user terminal within the range of base station BS_m is $f_{n,m}^{local}$, according to the computational mandate T_n^m and the local offloading ratio α_n^m, the delay $T_{n,m}^{local}$ required for local computation can be obtained as

$$T_{n,m}^{local} = \frac{\alpha_n^m d_n^m s_n^m}{f_{n,m}^{local}} \tag{9}$$

This, in turn, gives the total delay of local computation as

$$T_{local}^{exe} = \sum_{m=1}^{M} \sum_{n=1}^{N} \frac{\alpha_n^m d_n^m s_n^m}{f_{n,m}^{local}} \tag{10}$$

MEC server computing

Assuming that the CPU computing frequency assigned by the MEC server MEC_m to the computation task T_n^m subtask is $f_{n,m}^{MEC}$, and according to the computational mandate T_n^m and the MEC offloading ratio $\beta_n^{m,1}, \beta_n^{m,2}, \cdots, \beta_n^{m,m}, \cdots, \beta_n^{m,M}$, the delay $T_{n,m}^{MEC}$ required for the computation by the MEC server can be obtained as

$$T_{n,m}^{MEC} = \sum_{i=1}^{M} \frac{\beta_n^{m,i} d_n^m s_n^m}{f_{n,m}^{MEC}} \tag{11}$$

This, in turn, gives the total delay calculated by the MEC server as

$$T_{mec}^{exe} = \sum_{m=1}^{M} \sum_{n=1}^{N} \sum_{i=1}^{M} \frac{\beta_n^{m,i} d_n^m s_n^m}{f_{n,m}^{MEC}} \tag{12}$$

Cloud server computing

Assuming that the CPU computation frequency assigned by the cloud server *CS* to the computation task T_n^m subtask is $f_{n,m}^{cloud}$, and based on the computation task T_n^m and the cloud server offload ratio γ_n^m, the delay $T_{n,m}^{cloud}$ required by the cloud server for computation can be obtained as

$$T_{n,m}^{cloud} = \frac{\gamma_n^m d_n^m s_n^m}{f_{n,m}^{cloud}} \tag{13}$$

This, in turn, gives the total delay of the cloud server computation as

$$T_{cloud}^{exe} = \sum_{m=1}^{M} \sum_{n=1}^{N} \frac{\gamma_n^m d_n^m s_n^m}{f_{n,m}^{cloud}} \tag{14}$$

In summary, the adoption of partial offloading facilitates concurrent processing of computational mandates, thereby reducing the overall processing latency, denoted as T^{exe} of the computational tasks generated by the user terminal

$$T^{exe} = \max\left\{T_{local}^{exe}, T_{mec}^{exe}, T_{cloud}^{exe}\right\} \tag{15}$$

Blockchain model

Since blockchain nodes with low trust values may experience packet loss during offloading, each end device should select nodes with higher trust values for communication to enhance the safety and stability of the communication. The confidence values of M blockchain nodes are calculated using the trust value calculation method in "Blockchain MEC system trust value Computation" section and arranged in descending order of trust value. The first D nodes with the highest trust values among them can be selected as consensus nodes.

Assuming that the consensus node of the blockchain employs practical byzantine fault tolerance (PBFT) as the consensus mechanism, the implementation steps are as follows [35]:

In the first step, the blockchain node collects the transaction recorded by the computational task offload from the edge layer. Upon receiving these transactions, the master node undertakes the verification process for both the signature and the message authentication code (MAC). Assuming equal CPU cycles are required for generating and verifying the signature and MAC, denoted as φ and ϕ, respectively, the computational cost of the master node is

$$h_1 = \frac{\vartheta}{\theta}(\varphi + \phi) \tag{16}$$

where ϑ represents the maximum capacity of transactions that can be included within a block, and θ represents the weight of correct transactions.

In the second step, the master node dispatches a pre-prepare message to all sub-nodes. When the sub-node obtains a new block, it first verifies the signature and MAC of the block. Subsequently, it verifies the signature and MAC of the transaction. The computational cost of the sub-node during the pre-prepare process is

$$h_2 = (\vartheta + 1)(\varphi + \phi) \tag{17}$$

In the third step, each sub-node dispatches a prepare message to the rest of the sub-nodes. This node has to verify $2f(f = (D-1)/3)$ signatures and MACs sent from the other sub-nodes, and the sub-node has to generate 1 signature and $D-1$ MACs. The computational overhead incurred by the sub-nodes during the commit phase is

$$h_3 = \varphi + (D-1)\phi + 2f(\varphi + \phi) \tag{18}$$

In the fourth step, each sub-node sends a commit message to the rest of the sub-nodes, and the sub-nodes also need to verify $2f$ signature and MAC upon obtaining the commit message. The computational overhead of the sub-nodes in the commit phase is

$$h_4 = \varphi + (D-1)\phi + 2f(\varphi + \phi) \tag{19}$$

In the fifth step, the new block becomes a valid block after receiving $2f$ matching commit message and subsequently broadcasts it to the blockchain network layer, where the computational overhead of the sub-node is

$$h_5 = \vartheta(\varphi + \phi) \tag{20}$$

Therefore, the total consensus delay at the blockchain network layer is

$$T^{bt} = \max\left\{\frac{H_{bt}}{f_d^{bt}}\right\} \tag{21}$$

where $H_{bt} = h_1 + h_2 + h_3 + h_4 + h_5$ is the total computational overhead of the blockchain consensus process, and f_d^{bt} is the CPU cycle frequency of the dth consensus node.

Blockchain MEC system trust value computation

In this study, a comprehensive estimation method is used to determine the trustworthiness of consensus nodes, which includes both direct and indirect confidence factors. The direct confidence assessment is carried out using subjective logic, while the determination of indirect confidence involves soliciting opinions from third-party sources. It is assumed that the node confidence value is estimated according to a real number from 0 to 1. Similar to many related literatures, the critical value of trust is set to 0.5, and a node is considered credible when the confidence value is greater than 0.5; otherwise, it is not considered credible. The confidence value of a consensus node is calculated as described below [30].

Computing of direct confidence value

The computation of the direct confidence value is based on node honesty (NH) and node capacity. Therefore, a subjective logical framework must be employed to address the uncertainty in the task offloading process due to the inherent volatility and noise issues in the communication channel between the end device and the consensus node. Assume that the trust value of a terminal device UE_n^m to a communication base station BS_m is represented by the triad $\omega_{n\to m}^m = \{b_{n\to m}^m, d_{n\to m}^m, v_{n\to m}^m\}$, where $b_{n\to m}^m$,$d_{n\to m}^m$,$v_{n\to m}^m$denote trust, distrust, and uncertainty in turn, and that the relationship among them is

$$\begin{aligned} &b_{n\to m}^m, d_{n\to m}^m, v_{n\to m}^m \in [0,1] \\ &b_{n\to m}^m + d_{n\to m}^m + v_{n\to m}^m = 1 \end{aligned} \tag{22}$$

According to the confidence model in the literature, the node honesty NH can be obtained using the following equation

$$NH_{n\to m}^m = b_{n\to m}^m + \mu v_{n\to m}^m \tag{23}$$

where $0 \le \mu \le 1$ is a constant representing the magnitude of the impact of confidence uncertainty, and

$$\begin{aligned} b_{n\to m}^m &= (1 - v_{n\to m}^m)\frac{\alpha_{n\to m}^m}{\alpha_{n\to m}^m + \beta_{n\to m}^m} \\ d_{n\to m}^m &= (1 - v_{n\to m}^m)\frac{\beta_{n\to m}^m}{\alpha_{n\to m}^m + \beta_{n\to m}^m} \\ v_{n\to m}^m &= 1 - s_{n\to m}^m \end{aligned} \tag{24}$$

where $\alpha_{n\to m}^m$ and $\beta_{n\to m}^m$ represent the number of completed and uncompleted communications, respectively, and $s_{n\to m}^m$ represents the quality of the communication channel, indicating the chance of successful grouping.

$$\begin{aligned} \overset{new}{\alpha}{}_{n\to m}^m &= \alpha_{n\to m}^m + P_{n\to m}^m \times (\alpha_{n\to m}^m + \beta_{n\to m}^m) \\ \overset{new}{\beta}{}_{n\to m}^m &= \beta_{n\to m}^m - P_{n\to m}^m \times (\alpha_{n\to m}^m + \beta_{n\to m}^m) \end{aligned} \tag{25}$$

where $P_{n\to m}^m$ denotes the chance of dropping the packet, which can be determined using the following equation:

$$P_{n\to m}^m = 1 - \frac{\sum_b^c \omega_m(b) \times \omega_m(b)}{\sum_b^c \omega_m(b)} \tag{26}$$

where $\omega_m(b)$ denotes the weighting factor assigned to the historical link state of base station BS_m, such that $link = \omega_m(1), \omega_m(2), \cdots, \omega_m(b)$ represents the state of the historical link, and the weight value is derived from $\omega_m(b) = (2b_m/c_m(c_m+1))$, where b_m and c_m represent the sequence number of $\omega_m(b)$ in the link and the link state sequence number.

Additionally, this study assumes that all base stations share identical initial energy loss rate and energy criterion,

and malicious nodes usually lose energy abnormally when they attack. Therefore, this study uses energy as a measure of QoS confidence value as a way of determining whether a communication base station is malicious or not. Assuming that $Q^m_{n\to m}$ uses the ray projection method to represent the energy loss rate ($Q^m_{n\to m} \in [0,1]$),the node capacity (NC) is therefore given by the following equation:

$$NC^m_{n\to m} = \begin{cases} 1 - Q^m_{n\to m}, & \text{if } E^m_{n\to m} \geq \psi \\ 0, & \text{otherwise} \end{cases} \tag{27}$$

where $E^m_{n\to m}$ and ψ denote the remaining energy and energy threshold of a blockchain node, respectively.

To summarize, node confidence depends on NH and NC, and the direct confidence value of a blockchain node can be denoted as

$$V^{\text{direct}}_{n\to m} = \begin{cases} 0.5 + (NH^m_{n\to m} - 0.5) \times NC^m_{n\to m}, & \text{if } NH^m_{n\to m} \geq 0.5 \\ NH^m_{n\to m} \times NC^m_{n\to m}, & \text{otherwise} \end{cases} \tag{28}$$

Computation of indirect confidence value

Recommendations from third-party blockchain network layers also need to be considered to derive trust values. In this study, we assume that the reserve blockchain nodes agree to dedicate their resources to assist the end devices in offloading and computing tasks. When the end devices want to offload utilizing blockchain nodes, the reserve nodes around them will apply to the blockchain network layer for the opportunity to assist in the offloading task. Once the application is received, the blockchain network layer selects a appropriate reserve node by evaluating the suggested values associated with each potential reserve node . In this study, we assume that the blockchain updates and saves the recommendations of the reserve nodes in a timely manner. However, not every updated recommendation is trustworthy. If the selection is done based on the recommendations updated by the preparatory nodes at a time, untrustworthy preparatory nodes may be selected, leading to unreliable trust estimates. Therefore, it is necessary to verify whether the recommendation is trustworthy or not.

In this study, we present a simple approach that relies on the recommended reliability $R^{rel}_{n\to m}$ to evaluate the recommended values. The method calculates the average value R^{ave}_m of all update recommendations for the reserve node m and determines the disparity of this average and the specific recommendation. The greater the disparity, the less reliable the recommendation is. Eventually, the reliability $R^{rel}_{n\to m}$ of the recommendation can be expressed as:

$$R^{rel}_{n\to m} = 1 - \left| R^{rec,i}_{n\to m} - R^{ave}_m \right| \tag{29}$$

where $R^{rec,i}_{n\to m}$ is the suggested value for the ith update within the blockchain network layer.

If the recommender's recommendation credibility falls below 0.5, regardless of the magnitude of the recommendation value, we still cannot conclude that the recommendation is trustworthy based on this alone, and the final recommendation trust value should consider the credibility of the recommendation and the recommendation value:

$$R^{recom}_{n\to m} = \frac{\sum_{i=1}^{I} R^{rel}_{n\to m} \times R^{rec,i}_{n\to m}}{I} \tag{30}$$

where I represent the number of times the recommended value is updated. Subsequently, the confidence value of the blockchain node can then be derived as:

$$V^{\text{trust}}_{n\to m} = \begin{cases} V^{\text{direct}}_{n\to m}, & \text{if } \alpha^{\text{new}}_{n\to m} \geq \text{Th}_{\text{num}} \\ \omega_{\text{direct}} V^{\text{direct}}_{n\to m} + \omega_{\text{recom}} R^{\text{recom}}_{n\to m}, & \text{otherwise} \end{cases} \tag{31}$$

where ω_{direct} denotes the weight of the direct value,ω_{recom} denotes the weight of the recommended value, Th_{num} denotes the count of interactions occurring from the recommender to the blockchain network layer,$\omega_{direct} \in [0,1], \omega_{recom} \in [0,1], \omega_{direct} + \omega_{recom} = 1$.

Issue modeling

In this study, we raise an optimization problem whose objective is to minimize the latency by considering the total task offloading latency, the total computational task processing latency, and the blockchain network layer computational latency. The problem function can be represented as

$$\begin{aligned} F &= T^{trans} + T^{exe} + T^{bt} \\ &= \sum_{m=1}^{M}\sum_{n=1}^{N} \frac{(1-\alpha^m_n) d^m_n}{R_{n,m,f_n}} + \sum_{m=1}^{M}\sum_{n=1}^{N} \frac{\gamma^m_n d^m_n}{R_{m,c}} + \\ &\max\left\{ \sum_{m=1}^{M}\sum_{n=1}^{N} \frac{\alpha^m_n d^m_n s^m_n}{f^{local}_{n,m}}, \sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{i=1}^{M} \frac{\beta^{m,i}_n d^m_n s^m_n}{f^{MEC}_{n,m}}, \right. \\ &\left. \sum_{m=1}^{M}\sum_{n=1}^{N} \frac{\gamma^m_n d^m_n s^m_n}{f^{cloud}_{n,m}} \right\} \\ &+ \max\left\{ \frac{(h_1 + h_2 + h_3 + h_4 + h_5)}{f^{bt}_d} \right\} \end{aligned} \tag{32}$$

In turn, the proposed minimization delay function is obtained as

$$\begin{aligned} &\text{Minimize} \quad F \\ &\text{s.t.} \quad \text{C1: } \alpha^m_n + \beta^{m,1}_n + \cdots + \beta^{m,i}_n + \cdots + \beta^{m,M}_n + \gamma^m_n = 1 \\ &\qquad \text{C2: } \alpha^m_n \in [0,1], \beta^{m,i}_n \in [0,1], \gamma^m_n \in [0,1] \\ &\qquad \text{C3: } T^{trans} + T^{exe} + T^{bt} \leq \tau^m_n \end{aligned} \tag{33}$$

where C1 represents that the proportions of subtasks for local terminal computation, offloading to the MEC server,

and cloud server computation sum up to 1; C2 ensures that the proportions of subtasks for local computation, offloading to the MEC server, and cloud server computation are within the range of 0 to 1; and C3 ensures that the total latency for local, MEC server, and cloud server computation, offloading, and consensus must be less than the task completion cut-off time.

Solving based on WF-CSA optimization algorithm

An individual artificial fish swarm algorithm exhibits some randomness during the solution process, resulting in slow convergence and susceptibility to local optimum solution. To address this issue, this study introduces the wolf pack algorithm into the artificial fish swarm algorithm to enhance its global search capability. The algorithm achieves faster convergence and improved solutions through collaboration and information exchange among the wolves. In the wolf pack algorithm, problem-solving is based on the hunting skills of wolves, with each artificial wolf representing a feasible solution and the prey odor concentration corresponding to the fitness value of the objective function. The pack comprises three different roles: head wolf, scout wolf, and fierce wolf . Similarly, the artificial fish are divided into head fish, scout fish, and fierce fish.

The fish swarm algorithm incorporates four behaviors: foraging, aggregation, tail chasing, and random [36], while the wolf pack algorithm includes three behaviors: wandering, summoning, and siege [37]. Despite differences, there are similarities between these behaviors. In this study, the head fish is designated to perform aggregation and tail chasing behaviors, the explorer fish to perform foraging, wandering, and random behaviors, and the fierce fish to perform summoning and siege behaviors. This strategic assignment of behaviors aims to enhance the algorithm's exploration of the solution space more efficiently.

Headfish behavior

Aggregation behavior

Search for other neighboring fish $x_j(d_{ij} < Visual)$ within the field of view of the headfish,x_i where the total number of neighboring fish is n_f.If $n_f > 0$,calculate the center position x_c and the corresponding fitness value y_c within the field of view of the fierce fish. If $y_c/n_f > \delta y_i$(where δ is the crowding factor), the headfish x_i swims one step toward the center position of the neighboring fish

$$x_c = \sum_{j=1}^{n_f} x_j/n_f \tag{34}$$

$$x_{next} = x_i + \frac{x_c - x_i}{\|x_c - x_i\|} \cdot Step \cdot Rand() \tag{35}$$

If the step size is a static constant, increasing the Step is beneficial to promote fish convergence, but too large a Step will slow down the iteration speed. Compared to a static constant moving step, this study uses a dynamically updated moving step for artificial fish to accelerate convergence while improving the accuracy and stability of the algorithm. To prevent the step size from decreasing to 0, rendering the algorithm's inability to find a more optimal solution, the minimum value of the moving step size is set to τ. Simultaneously, the number of fitness changes V is introduced with an initial value of 0, and the value of V increases by 1 every time an artificial fish updates its optimal fitness.

When the algorithm initially starts running, the move step size *Step* is set to the default size, and a threshold N_2 for dynamic move step splitting is defined. When the number of iterations of the algorithm reaches a threshold, in the subsequent $T_1 \cdot N_2(T_1 > 1)$ iterations, the fish use *Step* given by the following equation

$$Step = \begin{cases} Step \div \rho & , N_2 \le V \le N_2 \cdot T_2 \\ Step \times \rho & , V > N_2 \cdot T_2 \end{cases} \tag{36}$$

where T_2 is a constant, and $1 < T_2 < T_1$,ρ are step factors and $\rho \in (0, 1)$. When the value of *Step* after updating is less than the minimum value τ, make the value of *Step* at this point τ, until the number of iterations reaches $T_1 \cdot N_2$ then return V to zero.

Tail chasing behavior

Search for other neighboring fish $x_j(d_{ij} < Visual)$ within the field of view of the headfish, x_i and the total number of neighboring fish is n_f. If $n_f > 0$, search for the nearby artificial fish $x_{\max}$ with the largest fitness value $y_{\max}$. If $y_{\max}/n_{\max} > \delta y_i$, the headfish x_i swims one step toward the artificial fish $x_{\max}$.

$$x_{next} = x_i + \frac{x_{next} - x_i}{\|x_{next} - x_i\|} \cdot Step \cdot Rand() \tag{37}$$

Scout fish behavior

Foraging behavior

The scout fish use visualization to sense the concentration of food and thus determine the direction of approach. Assuming that the location and fitness value of the ith scout fish are x_i and y_i respectively, and a different location in the scout fishs' field of view and its fitness value are $x_j$and y_j, respectively, if $y_j > y_i$, swim one step in the direction of the new location; otherwise, look for a different location again to determine whether the

condition can be satisfied, and if it is not yet satisfied. if it is not yet satisfied, move one step randomly. The scout fish x_i randomly determines a location x_j in its field of view, as given in the following equation:

$$x_{next} = x_i + \frac{x_j - x_i}{\left\| x_j - x_i \right\|} \cdot Step \cdot Rand() \tag{38}$$

$$x_{next} = x_i + Visual \cdot Rand() \tag{39}$$

Wandering behavior

x_i is the location of the scout fish i in dimension d. The scout fish advances in h directions with a wandering step $step_a$ and records the food concentration after advancing in each direction. Subsequently, it returns to its original position. Therefore, the position of the scout fish after moving one step in the pth ($p = 1, 2, \cdots, h$) orientation is

$$x_{next} = x_i + \sin\left(2\pi \frac{p}{h}\right) \times step_a \tag{40}$$

Random behavior

The scout fish typically swim irregularly through the water, aiming to expand their range of motion to search for food and make companions more efficiently.

$$x_{next} = x_i + Visual \cdot Rand() \tag{41}$$

Fierce fish behavior

Summoning behavior

As the artificial fish with the largest fitness value, the headfish will call out to surrounding fierce fish to take a raiding step $step_b$ to swim rapidly to its position. If the concentration of food found by the fierce fish is greater than the concentration of food perceived by the headfish, the headfish will update and re-summon the fierce fish, otherwise, the fierce fish will continue to remain on the run, i.e.

$$x_{next} = x_i + step_b \times \frac{g - x_i}{\left| g - x_i \right|} \tag{42}$$

where x_i represents the present location of the ith fierce fish, and g represents the present location of the head wolf.

Siege behavior

The fierce fish closer to the headfish will hunt for food in concert with the scout fish. If the artificial fish perceives a food concentration greater than that at the previous location, the scout fish will hunt for food at a hunting step $step_c$. Assuming that the location of the food is G and λ is a random number between [-1,1], the behavior of the fish siege is denoted as

$$x_{next} = x_i + \lambda \times step_c \times |G - x_i| \tag{43}$$

The relationship between the wandering step length $step_a$, raiding step length $step_b$, and hunting step length $step_c$ is shown in the following equation:

$$step_a = \frac{step_b}{2} = 2step_c = \frac{|\max_d - \min_d|}{S} \tag{44}$$

where S is the step factor, and $[\max_d - \min_d]$ is the range of values of the variable to be solved in the dth dimension.

Algorithm parameter improvement and process

Survival mechanism for the strong fish

Since the strong artificial fish will be allocated food preferentially, they are more likely to have a chance of survival compared to the weak artificial fish. Therefore, in the optimization algorithm, the R artificial fish with the lowest fitness value are eliminated in each round, and then R new artificial fish are generated to replenish them.

Dynamic field of view range Visual

In this study, it is presumed that the artificial fish's field of view is continuous, enabling the expansion of the optimization search interval in the early stage of the algorithm when the field of view is large. However, due to the increase of local optimal points in the later stages of the algorithm, too large a value of *Visual* may pose an obstacle to the convergence of the algorithm. Therefore, the value of *Visual* should be reduced appropriately. This adjustment aims to find the optimal fitness value through fewer iterations. The expression for *Visual* is as follows:

$$Visual = \begin{cases} \varepsilon \times Visual & , N_j \geq N_1 \\ Visual & , 1 < N_j < N_1 \end{cases} \tag{45}$$

where N_j represents the present number of iterations of the algorithm, N_1 is the threshold for dynamic segmentation of the field of view range, ε denotes the attenuation coefficient, and$\varepsilon \in (0, 1)$.

Crowding factor

The crowding factor δ represents the degree of crowding that the fish population can accommodate; when δ is large, it increases the search range of artificial fish, while when δ is small, it helps in localized range search. Therefore,

the spectrum of values attributed to δ significantly influences the algorithm's convergence. In this study, a nonlinear dynamic adjustment strategy of inertia weights is employed so that the crowding degree factor can improve the global optimization seeking ability of the fish population during the dynamic transition from large to small. The representation of the crowding degree factor is as follows:

$$\delta = 2z \times r - z \tag{46}$$

where $z = 2 - 2\left(\frac{2N_j}{N_{\max}} - \frac{N_j}{N_{\max}}^2\right)$ denotes the control parameter,N_j and $N_{\max}$ represent the present iteration count and the maximum allowable iterations, respectively. As the number of iterations increase, it gradually decreases from 2 to 0, denoting a random number between [0,1]. Algorithm 1 summarizes the wolf fish school cooperative search algorithm.

Algorithm 1 Wolf Fish Collaborative Search Algorithm(WF-CSA)

Simulation

Input: The dimension dim; population size of Leader fish a; population size of scout fish b; population size of Hunter fish c; Maximum Iterations $N_{\max}$;
Output: $best\_fitness$;$best\_position$

Initialize the populations x_i $(i = 1, 2, \ldots,$ dim)
while $N \leq N_{\max}$ **do**
  Update $Visual$ according to (45)
  Update $Step$ of fish swarm algorithm according to (36)
  Update the crowding factor δ according to (46)
  Substitute x_i into (33),and reorder according to the fitness value from small to large, to get the position of the leader fish, scout fish, and hunter fish x_a, x_b, x_c
  Substitute x_a into (34),(35) to get $x_a{}'$
  Substitute $x_a{}'$ into (37) to get $x_a{}''$
  for i = 1:b **do**
    Substitute x_b into (38),(39) to get $x_b{}'$
    Substitute $x_b{}'$ into (40) to get $x_b{}''$
    Substitute $x_b{}''$ into (41) to get $x_b{}'''$
  end for
  for i = 1:c **do**
    Substitute x_c into (42) to get $x_c{}'$
    Substitute $x_c{}'$ into (43) to get $x_c{}''$
  end for
  Substitute $x_a{}''$, $x_b{}'''$,$x_c{}''$ into (33) respectively, and reorder them to get x_a, x_b, and x_c
  Therefore $best\_position = x_a$
  Substitute x_a into (33) to get $best\_fitness$
end while

In this study, we examine a model consisting of a cloud server layer, a blockchain network layer, an edge layer, and a user terminal layer. The cloud server is located at the center of a 100m × 100m, and five base stations are evenly distributed in the same area, each paired with a corresponding MEC server. The system contains a total of 50 user terminals randomly distributed within a 10m radius of each base station.In this paper, we set the simulation parameters based on the research results on blockchain consensus in the literature [29] and the research on cloud-side collaboration partial offloading in reference [38]. The validity of the WF-CSA algorithm has been confirmed through simulation, employing specific parameters delineated in Table 3.

Figure 2 illustrates the trend of fitness values with increasing number of iterations under different algorithms. The horizontal coordinate indicates the number of iterations and the vertical coordinate indicates the best fitness value. The blue line denotes WF-CSA, the red line denotes WPA (Wolf Pack Algorithm), the yellow line denotes AFSA (Artificial Fish Swarm Algorithm) and the purple line denotes BPSO [39] (Binary Particle Swarm Optimisation). The graphs show a clear trend: the fitness

Table 3 Simulation parameters

Parameters	Values
Number of cloud server	1
Number of MEC servers	5
Number of base stations	5
Number of users	50
Bandwidth for users	10 MHZ
Size of the task	100000 bit
task complexity	[25,50] CPU cycle/bit
Computing capacity of user terminals	[1.5, 2.0]$\times 10^9$ cycle/s
CPU computing capacity of MEC servers	2.0×10^{10} cycle/s
CPU computing capacity of cloud servers	2.5×10^{12} cycle/s
Number of CPU cycles required to generate/verify a signature	1.0×10^6 cycle
Number of CPU cycles required to generate/verify a MAC	1.0×10^7 cycle
Maximum number of transactions in a block	1500
Deadline for completion of tasks	10 s
transmission power	1 W
noise power	0.01 W

values of all four algorithms gradually decrease as the number of iterations increases.The WF-CSA algorithm converges to lower fitness values faster with fewer iterations. This is because in the WF-CSA algorithm, the position of the headfish first undergoes aggregation and tail chasing behaviours to obtain a more optimal position. The optimised position of the headfish can coordinate the behaviour of the whole school more effectively and guide the scout fish and fierce fish to explore in a more favourable direction in a more organised and targeted way. Meanwhile compared to the WPA,AFSA and BPSO algorithms, the WF-CSA algorithm introduces more behavioural strategies, allowing the algorithm to explore the solution space in a more comprehensive way rather than being limited to specific search directions.The WF-CSA algorithm converges to lower fitness values faster with fewer iterations.

Figure 3 shows the trend of total delay with increasing number of users for different algorithms. The horizontal axis represents the number of users and the vertical axis represents the total delay. It is clear from the figure that the total delay under different algorithms tends to increase as the number of users increases. This is due to the fact that the more the number of users, the more the computational tasks are, and the processing of these tasks leads to an increase in computational latency, which in turn leads to a rise in the total latency. Therefore, when performing the actual system implementation, in order to keep the total delay of the system unchanged, the impact of the increase in the number of users can be offset by increasing the number of base stations and MEC servers at the same time. In addition, the complexity of the whole system can be reduced by increasing the computational power of the MEC servers, thus improving the scalability of the system in terms of the number of user terminals. By comparing the curves of different algorithms, it is found that the total delay obtained by the WF-CSA algorithm can be reduced by as much as 53.48%, 36.1% and

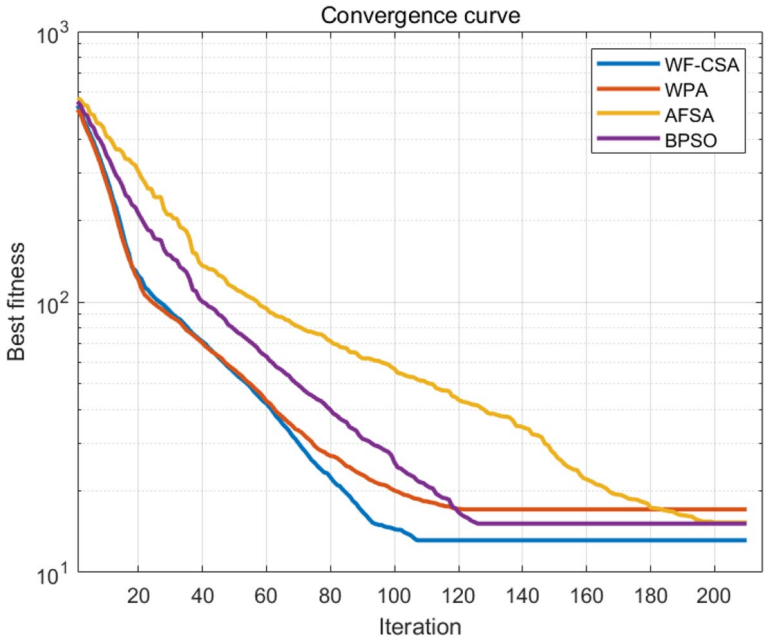

Fig. 2 The Correlation between Iteration Count and Optimal Fitness Value

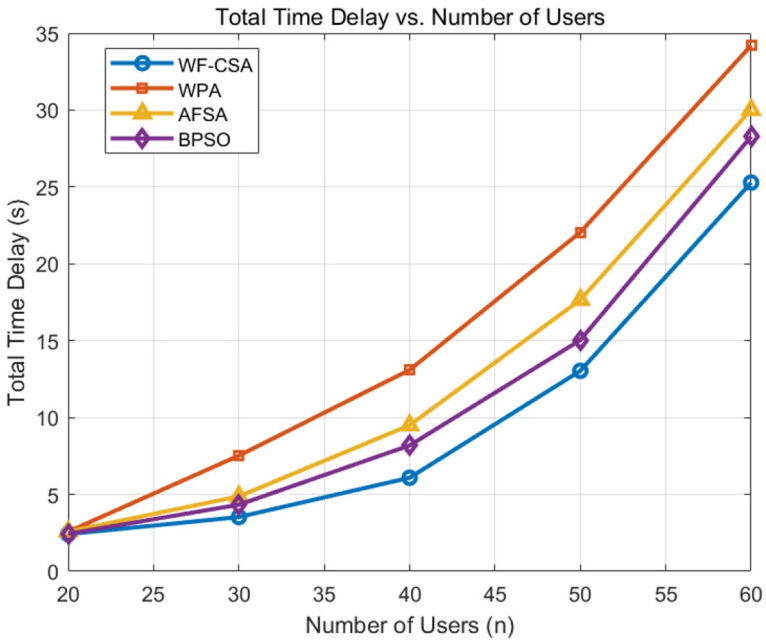

Fig. 3 Total time delay vs. number of users

25.66% compared to the WPA, AFSA and BPSO algorithms, respectively.

Figure 4 shows the trend of total delay with an increasing task size under different algorithms. The horizontal coordinate denotes the task size, and the vertical coordinate denotes the total delay. As observed from the figure, the total delay under different algorithms tends to increase as the task size increases. This is due to the task size increase, leading to a higher volume of data processing and computation. Additionally, the communication overhead between different nodes also increases, resulting in a larger total delay. When comparing the curves of different algorithms, WF-CSA reduces the total delay by as much as 42.58%, 28.58% and 15.93% as compared to AFSA, WPA and BPSO respectively.

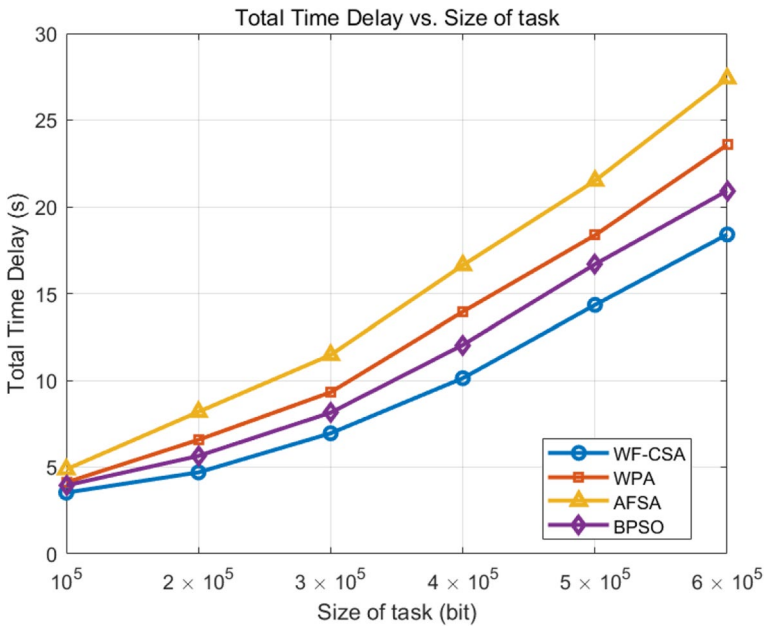

Fig. 4 Total time delay vs. size of task

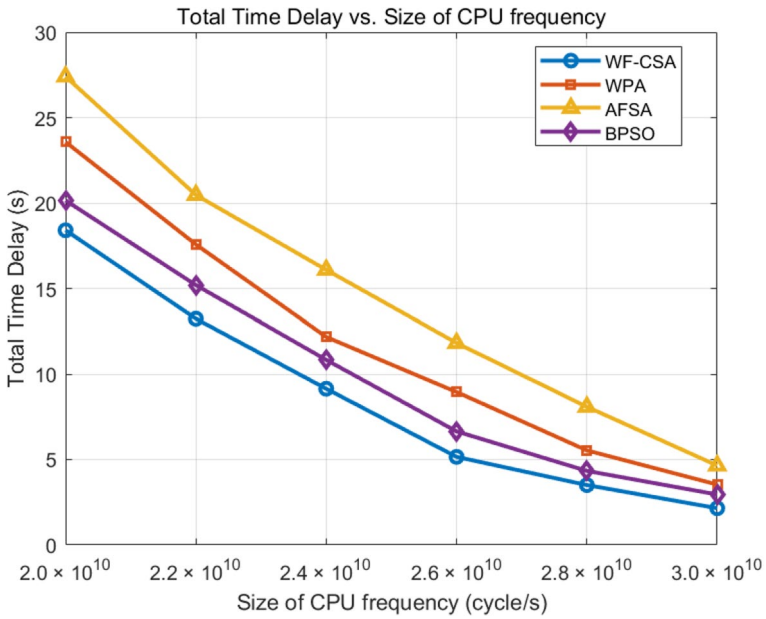

Fig. 5 Total time delay vs. size of CPU cycle frequency of MEC server

Figure 5 demonstrates the trend of total delay with an increasing CPU cycle frequency of the MEC server under different algorithms. The horizontal coordinate represents the CPU cycle frequency size of the MEC server, and the vertical coordinate represents the total delay. From the figure, it is evident that the total delay under different algorithms shows a decreasing trend as the CPU cycle frequency of the MEC server increases. This phenomena is attributed to the fact that with a higher CPU cycle frequency, the number of computational instructions executed per second also increases, resulting in improved processing speed of the tasks. Meanwhile, a higher CPU frequency may enhance task scheduling, making it more flexible and efficient, further reducing the

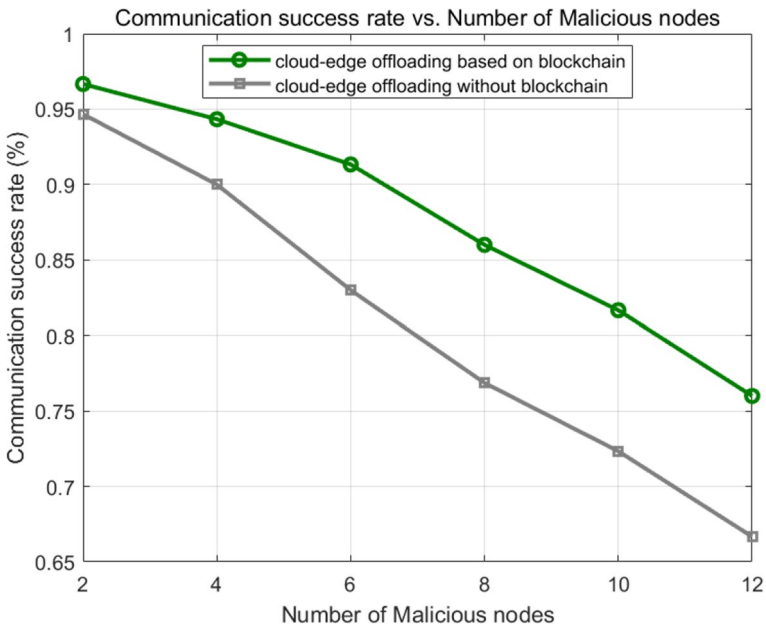

Fig. 6 Communication success rate vs. count of malicious nodes

total latency. Compared with AFSA, WPA and BPSO, the total delay obtained by WF-CSA can be reduced by up to 56.67%, 42.37% and 22.42%, respectively.

Figure 6 shows the trend of communication success rate as the number of malicious nodes increases in the cloud offloading system with and without blockchain. The horizontal axis represents the number of malicious nodes and the vertical axis represents the communication success rate. The figure shows that the communication success rate shows a significant decreasing trend as the number of malicious nodes increases. This is mainly due to the fact that malicious nodes may perform denial-of-service attacks, causing service interruptions and thus jeopardising the security and reliability of the system. Reducing the attacks of malicious nodes on the system through the blockchain PBFT consensus mechanism can effectively improve the communication success rate of the system, so the system designed in this paper has good scalability in dealing with the number of malicious nodes. The integration of blockchain into the cloud-side end offloading system can increase the communication success rate by 14.93% compared to the same system without blockchain.

Conclusion

In this study, we propose a collaborative task offloading model for 6G cloud network edge-end, incorporating multiple edge computing servers, communication base stations, user terminals, and a cloud server. The model enables the partial offloading of terminal device computational tasks to the MEC server and the cloud server for computation, or collaborative computation with other base stations having available computational resources . This model aims to reduce task processing time, ensure sufficient computing resources, and improve communication security and stability through the blockchain consensus mechanism.

Furthermore, this study proposes the CERMTOB algorithm, which formulates the minimization delay problem by incorporating the total task offloading delay, the total computational task processing latency, and the consensus delay of the blockchain network layer. The WF-CSA algorithm is used to optimal the offloading decisions with the aim of minimizing overall delay. The simulation results show the validity of the WF-CSA algorithm, showcasing reductions in total delay by up to 42.58%, 28.58% and 15.93% compared to the AFSA, WPA and BPSO algorithms across different task sizes. Furthermore, WF-CSA algorithm demonstrates superior results in scenarios involving changes in the count of users and the computational capacity of the MEC server. Additionally, the incorporation of blockchain in the cloud-side end offloading system contributes to an improved communication success rate, achieving improvements of up to 14.93% compared a system without blockchain.

Abbreviations

CERMTOB	Cloud-Edge Resource Management and Task Offloading in Blockchain Networks
WF-CSA	Wolf fish collaborative search algorithm
MN	Mobile node
DRL	Deep reinforcement learning
IAGT	Isotonic Action Generation Technique
BC-CED	blockchain-enabled cloud-edge device
CRL	collective reinforcement learning
QO-SRO	quasi-opposite search and rescue optimization
MDP	Markov Decision Process
RL	Reinforcement Learning
A3C	Asynchronous advantage actor-critic
OFDMA	Orthogonal Frequency Division Multiple Access
CS	Cloud server
BS	Base station
PBFT	practical byzantine fault tolerance
MAC	Message authentication code
NH	Node honesty
NC	Node capacity
WPA	Wolf Pack Algorithm
AFSA	Artificial Fish Swarm Algorithm
BPSO	Binary Particle Swarm Optimization

Acknowledgements

The authors express their gratitude to Beijing Information Science & Technology University for funding it through Researchers Supporting Program number (NO.2020KYNH212, NO. 2021CGZH302).

Authors' information

Shujie Tian received the B.S. degree in electronic information engineering from Beijing Information Science and Technology University in 2022. He is currently pursuing the M.S. degree in information and communication engineering from Beijing Information Science and Technology University. His research interests include wireless communications, resource allocation and blockchain.

Yuexia Zhang received her M.S. and Ph.D. degrees in information and communication engineering from Beijing University of Posts and Telecommunications in 2008. She has been a Full Professor at the School of Information and Communication Engineering of Beijing Information Science and Technology University since 2019. Her research interests include wireless cooperative communication technology, ultra-wideband technology and blockchain technology.

Yanxian Bi is currently a senior engineer in China Academy of Electronic and Information Technology, CETC Academy of Electronics and Information Technology Group Co., Ltd. He received his Ph.D. degree from the University of Beihang University in 2017. During 2015-2017, he studied in the University of Birmingham as a Visiting Scholar. His research interests focus on artificial intelligence.

Yuan Taifu, received his bachelor's degree in Electronic Information Engineering from Liaoning University of Science and Technology in 2014, and joined IBM China Research Institute in 2016 as a software development engineer.He joined Beijing Microchip Blockchain and Edge Computing Research Institute in 2019 as a Software Development Senior Engineer. His research interests focus on blockchain.

Authors' contributions

S. Tian wrote the main manuscript text and created all the figures. Y. Zhang designed the primary content of the experiments. Y. Bi provided all the support for the experimental environment. T. Yuan assisted in processing the trajectory data and helped organize the table information.All authors reviewed the manuscript.

Funding
This work was supported in part by Sub Project of National Key Research and Development plan in 2020 (NO. 2020YFC1511704), Beijing Science and Technology Project (Grant No. Z211100004421009), in part by the National Natural Science Younth Foundation of China (Grant No. 62301058).

Availability of data and materials
No datasets were generated or analysed during the current study.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 9 March 2024 Accepted: 19 April 2024
Published online: 06 May 2024

References

1. Jiang W, Han B, Habibi MA, Schotten HD (2021) The road towards 6G: a comprehensive survey. IEEE Open J Commun Soc 2:334–366. https://doi.org/10.1109/OJCOMS.2021.3057679
2. Zhang H, Shlezinger N, Guidi F, Dardari D, Eldar YC (2023) 6G Wireless Communications: From Far-Field Beam Steering to Near-Field Beam Focusing. IEEE Commun Mag 61(4):72–77. https://doi.org/10.1109/MCOM.001.2200259
3. Qi L, Liu Y, Zhang Y, Xu X, Bilal M, Song H (2022) Privacy-Aware Point-of-Interest Category Recommendation in Internet of Things. IEEE Internet Things J 9(21):21398–21408. https://doi.org/10.1109/JIOT.2022.3181136
4. Bharathiraja N, Shobana M, Vijay Anand M, Lathamanju R, Shanmuganathan C, Arulkumar V (2023) A secure and effective diffused framework for intelligent routing in transportation systems. Int J Comput Appl Technol 71(4):363–370. https://doi.org/10.1504/IJCAT.2023.132405
5. Nagu B, Arjunan T, Bangare ML, Karuppaiah P, Kaur G, Bhatt MW (2023) Ultra-low latency communication technology for Augmented Reality application in mobile periphery computing. J Behav Robot 14(1):20220112. https://doi.org/10.1515/pjbr-2022-0112
6. Banerjee A, Sufyanf F, Nayel MS, Sagar S (2018) Centralized Framework for Controlling Heterogeneous Appliances in a Smart Home Environment. International Conference on Information and Computer Technologies(ICICT), pp 78-82. https://doi.org/10.1109/INFOCT.2018.8356844
7. Sufyan F, Banerjee A (2023) Computation Offloading for Smart Devices in Fog-Cloud Queuing System. IETE J Res 69(3):1509–1521. https://doi.org/10.1080/03772063.2020.1870876
8. Sufyan F, Banerjee A (2019) Comparative Analysis of Network Libraries for Offloading Efficiency in Mobile Cloud Environment. International Journal of Advanced Computer Science and Applications 10(2): 574-584. https://doi.org/10.14569/IJACSA.2019.0100272
9. Sufyan F, Banerjee A (2020) Computation Offloading for Distributed Mobile Edge Computing Network: A Multiobjective Approach. IEEE Access 8:149915–149930. https://doi.org/10.1109/ACCESS.2020.3016046
10. Punia U, Batra T, Jindal U, Bharathiraja N, Tiwari RG, Pradeepa K (2023) An Improved Scheduling Algorithm for Grey Wolf Fitness Task Enrichment with Cloud. 2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT), pp 806-811. https://doi.org/10.1109/ICSSIT55814.2023.10061152
11. Prathiba SB, Raja G, Anbalagan S, Dev K, Gurumoorthy S, Sankaran AP (2022) Federated Learning Empowered Computation Offloading and Resource Management in 6G–V2X. IEEE Trans Network Sci Eng 9(5):3234–3243. https://doi.org/10.1109/TNSE.2021.3103124
12. Lin K, Li Y, Zhang Q, Fortino G (2021) AI-Driven Collaborative Resource Allocation for Task Execution in 6G-Enabled Massive IoT. IEEE Internet Things J 8(7):5264–5273. https://doi.org/10.1109/JIOT.2021.3051031
13. Qin P, Wang M, Zhao X, Geng S (2023) Content Service Oriented Resource Allocation for Space-Air-Ground Integrated 6G Networks: A Three-Sided Cyclic Matching Approach. IEEE Internet Things J 10(1):828–839. https://doi.org/10.1109/JIOT.2022.3203793
14. Goudarzi S, Soleymani SA, Wang W, Xiao P (2023) UAV-Enabled Mobile Edge Computing for Resource Allocation Using Cooperative Evolutionary Computation. IEEE Trans Aerosp Electron Syst 59(5):5134–5147. https://doi.org/10.1109/TAES.2023.3251967
15. Gong Y, Yao H, Wang J, Li M, Guo S (2022) Edge Intelligence-driven Joint Offloading and Resource Allocation for Future 6G Industrial Internet of Things. IEEE Trans Network Sci Eng. https://doi.org/10.1109/TNSE.2022.3141728
16. Qi L, Xu X, Wu X, Ni Q, Yuan Y, Zhang X (2023) Digital-Twin-Enabled 6G Mobile Network Video Streaming Using Mobile Crowdsourcing. IEEE J Sel Areas Commun 41(10):3161–3174. https://doi.org/10.1109/JSAC.2023.3310077
17. Ravindhar NV, Sasikumar S, Bharathiraja N (2024) Integration of cloud-based scheme with industrial wireless sensor network for data publishing in privacy of point source. Int J Comput Appl Technol 13(2):124–138. https://doi.org/10.1504/IJCC.2024.137408
18. Pandithurai O et al (2023) A Secured Industrial Wireless IoT Sensor Network Enabled Quick Transmission of Data with a Prototype Study. J Intell Fuzzy Syst 3445–3460. https://doi.org/10.3233/JIFS-224174
19. Xu X, Zhang X, Gao H, Xue Y, Qi L, Dou W (2020) BeCome: Blockchain-Enabled Computation Offloading for IoT in Mobile Edge Computing. IEEE Trans Ind Inf 16(6):4187–4195. https://doi.org/10.1109/TII.2019.2936869
20. Cao B et al (2023) Blockchain Systems, Technologies, and Applications: A Methodology Perspective. IEEE Commun Surv Tutorials 25(1):353–385. https://doi.org/10.1109/COMST.2022.3204702
21. Chishti MS, Sufyan F, Banerjee A (2021) Decentralized On-Chain Data Access via Smart Contracts in Ethereum Blockchain. IEEE Trans Netw Serv Manage 19(1):174–187. https://doi.org/10.1109/TNSM.2021.3120912
22. Huo R et al (2022) A Comprehensive Survey on Blockchain in Industrial Internet of Things: Motivations, Research Progresses, and Future Challenges. IEEE Commun Surv Tutorials 24(1):88–122. https://doi.org/10.1109/COMST.2022.3141490
23. Chen H, Luo X, Shi L, Cao Y, Zhang Y (2023) Security challenges and defense approaches for blockchain-based services from a full-stack architecture perspective. Blockchain: Res Appl 4(3):100135. https://doi.org/10.1016/j.bcra.2023.100135
24. Xu X, Gu J, Yan H, Liu W, Qi L, Zhou X (2023) Reputation-Aware Supplier Assessment for Blockchain-Enabled Supply Chain in Industry 4.0. IEEE Trans Ind Inf 19(4):5485-5494. https://doi.org/10.1109/TII.2022.3190380
25. Xiao Y, Zhang N, Lou W, Hou YT (2020) A Survey of Distributed Consensus Protocols for Blockchain Networks. IEEE Commun Surv Tutorials 22(2):1432–1465. https://doi.org/10.1109/COMST.2020.2969706
26. Xu J, Wang C, Jia X (2023) A survey of blockchain consensus protocols. ACM Comput Surv 55(278):1–35. https://doi.org/10.1145/3579845
27. Yao S et al (2022) Blockchain-Empowered Collaborative Task Offloading for Cloud-Edge-Device Computing. IEEE J Sel Areas Commun 40(12):3485–3500. https://doi.org/10.1109/JSAC.2022.3213358
28. Okegbile SD, Cai J, Alfa AS (2022) Performance Analysis of Blockchain-Enabled Data-Sharing Scheme in Cloud-Edge Computing-Based IoT Networks. IEEE Internet Things J 9(21):21520–21536. https://doi.org/10.1109/JIOT.2022.3181556
29. Li M et al (2022) Cloud-Edge Collaborative Resource Allocation for Blockchain-Enabled Internet of Things: A Collective Reinforcement Learning Approach. IEEE Internet Things J 9(22):23115–23129. https://doi.org/10.1109/JIOT.2022.3185289
30. Feng J, Yu FR, Pei Q, Chu X, Du J, Zhu L (2020) Cooperative Computation Offloading and Resource Allocation for Blockchain-Enabled Mobile-Edge Computing: A Deep Reinforcement Learning Approach. IEEE Internet Things J 7(7):6214–6228. https://doi.org/10.1109/JIOT.2019.2961707
31. Jain DK, Tyagi SKS, Neelakandan S, Prakash M, Natrayan L (2022) Metaheuristic Optimization-Based Resource Allocation Technique for Cybertwin-Driven 6G on IoE Environment. IEEE Trans Ind Inf 18(7):4884–4892. https://doi.org/10.1109/TII.2021.3138915

32. Zhang H, Liu X, Xu Y, Li D, Yuen C, Xue Q (2024) Partial Offloading and Resource Allocation for MEC-Assisted Vehicular Networks. IEEE Trans Veh Technol 73(1):1276–1288. https://doi.org/10.1109/TVT.2023.3306939
33. Hu H, Wang Q, Hu RQ, Zhu H (2021) Mobility-Aware Offloading and Resource Allocation in a MEC-Enabled IoT Network With Energy Harvesting. IEEE Internet Things J 8(24) 24:17541-17556. https://doi.org/10.1109/JIOT.2021.3081983
34. Zhao H, Deng S, Zhang C, Du W, He Q, Yin J (2019) A Mobility-Aware Cross-Edge Computation Offloading Framework for Partitionable Applications. 2019 IEEE International Conference on Web Services (ICWS), pp 193-200. https://doi.org/10.1109/ICWS.2019.00041
35. Qiu C, Yao H, Yu FR, Jiang C, Guo S (2020) A Service-Oriented Permissioned Blockchain for the Internet of Things. IEEE Trans Serv Comput 13(2):203–215. https://doi.org/10.1109/TSC.2019.2948870
36. Pourpanah F, Wang R, Lim CP et al (2023) A review of artificial fish swarm algorithms: Recent advances and applications. Artif Intell Rev 56(3):1867–1903. https://doi.org/10.1007/s10462-022-10214-4
37. Xu S, Li L, Zhou Z, Mao Y, Huang J (2022) A task allocation strategy of the UAV swarm based on multi-discrete wolf pack algorithm. Appl Sci 12(3):1331. https://doi.org/10.3390/app12031331
38. Su Q, Zhang Q, Li W, Zhang X (2024) Primal-Dual-Based Computation Offloading Method for Energy-Aware Cloud-Edge Collaboration. IEEE Trans Mob Comput 23(2):1534–1549. https://doi.org/10.1109/TMC.2023.3237938
39. Singh S, Kim DH (2023) Joint Optimization of Computation Offloading and Resource Allocation in C-RAN With Mobile Edge Computing Using Evolutionary Algorithms. IEEE Access 11:112693–112705. https://doi.org/10.1109/ACCESS.2023.3322650

Publisher's Note