Journal of Systems Architecture 118 (2021) 102167

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

# Cooperative computation offloading and resource allocation for delay minimization in mobile edge computing☆

Zhufang Kuang [a,*,1], Zhihao Ma [a,*], Zhe Li [a], Xiaoheng Deng [b,1]

[a] School of Computer and Information Engineering, Central South University of Forestry and Technology, Changsha, 410004, China
[b] School of Computer Science and Engineering, Central South University, Changsha 410010, China

ARTICLE INFO

*Keywords:*
Mobile edge computing
Cooperative computation offloading
Delay minimization
Resource allocation

ABSTRACT

Mobile edge computing (MEC) is a promising paradigm, which brings computation resources in proximity to mobile devices and allows the tasks of mobile devices to be offloaded to MEC servers with low latency. The joint problem of cooperative computation task offloading and resource allocation is a challenging issue. The joint problem of cooperative computation task offloading scheme and resource assignment in MEC is investigated in this paper, where the vertical cooperation among mobile devices, mobile edge server nodes and mobile cloud server nodes is considered, and the horizontal computation cooperation between edge nodes is considered as well. A computation offloading decision, cooperative selection, power allocation and CPU cycle frequency assignment problem is formulated. The objective is to minimize the latency while guaranteeing the constraint of transmission power, energy consumption and CPU cycle frequency. The formulated latency optimization problem is a nonconvex mixed-integer problem in general, which has binary variables and continuous variables. In order to solve the formulated problem. A joint iterative algorithm based on the Lagrangian dual decomposition, ShengJin Formula method, and monotonic optimization method is proposed. The CPU cycle frequence allocation is handled by the ShengJin Formula method due to the cubic equation of one variable about the CPU frequence allocation. The transmission power assignment is handled by the monotonic optimization method. In the algorithm convergence with different number of tasks, the proposed algorithm can quickly and effectively reach the convergence state and getting the minimum task execution delay. Numerical results demonstrate that the proposed algorithm outperforms the Full MEC, Full Local and Full Cloud three schemes in terms of execution latency.

## 1. Introduction

Nowadays, with the advantages of the 5G era and the rapidly increasing number of the edge devices, the amount of data has also increased rapidly to reach the level of the ZB [1]. In the era of big data, the data generated by massive edge devices which can no longer be efficiently processed by cloud computing. With the rapid development of Internet of Things and mobile Internet, the requirements of low energy consumption, high bandwidth, and low latency came into being. Mobile Edge Computing (MEC) can effectively solve these problems [2]. MEC is received extensive attention from academia and industry. A server to process and feedback the request of user at the edge closer to the user is set up in MEC. The computing ability of an edge server is similar to that of a cloud server, which is much greater than the computing ability of the local users, and it is closer to the edge device in geographical distance. MEC has been regarded as a complement of mobile cloud computing (MCC) to reduce energy consumption and the latency [3,4]. Now the MEC has become one of the key technologies of the next generation of mobile communications [5].

Recently, there are some research works about the computation offloading issues in MEC networks with different objectives, for example, energy efficient [6–11], latency [12–16]. However, these studies investigate computation offloading optimization schemes for reducing the energy consumption or the task execution latency. The vertical cooperation among mobile devices, mobile edge server nodes and

☆ This work was supported in part by the National Natural Science Foundation of China under Grants Nos. 62072477, 61309027, 61702562 and 61702561, the Hunan Provincial Natural Science Foundation of China under Grants No. 2018JJ3888, the Scientific Research Fund of Hunan Provincial Education Department, China under Grant No. 18B197, the National Key R&D Program of China under Grant No. 2018YFB1700200, the Hunan Key Laboratory of Intelligent Logistics Technology, China (2019TP1015).

* Corresponding authors.
*E-mail addresses:* zfkuangcn@163.com (Z. Kuang), 827272056@qq.com (Z. Ma), zhemao1992@qq.com (Z. Li), dxh@csu.edu.cn (X. Deng).
[1] Member, IEEE.

https://doi.org/10.1016/j.sysarc.2021.102167
Received 31 July 2020; Received in revised form 3 May 2021; Accepted 7 May 2021
Available online 19 May 2021

mobile cloud server nodes is ignored, and the horizontal computation task cooperation between mobile edge nodes, is not considered as well.

Cooperative Computation Offloading (CCO) is an effective technique to utilize the horizontal computation cooperation between mobile edge server nodes, and to utilize the vertical cooperation among mobile devices, mobile edge server nodes and mobile cloud server nodes, which can achieve a better resource utilization, balance the computation load, and further get low latency and improve the user's perceived performance. In CCO MEC, Each task can be offloaded to the mobile edge server nodes, or the remote mobile cloud nodes, or be processed locally. The different performance will be obtained with different offloading decisions. It is critical to make the offloading decision in the CCO MEC, for obtaining the lower task execution under the energy constraints, the mobile edge computing, mobile cloud computing, and the local computing could cooperate with each other. For example, the task processed locally may incur higher task execution delay, whereas the tasks offloaded to the mobile edge server nodes or remote mobile cloud nodes will result in additional transmission delay unavoidably. The cooperative computation offloading and resource allocation strategies in MEC have been investigated recently [17].

In MEC, the introduction of CCO can reduce the energy consumption or latency. However, the joint problem of offloading decision, cooperative selection and resource allocation are facing new challenging due to CCO and resource allocation are inter-coupled with each other. The joint problem of CCO and resource allocation includes offloading decision, cooperative selection, power allocation and CPU cycle frequency assignment. In order to minimize the latency while ensuring the transmission power constraint, energy consumption constraint and the CPU cycle frequency constraint, the optimal offloading decision, cooperative selection, and resource allocation are needed to obtain. Therefore, the cooperative computation offloading and resources allocation problem with the objective of minimizing the latency in MEC is studied, and the optimal scheme is put forward for CCO MEC.

In the last few years, as for improving energy efficiency, in [17], in order to improve the energy efficiency for latency-constrained MEC, the joint problem of optimizing the computation and communication resources allocation at both the user and the helper is investigated. In [18], the problem of computation task offloading with sleep control scheme with the goal of minimizing the long term energy consumption is investigated in MEC networks. In [19], for smart city scenario, to obtain the cooperation policy among fog nodes to enhance the users' quality of experience (QoE) is investigated. The goal is to improve energy effective. In [20], the computation offloading based on cooperations of multiple MEC-enabled base stations is investigated. The goal is to minimize the time and energy consumption. In [21], with the objective of minimizing the energy consumption, the joint problem of computation and communication resources in a multiple smart wearable devices MEC system is investigated. In [22], the joint problem of computation offloading and cooperative resource allocation in Wireless Power Transfer (WPT)-MEC system is studied. The objective is to maximization minimum energy efficiency. In [23], a cooperative offloading technique based on the Lagrangian suboptimal convergent computation offloading algorithm for multi-access MEC is proposed. The goal is to minimize the weight sum of energy consumption. In [24], in order to improve QoE, a cooperative task offloading mechanism for better serving mobile devices in MEC networks is investigated. A Deep Reinforcement Learning (DRL) named Double Dueling Deterministic Policy Gradient (Double DDPG) is proposed to make rational offloading policy. In [25], with the scenario considering a basic three-node consisting of a user, a helper, and an AP, a Non-Orthogonal Multiple-Access (NOMA)-aided cooperative computing in MEC system is investigated. Two optimization problems are formulated with different goal of minimizing energy consumption and maximizing offloading data. In [26], the cooperative edge computing that exploits parallel transmission over the whole resource block of NOMA is investigated. In [27], a CCO and resource allocation framework for blockchain-enabled MEC systems is investigated. The goal is to maximize the computation rate and the transaction throughput of blockchain systems. A deep reinforcement learning approach is proposed to solve the problem. In [28], the problem of learning-aided computation offloading for trusted collaborative MEC is investigated. An online learning-aided cooperative offloading mechanism is proposed. In [29], a multiuser cooperative mobile edge computing offloading in a multiuser interference environment is investigated. The goal is to minimize the total energy consumption of all mobile devices. In [30], the problem of cooperative scheduling for MEC with expected deadline guarantee is studied. In [31], the cooperative and distributed computation offloading for blockchain-empowered industrial Internet of Things (IoT) is investigated with the goal to minimize the economic cost.

As for reducing the latency, in [32], the joint problem of offloading decision and the computation resource allocation with the goal of minimizing the average task duration is studied. The scenario is a cooperative three-tier network by considering the vertical cooperation among mobile devices, mobile edge server nodes and mobile cloud server node, and the horizontal cooperation among mobile edge server nodes as well. However, the power allocation, and CPU cycle frequency assignment are not considered, the coupled among cooperative computation offloading and resource allocation are not considered as well. In [33], for latency sensitive tasks, the cooperation of mobile cloud computing and MEC in Internet of Things is studied. The single user computation offloading problem is handled by the branch and bound algorithm, and the multiuser computation offloading problem is designed an iterative heuristic resource allocation algorithm. In [34], the problem of computation offloading through the vehicular cloud (VC) is investigated. The goals is to minimize the overall response time. In [35], the performance gains of cooperative computation offloading for MEC enabled FiWi enhanced HetNets is studied. The objective is to minimize the average task execution time and energy consumption. An analytical framework to estimate the average response time and energy consumption for various offloading scenarios is presented. In [36], for the Industrial Internet of things (IIoT)-edge-cloud computing model, the multi-hop computation offloading problem with the goal of minimizing each task's computation time and energy consumption is investigated. In [37], a federated deep reinforcement learning-based cooperative edge caching framework is investigated. The goal is to reduce the performance loss and average delay. In [38], considering user mobility, distributed resources, tasks properties, and energy constraint, the joint problem of task assignment and power allocation with the goal of minimizing the total task execution latency is investigated.

However, the aforementioned algorithms and schemes with the objective of maximizing EE or minimizing latency in MEC networks is that the optimal coupling relationship among the cooperative relay selection, offloading decision and resource allocation for MEC networks is not considered. The cooperative computation offloading and resource allocation with the goal to minimize latency has not been investigated. In order to fill this gap, in this paper, the latency optimization problem of offloading decision, cooperative selection, power allocation and CPU cycle frequency assignment is investigated, while guaranteeing the transmission power constraint, energy consumption constraint and the CPU cycle frequency constraint. The main contributions of this paper are as follows:

- We model the offloading decision, cooperative scheme and resource allocation. The joint problem of the offloading decision, cooperative selection, power allocation and CPU cycle frequency assignment is considered.
- We construct a optimization offloading decision, cooperative relay selection and resource allocation problem with the goal of minimizing the total time of all the tasks. The formulated problem is a non-convex mixed-integer optimization problem.

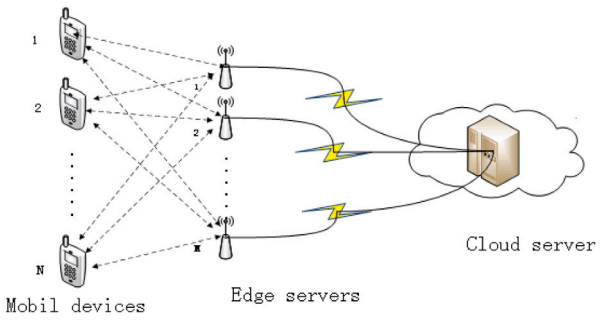


**Fig. 1.** System model.

- We solve the minimizing delay optimization problem, and propose a joint iterative algorithm of offloading decision, cooperative relay selection, power allocation, and CPU cycle frequency assignment based on the Lagrangian dual decomposition, ShenJing Formula method, and monotonic optimization method.

The rest of the paper is organized as follows. The system model and problem formulation are showed in Section 2. The joint optimization algorithm is demonstrated in Section 3. Numerical results are provided in Section 4, Finally, the conclusion is presented in Section 5.

## 2. System model and problem description

### 2.1. Network model

We consider a network scenario in MEC with $N$ mobile users, $M$ edge servers, and a single cloud server, as shown in Fig. 1. The number of mobile users and edge servers in the network are represented by $\mathbb{N} = \{1, 2, \ldots, N\}$ and $\mathbb{M} = \{1, 2, ..M\}$. Each mobile user $i \in \mathbb{N}$ has a task $I_i$, which is defined by $(D_i, C_i)$, where $D_i$ is the size of the amount of input data for the task, and $C_i$ is the number of CPU cycles required to process the task. Each task $I_i$ can be executed locally by itself or offloaded to the MEC servers or the cloud server. If the task is to be executed on the MEC servers, the mobile user sends the task to the target edge server. The result will be sent back to the user after the task is done on the edge servers. Note that if the computing resources of edge servers is insufficient or it cannot meet the constraint of transmission power or energy consumption, the task will be delivered to cloud server with cooperative computation offloading. We denote the $x_{i,j} \in \{0, 1\}$ as the offloading decision. Specifically, $x_{i,j} = 1$ indicates that the task $I_i$ of user $i$ is offloaded to MEC server $j$ to execute; otherwise $x_{i,j} = 0$. We denote the $y_{i,j} \in \{0, 1\}$ as the cooperative mode selection. Let $y_{i,j} = 1$ indicate whether the task $I_i$ of user $i$ is offloaded to the cloud server with the cooperative MEC server $j$. The main parameters used in this paper are shown in Table 1.

### 2.2. Communication model

The task can be offloaded to the MEC servers or the cloud server with cooperative computation offloading. The communication model is given as following.

(1) Offloading to MEC server: We denote $R_{i,j}$ as the transmission rate from mobile user $i$ to the MEC server $j$, according to Shannon Theory [39], which is given by

$$R_{i,j} = B \log_2 \left( 1 + \frac{g_0 \left( \frac{d_0}{d_i} \right)^{\theta} p_{i,j}}{N_0 B} \right) \tag{1}$$

where $p_{i,j}$ is the transmission power from mobile user $i$ to the MEC server $j$ for task $I_i$, $B$ is the channel bandwidth, $g0$ is the path loss constant, $\theta$ is the path loss exponent, $d0$ is the reference distance, $d_i$ is the distance between the mobile device $i$ to MEC server $j$, and $N_0$ is the noise power spectral density.

**Table 1**
Main parameters.

| Notation | Definition |
|---|---|
| $\mathbb{N}$ | The set of mobile users |
| $\mathbb{M}$ | The set of edge servers |
| $x_{i,j}$ | The offloading decision of the task $I_i$ |
| $y_{i,j}$ | The cooperative mode selection of the task $I_i$ |
| $R_{i,j}$ | The transmission rate from mobile user $i$ to the MEC server $j$ |
| $r_{i,j}$ | The transmission rate of the MEC server $j$ transmitting the task of mobile user $i$ to the cloud server |
| $T_i^l$ | The execution time of the task executed locally |
| $E_i^l$ | The energy consumption of the task executed locally |
| $T_{i,j}$ | The transmission time executed on the MEC server |
| $E_{i,j}$ | The transmission energy consumption executed on the MEC server |
| $T_{i,j}^{Mec,c}$ | The task processing time executed on the MEC server |
| $E_{i,j}^{Mec,c}$ | The task computing energy consumption executed on the MEC server |
| $T_{i,j}^{Mec}$ | The task total execution time executed on the MEC server |
| $E_{i,j}^{Mec}$ | The task total energy consumption executed on the MEC server |
| $t_{i,j}$ | The task transmission time from the edge server $j$ to the cloud server |
| $e_{i,j}$ | The task transmission energy consumption from the edge server $j$ to the cloud server |
| $T_{i,j}^{Cloud,c}$ | The task processing time executed on the cloud server |
| $T_{i,j}^{Cloud}$ | The task total processing time executed on the cloud server |

(2) Offloading to the cloud server: When the task is offloaded to the cloud server, the task is transmitted to the MEC server firstly, then it is transmitted to the cloud server. We denote $r_{i,j}$ as the transmission rate of the MEC server $j$ transmitting the task $I_i$ of mobile user $i$ to the cloud server.

$$r_{i,j} = B \log_2 \left( 1 + \frac{g_0 \left( \frac{d_0}{d_j} \right)^{\theta} q_{i,j}}{N_0 B} \right) \tag{2}$$

where $q_{i,j}$ is the transmission power from the MEC server $j$ to the cloud server for transmitting the task $I_i$, $d_j$ is the distance between the MEC server $j$ to the cloud server.

### 2.3. Computing model

The task can be executed locally or on MEC servers or cloud server with cooperative computation offloading. The computing model is given as following.

(1) Executed locally: When the task is executed locally. The execution time $T_i^l$ and energy consumption $E_i^l$ are expressed as

$$T_i^l = \frac{C_i}{f_{i,Loc}} \tag{3}$$

$$E_i^l = \mu C_i \left( f_{i,Loc} \right)^2 \tag{4}$$

where $f_{i,Loc}$ is the allocated CPU frequency for $I_i$ locally, and $\mu$ is the coefficient related to the chip architecture.

(2) Executed on the MEC servers: When the task is offloaded to the MEC servers for execution. The transmission time $T_{i,j}$ and energy consumption $E_{i,j}$ are expressed by

$$T_{i,j} = \frac{D_i}{R_{i,j}} \tag{5}$$

$$E_{i,j} = p_{i,j} T_{i,j} \tag{6}$$

The task processing time $T_{i,j}^{Mec,c}$ and the task computing energy consumption $E_{i,j}^{Mec,c}$ on the MEC server are expressed by

$$T_{i,j}^{Mec,c} = \frac{C_i}{f_{i,j,Mec}} \tag{7}$$

$$E_{i,j}^{Mec,c} = \mu C_i \left(f_{i,j,Mec}\right)^2 \tag{8}$$

where $f_{i,j,Mec}$ is the CPU frequency allocated by MEC server $j$ for processing the task $I_i$.

When the task offloaded to the MEC server, the task total time $T_{i,j}^{Mec}$ and the task total energy consumption $E_{i,j}^{Mec}$ are expressed by

$$T_{i,j}^{Mec} = T_{i,j} + T_{i,j}^{Mec,c} \tag{9}$$

$$E_{i,j}^{Mec} = E_{i,j} + E_{i,j}^{Mec,c} \tag{10}$$

(3) Executed on the cloud server: When the task is offloaded to the cloud server for execution via the edge server cooperative computation offloading. The task transmission time $t_{i,j}$ and transmission energy consumption $e_{i,j}$ are expressed by

$$t_{i,j} = \frac{D_i}{r_{i,j}} \tag{11}$$

$$e_{i,j} = q_{i,j} t_{i,j} \tag{12}$$

The task processing time $T_{i,j}^{Cloud,c}$ on the cloud server is given by

$$T_{i,j}^{Cloud,c} = \frac{C_i}{f_{Cloud}} \tag{13}$$

where $f_{Cloud}$ is the CPU frequency assigned to the tasks when the tasks is offloaded to the cloud server, it is a constant in this paper. The task total processing time $T_{i,j}^{Cloud}$ on cloud server is as follows:

$$T_{i,j}^{Cloud} = T_{i,j} + t_{i,j} + T_{i,j}^{Coud,c} \tag{14}$$

### 2.4. Problem formulation

We investigate the joint problem of offloading decision, cooperative mode selection and resource allocation. The resource allocation includes transmission power allocation and CPU frequency assignment of the mobile users and the MEC servers. The objective is to minimize the task total completing delay. The objective function is shown in (15). The constraints of energy consumption, CPU cycle frequency and transmission power of the mobile users and the MEC servers are considered, respectively. But, the total energy consumption constraints on cloud server are not considered in this paper.

$$\min_{\chi} T = \sum_{i=1}^{N} \left[ T_i^l \left(1 - \sum_{j=1}^{M} \left(x_{i,j} + y_{i,j}\right)\right) + \sum_{j=1}^{M} x_{i,j} T_{i,j}^{Mec} + \sum_{j=1}^{M} y_{i,j} T_{i,j}^{Cloud} \right] \tag{15}$$

Subject to

$$\sum_{j=1}^{M} \left(x_{i,j} + y_{i,j}\right) \le 1 \quad \forall i \in \mathbb{N} \tag{16}$$

$$E_i^l \left(1 - \sum_{j=1}^{M} \left(x_{i,j} + y_{i,j}\right)\right) + \sum_{j=1}^{M} \left(x_{i,j} + y_{i,j}\right) E_{i,j} \le E_{i,Loc} \quad \forall i \in \mathbb{N} \tag{17}$$

$$\sum_{i=1}^{N} \left(x_{i,j} E_{i,j}^{Mec,c}\right) + \sum_{i=1}^{N} \left(y_{i,j} e_{i,j}\right) \le E_{j,Mec} \quad \forall j \in \mathbb{M} \tag{18}$$

$$f_{i,loc} \left(1 - \sum_{j=1}^{M} \left(x_{i,j} + y_{i,j}\right)\right) \le f_{i,Loc}^{Max} \quad \forall i \in \mathbb{N} \tag{19}$$

$$\sum_{i=1}^{N} \left(x_{i,j} f_{i,j,Mec}\right) \le f_{j,Mec}^{Max} \quad \forall j \in \mathbb{M} \tag{20}$$

$$\sum_{j=1}^{M} \left(x_{i,j} p_{i,j} + y_{i,j} p_{i,j}\right) \le P_{i,Loc}^{Max} \quad \forall i \in \mathbb{N} \tag{21}$$

$$\sum_{i=1}^{N} \left(y_{i,j} q_{i,j}\right) \le P_{j,Mec}^{Max} \quad \forall j \in \mathbb{M} \tag{22}$$

$$x_{i,j} \in \{0,1\}, y_{i,j} \in \{0,1\}, p_{i,j} \ge 0, q_{i,j} \ge 0, f_{i,Loc} \ge 0, f_{i,j,Mec} \ge 0 \tag{23}$$

where $\chi = \left\{x_{i,j}, y_{i,j}, p_{i,j}, q_{i,j}, f_{i,Loc} f_{i,j,Mec}\right\}$ is the optimization variables. Constraint (16) indicates that the task can only be executed locally or offloaded to the MEC server and cloud server. Eqs. (17) and (18) represent local energy consumption constraint and MEC server energy consumption constraints, respectively. Denote $E_{i,Loc}$ and $E_{j,Mec}$ as the value of maximizing energy consumption for mobile user $i$ and MEC server $j$, respectively. Eqs. (19) and (20) are CPU frequency constraints of local user $i$ and edge server $j$, respectively. Denote $f_{i,Loc}^{Max}$ and $f_{j,Mec}^{Max}$ as the maximizing CPU frequency for mobile user $i$ and MEC server $j$, respectively. Eqs. (21) and (22) define the transmission power constraints of local user $i$ and edge server $j$, respectively, where $P_{i,Loc}^{Max}$ and $P_{j,Mec}^{Max}$ is the maximizing transmission power. Eqs. (23) represents the value range of each optimization variable.

## 3. Optimization algorithm

### 3.1. Solving the problem

In order to address the formulated latency optimization problem, the convex optimization method are used. The above formulated problem includes binary variables and continuous variables. The continuous optimization variables included the CPU frequency and transmission power of mobile user and MEC server, and the binary optimization variables contain offloading decision and cooperative mode selection. The binary variables $x_{i,j}$ and $y_{i,j}$ are relaxed into continuous variable according to the composite function criterion, the concave and convex nature of the logarithmic function and the nature of the perspective function can prove that the objective and constraint functions are concave functions. The Lagrange function of the formulated problem with constraints (16)–(22) is given by

$$\begin{aligned} H(\chi, \lambda) = & \sum_{i=1}^{N} \left[ T_i^l \left(1 - \sum_{j=1}^{M} \left(x_{i,j} + y_{i,j}\right)\right) + \sum_{j=1}^{M} x_{i,j} T_{i,j}^{Mec} \right. \\ & \left. + \sum_{j=1}^{M} y_{i,j} T_{i,j}^{Cloud} \right] + \sum_{i=1}^{N} \lambda_{1,i} \left[ \sum_{j=1}^{M} \left(x_{i,j} + y_{i,j}\right) - 1 \right] \\ & + \sum_{i=1}^{N} \lambda_{2,i} \left[ E_i^l \left(1 - \sum_{j=1}^{M} \left(x_{i,j} + y_{i,j}\right)\right) + \sum_{j=1}^{M} x_{i,j} E_{i,j} \right. \\ & \left. + \sum_{j=1}^{M} y_{i,j} E_{i,j} - E_{i,loc} \right] \\ & + \sum_{j=1}^{M} \lambda_{3,j} \left[ \sum_{i=1}^{N} x_{i,j} E_{i,j}^{Mec,c} + \sum_{i=1}^{N} y_{i,j} e_{i,j} - E_{j,Mec} \right] \\ & + \sum_{i=1}^{N} \lambda_{4,i} \left[ f_{i,loc} \left(1 - \sum_{j=1}^{M} \left(x_{i,j} + y_{i,j}\right)\right) - f_{i,Loc}^{Max} \right] \\ & + \sum_{j=1}^{M} \lambda_{5,j} \left[ \sum_{i=1}^{N} x_{i,j} f_{i,j,Mec} - f_{i,j,Mec}^{Max} \right] \\ & + \sum_{i=1}^{N} \lambda_{6,i} \left[ \sum_{j=1}^{M} x_{i,j} p_{i,j} + \sum_{j=1}^{M} y_{i,j} p_{i,j} - P_{i,Loc}^{Max} \right] \\ & + \sum_{j=1}^{M} \lambda_{7,j} \left[ \sum_{i=1}^{N} y_{i,j} q_{i,j} - P_{j,Mec}^{Max} \right] \end{aligned} \tag{24}$$

where $\chi = \left\{x_{i,j}, y_{i,j}, p_{i_i,j}, q_{i,j}, f_{i,loc}, f_{i,j,Mec}\right\}$ is the optimization variables, and the Lagrange multiplier variable matrix $\lambda$ =

$\left[\lambda_{1,i}, \lambda_{2,i}, \lambda_{3,j}, \lambda_{4,i}, \lambda_{5,j}, \lambda_{6,i}, \lambda_{7,j}\right]$ is the Lagrange multipliers corresponding to constraints (16)–(22).

The dual function of the mathematical model is defined as $h(\lambda) = \min_{\chi} H(\lambda, \chi)$. Then the dual problem is $\min_{\lambda} h(\lambda)$, s.t. $\lambda \geq 0$.

(1) Solving the CPU frequency allocation variables: The Karush–Kuhn–Tucker (KKT) conditions and the ShengJin Formula methods [40,41] are adopted to obtain the optimal solution. We take the first derivative of $\Delta(\chi, \lambda)$ with respect to $f_{i,loc}$ and $f_{i,j,Mec}$, respectively, and set them to zero, which are given by

$$\begin{aligned}\frac{\partial \Delta}{\partial f_{i,loc}} =& -\frac{C_i}{f_{i,loc}^2}\left(1-\sum_{j=1}^{M}\left(x_{i,j}+y_{i,j}\right)\right) \\ &+ 2\lambda_{2,i}\mu C_i f_{i,loc}\left(1-\sum_{j=1}^{M}\left(x_{i,j}+y_{i,j}\right)\right) \\ &+ \lambda_{4,i}\left(1-\sum_{j=1}^{M}\left(x_{i,j}+y_{i,j}\right)\right)=0\end{aligned} \tag{25}$$

$$\begin{aligned}\frac{\partial \Delta}{\partial f_{i,j,Mec}} =& x_{i,j}\left(\frac{C_i}{-f_{i,j,Mec}^2}\right) \\ &+ \lambda_{3,j}x_{i,j}2\mu C_i f_{i,j,Mec} + \lambda_{5,j}x_{i,j}=0\end{aligned} \tag{26}$$

Solving Eqs. (25) and (26), we can obtain the cubic equation of one variable about $f_{i,loc}$ and $f_{i,j,Mec}$, respectively, which are expressed by

$$2\lambda_{2,i}\mu C_i f_{i,loc}^3 + \lambda_{4,i} f_{i,loc}^2 - C_i = 0 \tag{27}$$

$$2\lambda_{3,j}\mu C_i f_{i,j,Mec}^3 + \lambda_{5,j} f_{i,j,Mec}^2 - C_i = 0 \tag{28}$$

The optimization CPU frequency allocation variables $f_{i,loc}^*$ and $f_{i,j,Mec}^*$ are obtained by the ShengJin Formula method. The solve steps are as follows:

- Calculating the multiple root discriminants and the total discriminant. The cubic equation of one variable is defined as $aX^3+bX^2+cX+d = 0 (a,b,c,d \in R, a \neq 0)$. For (27), $a = 2\lambda_{2,i}\mu C_i, b = \lambda_{4,i}, c = 0, d = -C_i$. For (28), $a = 2\lambda_{3,j}\mu C_i, b = \lambda_{5,j}, c = 0, d = -C_i$. The multiple root discriminants are calculated as following: $A = b^2 - 3ac, B = bc - 9ad, C = c^2 - 3bd$. The total discriminant is calculated as following: $\Delta = B^2 - 4AC$.
- When $A = B = 0$, according to the 1st ShengJin Formula, the three equal solutions of the unary cubic equation are expressed by

$$X_1 = X_2 = X_3 = \frac{-b}{3a} = \frac{-c}{b} = \frac{-3d}{c} \tag{29}$$

- When $\Delta = B^2 - 4AC > 0$, according to the 2nd ShengJin Formula, the three solutions of the unary cubic equation are given by

$$X_1 = \frac{-b - (\sqrt[3]{Y_1} + \sqrt[3]{Y_2})}{3a} \tag{30}$$

$$X_2 = X_3 = \frac{-\mathrm{b} + \frac{1}{2}(\sqrt[3]{Y_1} + \sqrt[3]{Y_2}) \pm \frac{\sqrt{3}}{2}(\sqrt[3]{Y_1} + \sqrt[3]{Y_2})i}{3a} \tag{31}$$

where $Y_1 = Y_2 = A\mathrm{b} + 3\mathrm{a}\left(\frac{-B \pm \sqrt{B^2-4AC}}{2}\right), \mathrm{i}^2 = -1$.

- When $\Delta = B^2 - 4AC = 0$, according to the 3rd ShengJin Formula, the three solutions of the unary cubic equation are expressed by

$$X_1 = \frac{-b}{a} + K \tag{32}$$

$$X_2 = X_3 = \frac{-K}{2} \tag{33}$$

where $K = \frac{B}{A}, (A \neq 0)$.

- When $\Delta = B^2 - 4AC < 0$, according to the 4th ShengJin Formula, the three solutions of the unary cubic equation are given by

$$X_1 = \frac{-b - 2\sqrt{A}\cos\frac{\theta}{3}}{3a} \tag{34}$$

**Algorithm 1** CA-ShengJin Algorithm

**Input:** $a, b, c$
**Output:** $X_1, X_2, X_3$

```
Initialize the cubic equation of one variable: ax^3 + bx^2 + cx + d = 0
Calculate A = b^2 − 3ac
Calculate B = bc − 9ad
Calculate C = c^2 − 3bd
Calculate Δ = B^2 − 4AC
if A = B = 0 then
    X1 = X2 = X3 = −b/3a = −c/b = −3d/c
else if Δ > 0 then
    X1 = [−b−(∛Y1 + ∛Y2)] / 3a
    X2 = X3 = (−b + 1/2(∛Y1 + ∛Y2) ± √3/2 (∛Y1 + ∛Y2)i) / 3a
else if Δ = 0 then
    X1 = −b/a + K
    X2 = X3 = −K/2
else if Δ < 0 then
    X1 = (−b − 2√A cos θ/3) / 3a
    X2 = X3 = (−b + √A(cos θ/3 ± √3 sin θ/3)) / 3a
end if
```

$$X_2 = X_3 = \frac{-b + \sqrt{A}\left(\cos\frac{\theta}{3} \pm \sqrt{3}\sin\frac{\theta}{3}\right)}{3a} \tag{35}$$

where $\theta = \arccos \mathrm{T}$, and $\mathrm{T} = \frac{2Ab-3aB}{2\sqrt{A^3}} (A > 0, -1 < T < 1)$.

The solving the CPU frequency allocation variable by ShengJin Formula method (CA-ShengJin) is described in Algorithm 1.

(2) Solving the transmission power allocation variables: The Karush–Kuhn–Tucker (KKT) conditions and bisection search methods are adopted to obtain the optimal solution. We take the first derivative of $\Delta(\chi, \lambda)$ with respect to $p_{i,j}$ and $q_{i,j}$, respectively, and set them to zero, which are given by

$$\begin{aligned}\frac{\partial \Delta}{\partial p_{i,j}} =& \left(x_{i,j}+y_{i,j}\right)\frac{D_i}{\left(-R_{i,j}^2\right)}\frac{\partial R_{i,j}}{\partial p_{i,j}} \\ &+ \lambda_{2,i}\left[\left(x_{i,j}+y_{i,j}\right)\left(\frac{D_i}{R_i} + p_{i,j}\frac{D_i}{\left(-R_{i,j}^2\right)}\frac{\partial R_{i,j}}{\partial p_{i,j}}\right)\right] \\ &+ \lambda_{6,i}\left(x_{i,j}+y_{i,j}\right)=0\end{aligned} \tag{36}$$

$$\begin{aligned}\frac{\partial \Delta}{\partial q_{i,j}} =& y_{i,j}\frac{D_i}{\left(-r_{i,j}^2\right)}\frac{\partial r_{i,j}}{\partial q_{i,j}} \\ &+ \lambda_{3,j}y_{i,j}\left(\frac{D_i}{r_{i,j}} + q_{i,j}\frac{D_i}{\left(-r_{i,j}^2\right)}\frac{\partial r_{i,j}}{\partial q_{i,j}}\right) + \lambda_{,j}y_{i,j}=0\end{aligned} \tag{37}$$

Solving Eqs. (36) and (37), we can obtain the logarithmic equation of one variable about $p_{i,j}$ and $q_{i,j}$ as follows:

$$\begin{aligned}&\lambda_{2,i}\left[D_i B\log_2\left(1+A_{i,j}p_{i,j}\right) - p_{i,j}\frac{D_i B A_{i,j}}{\left(1+A_{i,j}p_{i,j}\right)\ln 2}\right] \\ &+ \lambda_{6,i}B^2\log_2^2\left(1+A_{i,j}p_{i,j}\right) - \frac{D_i B A_{i,j}}{\left(1+A_{i,j}p_{i,j}\right)\ln 2} = 0\end{aligned} \tag{38}$$

$$\begin{aligned}&\lambda_{3,j}\left[D_i B\log_2\left(1+S_j q_{i,j}\right) - q_{i,j}\frac{D_i B S_j}{\left(1+S_j q_{i,j}\right)\ln 2}\right] \\ &+ \lambda_{7,j}B^2\log_2^2\left(1+S_j q_{i,j}\right) - \frac{D_i B S_j}{\left(1+S_j q_{i,j}\right)\ln 2} = 0\end{aligned} \tag{39}$$

where $A_{i,j} = \frac{g_0\left(\frac{d_0}{d_{i,j}}\right)^{\theta}}{N_0 B}$ and $S_j = \frac{g_0\left(\frac{d_0}{d_j}\right)^{\theta}}{N_0 B}$

**Algorithm 2** PAB Algorithm

**Input:** $a, b, \omega$
**Output:** $c$

Initialize: Confirming interval [a,b]
$c = \frac{a+b}{2}$
**while** $f(c) \neq 0$ **do**
  **if** $f(a) \cdot f(c) < 0$ **then**
    $b \leftarrow c$
  **else**
    $a \leftarrow c$
  **end if**
**end while**

The logarithmic equation is difficult to handle. The second order partial derivative with respect to $p_{i,j}$ and $q_{i,j}$ is obtained. The partial derivation of Eqs. (38) and (39) are given by

$$\frac{\partial^2 \Delta}{\partial \left(p_{i,j}\right)^2} = \lambda_{2,i} D_i B A_{i,j} \log_2\left(1 + A_{i,j} p_{i,j}\right) + \lambda_{6,i} B^2 A_{i,j} \left[\frac{2\log_2\left(1 + A_{i,j} p_{i,j}\right)}{\ln 2} + \log_2^2\left(1 + A_{i,j} p_{i,j}\right)\right] \tag{40}$$

$$\frac{\partial^2 \Delta}{\partial \left(q_{i,j}\right)^2} = \lambda_{3,j} D_i B S_j \log_2\left(1 + S_j q_{i,j}\right) + \lambda_{\eta,j} B^2 S_j \left[\frac{2\log_2\left(1 + S_j q_{i,j}\right)}{\ln 2} + \log_2^2\left(1 + S_j q_{i,j}\right)\right] \tag{41}$$

As we can know from through analysis that (40) and (41) are always greater than 0. Then, the Eqs. (38) and (39) are monotonically increasing in the definition domain. So, the optimal $p^*_{i,j}$ and $q^*_{i,j}$ are obtained based on bisection search method. Let $f\left(p_{i,j}\right) = \frac{\partial \Delta}{\partial p_{i,j}}$ and $f\left(q_{i,j}\right) = \frac{\partial \Delta}{\partial q_{i,j}}$. Then, the optimal solution is obtained by continuous bipartite approximation in the interval $\left[0, P^{Max}_{i,loc}\right]$ and [0, $p^{Max}_{j,Mec}$], respectively. The solving steps are as follows:

- Step 1: Give interval [a, b] and mathematical error $\omega$. Verify $f(a) \cdot f(b) < 0$;
- Step 2: Find the midpoint of the interval [a, b], $c = \frac{a+b}{2}$;
- Step 3: Calculate $f(c)$, if $f(c) = 0, c$ is zero point (optimal value) of function;
- Step 4: If $f(a) \cdot f(c) < 0$, let $b = c$;
- Step 5: If $f(c) \cdot f(b) < 0$, let $a = c$;
- Step 6: Whether the difference between two points in the interval reaches accuracy $\omega$, if $|a - b| < \omega$, then get the approximate value of a (or b) of Eqs. (38) and (39), and end the optimal value solution search step. Otherwise, go to step 2.

The transmission Power Allocation variables solved by the Bisection search method (PAB) is described in Algorithm 2.

(3) Solving the offload decision variable and cooperative mode selection variable: The variable $x_{i,j}$ decide the task offloading to MEC server $j$, and the variable $y^*_{i,j}$ decide tasks offloading to the cloud server with cooperative computation offloading.

The offloading decision variables $x_{i,j}$ and $y_{i,j}$ are binary variable. In order to obtain the optimal value, the Lagrange function formula (24) can be transformed into the following:

$$\begin{aligned}\Delta(\lambda, \chi) = &\sum_{i=1}^{N}\left[T^l_i + \lambda_{2,i} E^l_i + \lambda_{4,j} f_{i,loc}\left(1 - \sum_{j=1}^{M}\left(x_{i,j} + y_{i,j}\right)\right)\right. \\ &+ \sum_{j=1}^{M} x_{i,j}\left(T^{Mec}_{i,j} + \lambda_{1,i} + \lambda_{2,i} E_{i,j} + \lambda_{3,j} E^{Mec,c}_{i,j}\right. \\ &\left.+\lambda_{s,j} f_{i,j,Mec} + \lambda_{6,i} p_{i,j}\right) + \sum_{j=1}^{M} y_{i,j}\left(T^{Cloud}_{i,j} + \lambda_{1,i}\right. \\ &\left.\left.+\lambda_{2,i} E_{i,j} + \lambda_{3,j} e_{i,j} + \lambda_{6,i} p_{i,j} + \lambda_{1,j} q_{i,j}\right)\right] + \sum_{i=1}^{N}(-1) \\ &+ \sum_{i=1}^{N} \lambda_{2,i}\left(-E_{i,loc}\right) + \sum_{j=1}^{M} \lambda_{3,j}\left(-E_{j,Mec}\right) \\ &+ \sum_{i=1}^{N} \lambda_{4,i}\left(-f^{Max}_{i,Loc}\right) + \sum_{j=1}^{M} \lambda_{5,j}\left(-f^{Max}_{i,j,Mec}\right) \\ &+ \sum_{i=1}^{N} \lambda_{6,i}\left(-P^{Max}_{i,Loc}\right) + \sum_{j=1}^{M} \lambda_{7,j}\left(-P^{Max}_{j,Mec}\right)\end{aligned} \tag{42}$$

The offloading decision variable $x_{i,j}$ and the cooperative mode selection variable $y_{i,j}$ can be obtained by the follows:

$$x_{i,j} = \begin{cases} 1, & \zeta_{i,j} = \zeta^x_{i,j} \quad \& \quad (i,j) = \arg\min\left(\zeta_{i,j}\right) \\ 0, & \text{else} \end{cases} \tag{43}$$

$$y_{i,j} = \begin{cases} 1, & \zeta_{i,j} = \zeta^y_{i,j} \quad \& \quad (i,j) = \arg\min\left(\zeta_{i,j}\right) \\ 0, & \text{else} \end{cases} \tag{44}$$

where $\zeta_{i,j}$, $\zeta^x_{i,j}$ and $\zeta^y_{i,j}$ are expressed as

$$\begin{aligned}\zeta_{i,j} = \min\Big[&(T^l_i + \lambda_{2,i} E^l_i + \lambda_{4,i} f_{i,loc}), (T^{Mec}_{i,j} + \lambda_{1,i} \\ &+ \lambda_{2,i} E_{i,j} + \lambda_{3,j} E^{Mec,c}_{i,j} + \lambda_{5,j} f_{i,j,Mec} + \lambda_{6,i} p_{i,j}), (T^{Cloud}_{i,j} \\ &+ \lambda_{1,i} + \lambda_{2,i} E_{i,j} + \lambda_{3,j} e_{i,j} + \lambda_{6,i} p_{i,j} + \lambda_{7,j} q_{i,j})\Big]\end{aligned} \tag{45}$$

$$\begin{aligned}\zeta^x_{i,j} = &T^{Mec}_{i,j} + \lambda_{1,i} + \lambda_{2,i} E_{i,j} + \lambda_{3,j} E^{Mec,c}_{i,j} \\ &+ \lambda_{5,j} f_{i,j,Mec} + \lambda_{6,i} p_{i,j}\end{aligned} \tag{46}$$

$$\begin{aligned}\zeta^y_{i,j} = &T^{Cloud}_{i,j} + \lambda_{1,i} + \lambda_{2,i} E_{i,j} + \lambda_{3,j} e_{i,j} \\ &+ \lambda_{6,i} p_{i,j} + \lambda_{7,j} q_{i,j}\end{aligned} \tag{47}$$

According to interrelation of offloading decision and cooperative mode selection. The combinations of variable $x_{i,j}$ and the variable $y_{i,j}$ are given by

- $(x_{i,j}, y_{i,j})$ = (0,0): task $i$ executed locally on device;
- $(x_{i,j}, y_{i,j})$ = (1,0): task $i$ executed on MEC server $j$;
- $(x_{i,j}, y_{i,j})$ = (0,1): task $i$ sent to MEC server $j$, further sent and executed on cloud server;
- $(x_{i,j}, y_{i,j})$ = (1,1): not possible.

(4) Lagrangian Multiplier Update: After the above procedures, we can update the Lagrangian multiplier vector $\lambda$, for the dual problem $\min_\lambda h(\lambda)$, s.t. $\lambda \geq 0$ by utilizing the above obtained solution, which is obtained for given $\lambda$. Specifically, we can update $\lambda$ as follows by using the subgradient method [42].

$$\lambda_{1,i}(m+1) = \lambda_{1,i}(m) + \alpha_1\left[\sum_{j=1}^{M}\left(x_{i,j} + y_{i,j}\right) - 1\right] \tag{48}$$

$$\begin{aligned}\lambda_{2,i}(m+1) = \lambda_{2,i}(m) + \alpha_2\Bigg[&E^l_i\left(1 - \sum_{j=1}^{M}\left(x_{i,j} + y_{i,j}\right)\right) \\ &+ \sum_{j=1}^{M}(x_{i,j} + y_{i,j})E_{i,j} - E_{i,loc}\Bigg]\end{aligned} \tag{49}$$

**Algorithm 3** CODM Algorithm

**Input:** $D_i, C_i, B, f_{Cloud}, N_0, d_i, d_j, \theta, d_0$
**Output:** $x_{i,j}, y_{i,j}, p_{i,j}, q_{i,j}, f_{i,Loc}, f_{i,j,Mec}$

Initialize: $\lambda, \alpha, m = 1, H_0^* = 0, \Delta = 1$
**while** $\Delta > 10^{-6}$ **do**
  Solve $f_{i,loc}^*$ and $f_{i,j,Mec}^*$ using Algorithm 1
  Solve $p_{i,j}^*$ and $q_{i,j}^*$ using Algorithm 2
  Calculate $T_i^l, T_{i,j}^{Mec}, T_{i,j}^{cloud}$ according to (3), (9), (14)
  Calculate $E_i^l, E_{i,j}, E_{i,j}^{Mec}, e_{i,j}$ according (4), (6), (10), (12)
  Solve the offload decision variable $x_{i,j}^*$ according to (43)
  Solve the mode selection variable $y_{i,j}^*$ according to (44)
  Update Lagrangian Multiplier $\lambda$ according to (48)–(54)
  Calculate $H_m^*$ according to (24)
  $m \leftarrow m + 1$
  $\Delta = \left| H_m^* - H_{m-1}^* \right|$
**end while**

$$\lambda_{3,j}(m+1) = \lambda_{3,j}(m) + \alpha_3 \left( \sum_{i=1}^{N} x_{i,j} E_{i,j}^{Mec,c} + \sum_{i=1}^{N} y_{i,j} e_{i,j} - E_{j,Mec} \right) \tag{50}$$

$$\lambda_{4,i}(m+1) = \lambda_{4,i}(m) + \alpha_4 \left[ f_{i,loc} \left( 1 - \sum_{j=1}^{M} \left( x_{i,j} + y_{i,j} \right) \right) - f_{i,Loc}^{Max} \right] \tag{51}$$

$$\lambda_{5,j}(m+1) = \lambda_{5,j}(m) + \alpha_5 \left( \sum_{i=1}^{N} x_{i,j} f_{i,j,Mec} - f_{i,j,Mec}^{Max} \right) \tag{52}$$

$$\lambda_{6,i}(m+1) = \lambda_{6,i}(m) + \alpha_6 \left[ \sum_{j=1}^{M} (x_{i,j} + y_{i,j}) p_{i,j} - P_{i,Loc}^{Max} \right] \tag{53}$$

$$\lambda_{7,j}(m+1) = \lambda_{7,j}(m) + \alpha_7 \left[ \sum_{i=1}^{N} y_{i,j} q_{i,j} - P_{j,Mec}^{Max} \right] \tag{54}$$

where $\alpha = [\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7]$ is step size for Lagrangian multiplier vector $\lambda$.

### 3.2. Proposed algorithm

We summarize the proposed the Cooperative relay selection, Offloading Decision, power allocation, and CPU cycle frequency assignment Method (CODM) is described in Algorithm 3.

## 4. Numerical results

In this section, the proposed CODM algorithm is evaluated. The simulation parameters are summarized in Table 2.

In order to analyze the convergence of the algorithm proposed in the paper, in this section, the convergence of the CODM algorithm is compared and analyzed by setting three sets of comparative experiments. We set up three sets of experiments, these comparative experiments have 10, 20 and 30 tasks, respectively. The simulation results can be seen from Fig. 2, The total task execution delay increases with the number of tasks. The convergence of iterations of 10, 20 and 30 tasks are around 175, 180 and 180 generations, respectively. It can be seen from that the proposed CODM algorithm used to optimize local power, local CPU frequency, transmission power and MEC CPU frequency in different network environments, it can quickly and effectively reach the convergence state and getting the minimum task execution delay. The total task execution delay is 5 s when the number of tasks is 10.

Fig. 3 compares the impact of different number of tasks on the total execution delay of tasks, and compares the CODM algorithm with the other three task execution methods. We can see from Fig. 3, with the number of tasks continues to increase, the total task execution time continues to increase. The proposed CODM can obtain the lowest execution delay than the other three algorithms. For the Full Local algorithm, the task execution delay is the highest of the four methods. For the cloud server for execution, although the cloud server can be allocated the highest frequency, but the cloud server is farthest from the users, when the number of tasks are not enough, the delay of tasks executed on cloud server is the largest. As the number of task increased to 13, the execution delay starts to be less than the delay of all tasks executed locally, and the gap continues to increase. For the Full MEC algorithm, the task execution delay is smaller than the previous Full Local and Full Cloud methods.

**Table 2**
Simulation parameters.

| Notation | Value | Definition |
|---|---|---|
| $N$ | 10 | Number of mobile users |
| $M$ | 3 | Number of edge servers |
| $D_i$ | (0, 200) | Input data size (kb) |
| $C_i$ | (0, $16 \times 10^8$) | Number of required CPU cycles (Cycles) |
| $g_0$ | −40 | Path loss constant (dB) |
| $d_j$ | (100, 200) | Distance between MEC server and cloud server (m) |
| $d_i$ | (80, 100) | Distance between users and MEC servers (m) |
| $\theta$ | 3 | Path loss exponent |
| $N_0$ | −174 | Gaussian white noise power density (dB/Hz) |
| $B$ | 20 | Bandwidth (kHz) |
| $\mu$ | $10^{-25}$ | Coefficient related to the chip architecture |
| $E_{i,Loc}$ | 100 | Local energy consumption constraints (mJ) |
| $E_{i,Mec}$ | 1600 | MEC server energy consumption constraints (mJ) |
| $f_{i,Loc}^{Max}$ | 0.8 | Local maximum frequency (GHz) |
| $f_{i,Mec}^{Max}$ | 3.2 | Maximum frequency of MEC servers (GHz) |
| $f_{\text{Cloud}}$ | 6 | Maximum frequency of cloud server (GHz) |
| $P_{i,Mec}^{Max}$ | 300 | Maximum transmission power of MEC server (mW) |
| $P_{i,loc}^{Max}$ | 100 | Local maximum transmission power (mW) |

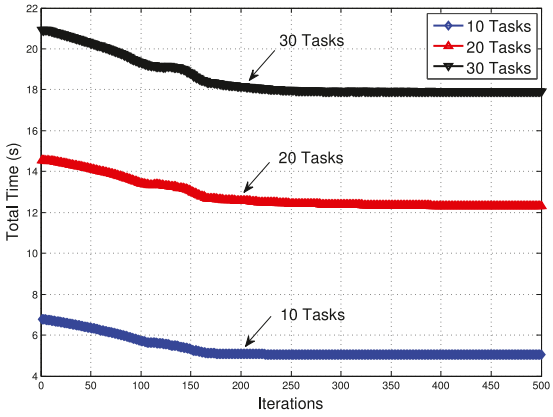


**Fig. 2.** Algorithm convergence.

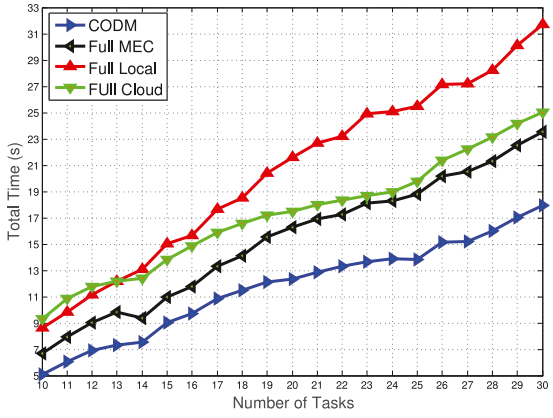


**Fig. 3.** Comparison of task delay under different task numbers.

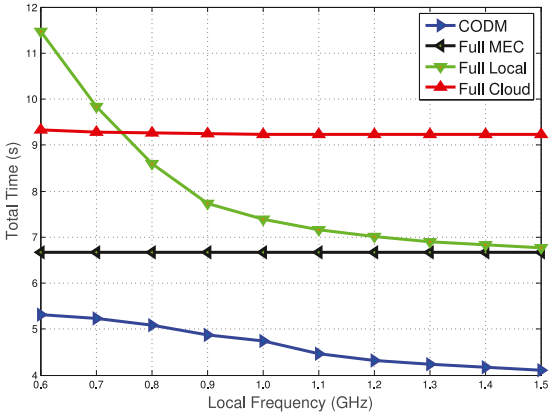


**Fig. 4.** Effect of different local frequency constraints on task execution delay.

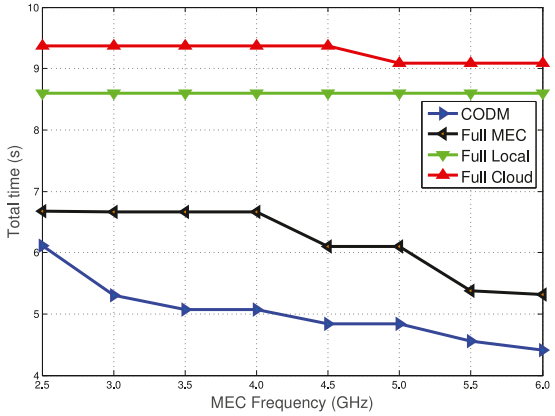


**Fig. 5.** Effect of different MEC frequency constraints on task execution delay.

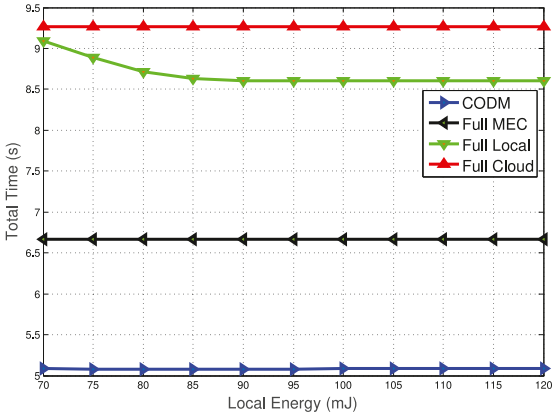


**Fig. 6.** Effect of different local energy consumption constraints on task delay.

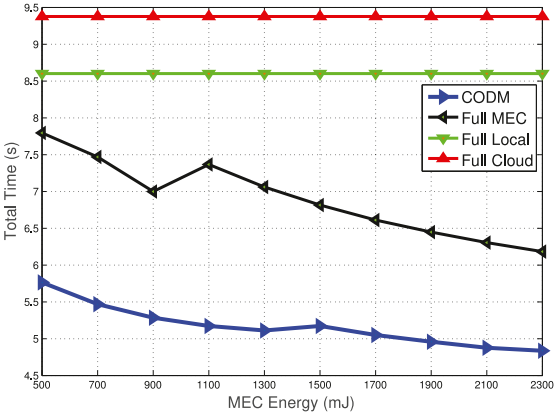


**Fig. 7.** Effect of different MEC energy consumption constraints on task delay.

Fig. 4 shows the impact of different local CPU frequency constraints on the total task execution delay, and compares the CODM algorithm with the other three task execution methods. As we can see from Fig. 4, the number of tasks is 10. The CODM algorithm can obtain the lowest execution delay, because of the optimization local CPU frequency can be archived by CODM algorithm. The task execution delay of CODM is significantly lower than the other three methods, and the total execution delay of the task will decrease with the increasing of the local CPU frequency constraint. For Full Local method, the task execution delay decreases with the local CPU frequency constraints. When the local CPU frequency increase, the locally allocated frequencies also increase, so the execution delay of the task decreased. And then, the speed of the delay reduction also be decreased with the local CPU frequency constraints. For the Full Cloud and the Full MEC methods, the increasing of local CPU frequency constraint does not affect the execution delay of the cloud server and MEC server. The delay of tasks executed on MEC servers is smaller than the task offloaded to cloud server, because of MEC servers is closer to users.

Fig. 5 presents the impact of different MEC CPU frequency constraints on the total task execution delay. The number of tasks is 10 as well. We can see from Fig. 5. Task execution delay obtained by the CODM algorithm is significantly lower than the total task execution delay of the other three methods, this is because of the optimization of CODM algorithm, which decreases continuously with the increasing of the MEC CPU frequency constraint. For the Full MEC method, the delay of task executed on MEC servers decreases with the MEC CPU frequency constraint. The reason is that the allocated frequency of the MEC server also be increased with the MEC CPU frequency constraint of the MEC server increases. For the Full Cloud method, the increase of MEC frequency constraint has almost no effect on its execution delay as well. The execution delay decreases with $f_{i,Mec}^{Max} = 4.5$. The reason is that offloading decision changes, the more tasks are executed by MEC server. For the Full Local method, the increase of MEC frequency constraint has almost no effect on its execution delay.

Fig. 6 illustrates the impact of different local energy consumption constraints on the total task execution delay, and compares the CODM algorithm with the other three task execution methods. We can see from Fig. 6, the proposed CODE algorithm can obtain the lowest execution delay than the other three methods. For the CODE, the Full Cloud, and the Full MEC methods, the task execution latency dos not change at all with different local energy consumption constraints. For the Full Local method, the execution delay of the tasks decreases with the increasing of the local energy consumption constraint, when the $E_{i,Loc}$ is less then 85 mJ. After then, the local energy consumption constraints has no effect on the execution latency as well.

Fig. 7 compares the impact of different MEC energy consumption constraints on the total task execution delay, and compares the CODM algorithm with the other three task execution methods. As we can see from Fig. 7, the task execution delay of the proposed method is significantly lower than the other three methods. The task execution delay decreases with the energy consumption constraint of the MEC server increasing. For the Full MEC method, the task execution delay decreases with the increasing of the energy consumption constraint of the MEC server due to the positive correlation between energy consumption and frequency. For the Full Cloud and the Full Local methods, the changes of MEC server energy consumption constraints have no effect on task execution delay.

## 5. Conclusion

In this paper, the optimization of offloading decision, cooperative relay selection, power allocation and CPU cycle frequency assignment is investigated. Our objective is to minimize the task execution latency. The vertical cooperation among mobile devices, mobile edge server nodes and mobile cloud server nodes is considered, and the horizontal computation cooperation among mobile edge nodes is considered as well. The convex optimization theory is adopted to solve the optimization problem. A joint iterative algorithm of offloading decision, cooperative relay selection, and resource allocation based on the Lagrangian dual decomposition, ShenJing Formula method, and monotonic optimization method is proposed. Simulation results show the proposed CODM algorithm can obtain the lower execution latency for different network parameters outperforms the existing schemes. In the future, considering the transmission time of the task results, energy harvesting-based cooperative computation offloading and resource allocation in MEC systems will be studied.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## References


[1] S. Wan, R. Gu, T. Umer, K. Salah, X. Xu, Toward offloading internet of vehicles applications in 5g networks, IEEE Trans. Intell. Transp. Syst. PP (99) (2021) 1–9.

[2] X. Wang, Y. Han, V.C.M. Leung, D. Niyato, X. Yan, X. Chen, Convergence of edge computing and deep learning: A comprehensive survey, IEEE Commun. Surv. Tutor. 22 (2) (2020) 869–904.

[3] Z. Kuang, L. Li, J. Gao, L. Zhao, A. Liu, Partial offloading scheduling and power allocation for mobile edge computing systems, IEEE Internet Things J. 6 (4) (2019) 6774–6785.

[4] H. Dai, X. Zeng, Z. Yu, T. Wang, A scheduling algorithm for autonomous driving tasks on mobile edge computing servers, J. Syst. Archit. 94 (2019) 14–23.

[5] G. Premsankar, M.D. Francesco, T. Taleb, Edge computing for the internet of things: A case study, IEEE Internet Things J. 5 (2) (2018) 1275–1284.

[6] X. Chen, Y. Cai, L. Li, M. Zhao, B. Champagne, L. Hanzo, Energy-efficient resource allocation for latency-sensitive mobile edge computing, IEEE Trans. Veh. Technol. 69 (2) (2020) 2246–2262.

[7] M. Sheng, Y. Wang, X. Wang, J. Li, Energy-efficient multiuser partial computation offloading with collaboration of terminals, radio access network, and edge server, IEEE Trans. Commun. 68 (3) (2020) 1524–1537.

[8] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, X.S. Shen, Energy efficient dynamic offloading in mobile edge computing for internet of things, IEEE Trans. Cloud Comput. PP (no) (2020) 1.

[9] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, X.S. Shen, Toffee: Task offloading and frequency scaling for energy efficiency of mobile devices in mobile edge computing, IEEE Trans. Cloud Comput. PP (no) (2020) 1.

[10] T. Wang, D. Zhao, S. Cai, W. Jia, A. Liu, Bidirectional prediction-based underwater data collection protocol for end-edge-cloud orchestrated system, IEEE Trans. Ind. Inform. 16 (7) (2020) 4791–4799.

[11] S. Wan, X. Li, Y. Xue, W. Lin, X. Xu, Efficient computation offloading for internet of vehicles in edge computing-assisted 5g networks, J. Supercomput. 76 (4) (2020) 2518–2547.

[12] L. Qian, Y. Wu, J. Ouyang, Z. Shi, B. Lin, W. Jia, Latency optimization for cellular assisted mobile edge computing via non-orthogonal multiple access, IEEE Trans. Veh. Technol. 69 (5) (2020) 5494–5507.

[13] U. Saleem, Y. Liu, S. Jangsher, X. Tao, Y. Li, Latency minimization for D2d-enabled partial computation offloading in mobile edge computing, IEEE Trans. Veh. Technol. 69 (4) (2020) 4472–4486.

[14] C. Shu, Z. Zhao, Y. Han, G. Min, H. Duan, Multi-user offloading for edge computing networks: A dependency-aware and latency-optimal approach, IEEE Internet Things J. 7 (3) (2020) 1678–1689.

[15] J. Luo, X. Deng, H. Zhang, H. Qi, QoE-driven computation offloading for edge computing, J. Syst. Archit. 97 (2019) 34–39.

[16] C. Chen, B. Liu, S. Wan, P. Qiao, Q. Pei, An edge traffic flow detection scheme based on deep learning in an intelligent transportation system, IEEE Trans. Intell. Transp. Syst. PP (99) (2021) 1–13.

[17] X. Cao, F. Wang, J. Xu, R. Zhang, S. Cui, Joint computation and communication cooperation for energy-efficient mobile edge computing, IEEE Internet Things J. 6 (3) (2019) 4188–4200.

[18] S. Wang, X. Zhang, Z. Yan, W. Wang, Cooperative edge computing with sleep control under nonuniform traffic in mobile edge networks, IEEE Internet Things J. 6 (3) (2019) 4295–4306.

[19] Y. Dong, S. Guo, J. Liu, Y. Yang, Energy-efficient fair cooperation fog computing in mobile edge networks for smart city, IEEE Internet Things J. 6 (5) (2019) 7543–7554.

[20] W. Fan, Y. Liu, B. Tang, F. Wu, Z. Wang, Computation offloading based on cooperations of mobile edge computing-enabled base stations, IEEE Access 6 (2018) 22622–22633.

[21] Y. Li, G. Xu, J. Ge, X. Fu, P. Liu, Communication and computation cooperation in wireless network for mobile edge computing, IEEE Access 7 (2019) 106260–106274.

[22] L. Ji, S. Guo, Energy-efficient cooperative resource allocation in wireless powered mobile edge computing, IEEE Internet Things J. 6 (3) (2019) 4744–4754.

[23] Z. Hong, W. Chen, H. Huang, S. Guo, Z. Zheng, Multi-hop cooperative computation offloading for industrial IoT-edge-cloud computing environments, IEEE Trans. Parallel Distrib. Syst. 30 (12) (2019) 2759–2774.

[24] X. He, H. Lu, H. Huang, Y. Mao, K. Wang, S. Guo, Qoe-based cooperative task offloading with deep reinforcement learning in mobile edge networks, IEEE Wirel. Commun. 27 (3) (2020) 111–117.

[25] Y. Huang, Y. Liu, F. Chen, Noma-aided mobile edge computing via user cooperation, IEEE Trans. Commun. 68 (4) (2020) 2221–2235.

[26] Y. Liu, Exploiting NOMA for cooperative edge computing, IEEE Wirel. Commun. 26 (5) (2019) 99–103.

[27] J. Feng, F.R. Yu, Q. Pei, X. Chu, J. Du, L. Zhu, Cooperative computation offloading and resource allocation for blockchain-enabled mobile-edge computing: A deep reinforcement learning approach, IEEE Internet Things J. 7 (7) (2020) 6214–6228.

[28] Y. Li, X. Wang, X. Gan, H. Jin, L. Fu, X. Wang, Learning-aided computation offloading for trusted collaborative mobile edge computing, IEEE Trans. Mob. Comput. PP (99) (2020) 1.

[29] C. Gong, F. Lin, X. Gong, Y. Lu, Intelligent cooperative edge computing in the internet of things, IEEE Internet Things J. PP (no) (2020) 1.

[30] C. Liu, K. Li, J. Liang, K. Li, Cooper-sched: A cooperative scheduling framework for mobile edge computing with expected deadline guarantee, IEEE Trans. Parallel Distrib. Syst. PP (99) (2021) 1.

[31] W. Chen, Z. Zhang, Z. Hong, C. Chen, J. Wu, S. Maharjan, Z. Zheng, Y. Zhang, Cooperative and distributed computation offloading for blockchain-empowered industrial internet of things, IEEE Internet Things J. 6 (5) (2019) 8433–8446.

[32] Y. Wang, X. Tao, X. Zhang, P. Zhang, Y.T. Hou, Cooperative task offloading in three-tier mobile computing networks: An ADMM framework, IEEE Trans. Veh. Technol. 68 (3) (2019) 2763–2776.

[33] Z. Ning, P. Dong, X. Kong, F. Xia, A cooperative partial computation offloading scheme for mobile edge computing enabled internet of things, IEEE Internet Things J. 6 (3) (2019) 4804–4814.

[34] F. Sun, F. Hou, N. Cheng, M. Wang, H. Zhou, L. Gui, X. Shen, Cooperative task scheduling for computation offloading in vehicular cloud, IEEE Trans. Veh. Technol. 67 (11) (2018) 11049–11061.

[35] A. Ebrahimzadeh, M. Maier, Cooperative computation offloading in fiwi enhanced 4g hetnets using self-organizing MEC, IEEE Trans. Wirel. Commun. 19 (7) (2020) 4480–4493.

[36] J.H. Anajemba, Y. Tang, C. Iwendi, M. Alenezi, M. Mittal, Optimal cooperative offloading scheme for energy efficient multi-access edge computation, IEEE Access 8 (2020) 53931–53941.

[37] X. Wang, C. Wang, X. Li, V.C.M. Leung, T. Taleb, Federated deep reinforcement learning for internet of things with decentralized cooperative edge caching, IEEE Internet Things J. PP (no) (2020) 1.

[38] U. Saleem, Y. Liu, S. Jangsher, Y. Li, T. Jiang, Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing, IEEE Trans. Wireless Commun. 20 (1) (2021) 360–374.

[39] C.E. Shannon, A mathematical theory of communication, ACM SIGMOBILE Mob. Comput. Commun. Rev. 5 (1) (2001) 3–55.

[40] C. Li, W. Dong, L. Ding, H. Zhang, H. Sun, Transfer characteristics of the nonlinear parity-time-symmetric wireless power transfer system at detuning, Energies 13 (19) (2020) 5175.

[41] H. Liu, Q. Liu, S. Zhou, C. Li, S. Yuan, A NURBS interpolation method with minimal feedrate fluctuation for CNC machine tools, Int. J. Adv. Manuf. Technol. 78 (5–8) (2015) 1241–1250.

[42] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.