Collaborative Task Processing and Resource Allocation Based on Multiple MEC Servers

Lei Shi[1,2], Shilong Feng[1,2(✉)], Rui Ji[1,2], Juan Xu[1,2], Xu Ding[1,3], and Baotong Zhan[4]

[1] School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230009, China
fsl@mail.hfut.edu.cn
[2] Engineering Research Center of Safety Critical Industrial Measurement and Control Technology, Ministry of Education, Hefei 230009, China
[3] Institute of Industry and Equipment Technology, Hefei University of Technology, Hefei 230009, China
[4] Taian Hualu Metalforming Machine Tool Co., Ltd., Taian 271000, Shandong, China

Abstract. Mobile Edge Computing (MEC), an emerging computing paradigm, shifts computing and storage capabilities from the cloud to the network edge, aiming to meet the delay requirements of emerging applications and save backhaul network bandwidth. However, compared to cloud servers, MEC servers have limited computing and storage capabilities, which cannot meet the massive offloading demands of users during high-load periods. In this context, this paper proposes a multi-ENs collaborative task processing model. The model aims to formulate optimal offloading decisions and allocate computing resources for tasks to minimize system delay and cost. To solve this problem, we propose an online algorithm based on Lyapunov optimization called OKMTA, which can work online without the need for predicting future information. Specifically, the problem is formulated as a mixed-integer nonlinear programming (MINLP) problem and decomposed into two subproblems for solution. By using the Lagrange multiplier method to solve the computing resource allocation problem of tasks, and by using matching theory to solve the offloading decision problem of tasks. The simulation results show that our algorithm can achieve near-optimal delay performance while satisfying the long-term system average cost constraint.

Keywords: Mobile Edge Computing · Lyapunov Optimization · Collaborative Task Processing · Resource Allocation

1 Introduction

With the development of the Internet of Things(IoTs), an increasing number of resource-intensive tasks are being deployed on user devices, such as mobile

The work is supported by the Key Technology Research and Development Project of Hefei, NO. 2021GJ029.

H. Gao et al. (Eds.): CollaborateCom 2023, LNICST 561, pp. 385–402, 2024.
https://doi.org/10.1007/978-3-031-54521-4_21

games, video analysis, and virtual/augmented reality (VR/AR). However, the limited computing and storage resources of these devices are often difficult to meet the processing demands of user tasks [1]. Cloud computing becomes an effective solution for offloading tasks from local devices to remote clouds for processing. However, the large data transmission volume and long backhaul distances between users and remote clouds can lead to large communication overhead, resulting in prolonged system delay and high offloading cost [2].

To solve this problem, mobile edge computing (MEC) technology as a new computing paradigm has attracted much attention in academia [3,4]. By offloading tasks from terminal devices to nearby MEC servers, it provides users with low-delay and high-bandwidth services [5,6]. In [7], Ren et al. proposed a latency-optimized method for resource allocation, aiming to minimize the communication delay between mobile devices and MEC servers. Chen et al. discussed the problem of energy efficient dynamic offloading in MEC for internet of things, aiming to minimize the average transmission energy consumption and guarantee the performance of devices [8].

However, compared to remote cloud servers, MEC servers have limited communication and computing capabilities, which cannot satisfy a large number of offloading demands from users during high-load periods. Therefore, to effectively process tasks, the collaboration between cloud computing and edge computing has become a new research trend. Delay-sensitive tasks can be offloaded to the edge for processing, while delay-tolerant tasks can be offloaded to the remote cloud server for processing [9]. Concurrently, optimizing task offloading decisions and resource allocation to minimize task processing delay and cost has become a crucial aspect of cloud-edge collaborative processing for computing tasks [9–11]. Ren et al. proposed a cloud-edge collaborative computing framework, aiming to minimize task processing delay by combining remote cloud and edge resources [9]. In MEC networks with limited communication capabilities, [10] detailed a scheme for cloud-edge-end collaborative task offloading to improve system performance and user experience. Dai et al. considered the cooperation scenarios of cloud computing and MEC, and designed an iterative heuristic MEC resource allocation algorithm to solve the problem of multi-user computing offloading in [11]. However, the above articles only consider the scenario of collaborative task processing between a single MEC server and a remote cloud server, ignoring the collaboration among multiple MEC servers. Therefore, it is necessary to consider the collaboration among multiple MEC servers to allocate reasonable computing resources to users, thereby improving the quality of user experience [12,13]. Aiming at the problem of task offloading in vehicle edge computing networks, [14] proposed a load balancing scheme. Through the collaboration among multiple MEC servers and the cloud servers, tasks are jointly processed, thereby reducing task processing delay.

As one of the rapidly developing technologies, edge caching is getting more and more attention from people [15]. The edge caching technology can store different types of services that users require on MEC servers during off-peak periods, thereby reducing both the delay and energy consumption of user tasks

[16,17]. Recently, several studies have proposed to use edge caching technology in the MEC system to minimize task processing delay or energy consumption [18–20]. In [18], the authors proposed a cooperative content placement problem to minimize the delivery delay of location-based content and the service cost of two types of content. The authors proposed an energy-efficient task caching and offloading scheme in [19]. This scheme takes into account both the resource utilization of MEC servers and user experience. By storing part or all of the task data on the MEC server can improve task execution efficiency. [20] integrated three computing layers of vehicle, network edge and high-altitude platform station to build an intelligent transportation system framework. By considering the computing offloading strategy of the vehicle, planning the caching strategy of the basic data at the edge of the network, and coordinating resource scheduling to improve the delay performance of the application.

Due to the limited storage capacity of MEC server, it must store the services required by users while meeting the capacity constraints. This makes us still face huge challenges when solving the above problems. Some current research mainly focuses on a single MEC server for caching services to minimize task processing delay [21]. However, the research on mutual collaborative caching service among MEC servers is not deep enough. Therefore, we need to design a method for multiple MEC servers to collaborative service caching to meet task requests.

Against this background, we proposes a multiple servers collaborative task processing model. The aim is to minimize system delay and cost by making optimal offloading decisions and computing resources for tasks. The main contributions of this paper are as follows:

1) Under the constraint of satisfying the long-term average system cost, the system delay is minimized by making optimal offloading decisions and computing resources for tasks.
2) To solve the above problems, this paper based on Lyapunov optimization proposes an online algorithm(OKMTA), which is executed in an online manner without predicting future information.
3) By using the Lagrange multiplier method to solve the computing resource allocation problem of tasks, and by using matching theory to solve the offloading decision problem of tasks.
4) The simulation results show that the scheme proposed in this paper not only achieves the near-optimal delay effect, but also keeps the system cost low.

The rest of the paper is organized as follows: Section 2 presents the system model and formulates the optimization problem. Section 3 designs the OKMTA algorithm to solve the optimization problem. Section 4 compares the performance through simulation results. Finally, Sect. 5 concludes the paper.

2 System Model and Problem Formulation

In this section, we will introduce the system model. Suppose the whole network consists of one remote cloud server, multiple edge nodes (EN) and multiple

users (as shown in Fig. 1). Suppose these users have many different types of task requests that need to be executed in the whole network. For each user's task request, it may need the execution of several services, which are stored on ENs and cloud servers. Suppose the cloud server has all services for all task requests, while for each EN, it randomly has part of these services. Based on the location feature, each user can only communicate with its associated EN, while ENs can communicate with each other. This means that when a user sends a task request to its associated EN, the required services may not be stored on this EN. Then this EN may need to send this request to other ENs or request downloading services from the cloud server. Our optimization goal is to minimize the system delay and system cost by formulating optimal offloading decisions and computing resources for task requests. Next, we will introduce the network and service models, communication model, computing model and system cost model in this paper, respectively.

2.1 Network and Service Models

Suppose there are N ENs in the whole network, and denote $\mathcal{N}$ as the set of ENs. For each $EN_j(\in \mathcal{N})$, denote B_j, F_j and G_j as the communication resource, the calculation capability, and the storage capacity respectively. Suppose there are K services in the whole network, and denote $\mathcal{K}$ as the set of services. Denote c_k as the required storage space for service k. When initializing, each EN will randomly download services from the cloud server. Denote $\mathcal{K}_j$ as the service set of EN_j, we have

$$\sum_{k \subseteq \mathcal{K}_j} c_k \leq G_j. \tag{1}$$

Suppose there are M users in the network model, and denote $\mathcal{M}$ as the set of users. For each user $i(\in \mathcal{M})$, it may have a computationally intensive task needed to be processed during the scheduling time. Denote U_i as the task for user i, and we use the quadruple $\{D_i, C_i, K_i, T_i^{max}\}$ to represent resource requirements of U_i, where D_i is the workload data volume, C_i is the required computing resources, K_i is the set of required services, and T_i^{max} is the threshold of time delay.

2.2 Communication Model

Suppose we use the Orthogonal Frequency Division Multiple Access (OFDMA) technology for wireless communication, thus we can ignore the interference between users. Denote $R_{i,j}$ as the uplink transmission rate between user i and EN_j, we have

$$R_{i,j} = B_j log_2(1 + \frac{p_{i,j} h_{i,j}}{\sigma^2}), \tag{2}$$

where B_j is the available spectrum bandwidth, $p_{i,j}$ is the uplink transmission power, σ^2 is the noise power and $h_{i,j}$ is the state of the uplink channel between

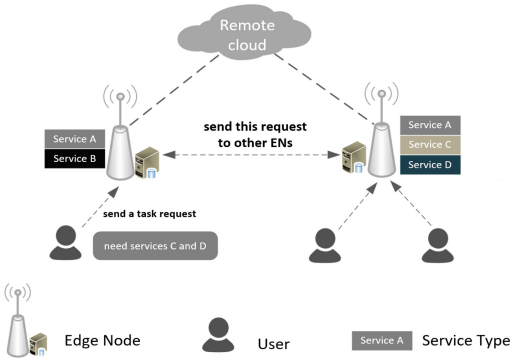

Fig. 1. Multi-ENs collaborative task processing model

user i and EN_j. Denote $T_{i,j}^{up}$ as the uplink transmission delay for task U_i to EN_j, we have

$$T_{i,j}^{up} = \frac{D_i}{R_{i,j}}. \tag{3}$$

In our scheduling scenario, we only consider the uplink transmission delay and thus ignore the downlink transmission delay.

Denote $x_{i,j}^t \in \{0,1\}$ as a binary decision variable on whether task U_i is executed on associated EN_j in time slot t. $x_{i,j}^t = 1$ means task U_i is executed on the associated EN_j, otherwise $x_{i,j}^t = 0$.

Denote $z_{i,j,j'}^t \in \{0,1\}$ as a binary decision variable on whether task U_i is executed on other $EN_{j'}$ in time slot t. $z_{i,j,j'}^t = 1$ means that task U_i is sent to $EN_{j'}$ for execution, otherwise $z_{i,j,j'}^t = 0$. When the user i sends task request to EN_j, EN_j needs to send a task request to $EN_{j'}$ if EN_j doesn't have the required service k. Based on this situation, $x_{i,j}^t = 0$ and $z_{i,j,j'}^t = 1$.

Denote $R_{i,j,j'}^{round}$ as the transmission rate of the task U_i between EN_j and $EN_{j'}$. Denote $T_{i,j,j'}^{round}$ as the round trip delay of task U_i between EN_j and $EN_{j'}$, we have

$$T_{i,j,j'}^{round} = \frac{D_i}{R_{i,j,j'}^{round}}. \tag{4}$$

When EN_j executes task U_i, if the required service k is missing, it needs to download this service from the cloud server to EN_j. Denote $R_{c,j}^{down}$ as the download rate between the cloud server and EN_j. Denote $T_{c,j}^{down}$ as the download delay for the required set of services from the cloud server to EN_j. We have

$$T_{c,j}^{down} = \sum_{k \in K_i \cap k \notin K_j} \frac{c_k}{R_{c,j}^{down}}. \tag{5}$$

Thus the total transmission delay mainly consists of three parts: uplink transmission delay, round trip delay and download delay. Denote T_i^{trans} as the total transmission delay, we have

$$T_i^{trans} = T_{i,j}^{up} + x_{i,j}^t T_{c,j}^{down} + \sum_{j' \in \mathcal{N}} z_{i,j,j'}^t (1 - x_{i,j}^t)(T_{i,j,j'}^{round} + T_{c,j'}^{down}). \quad (6)$$

2.3 Computing Model

When user i sends task U_i to EN_j, EN_j will allocate computing resources for user i. Denote $f_{i,j}^{mec}$ as the computing resources allocated by EN_j to user i. Since the computing resources allocated to users by each EN cannot exceed the total computing resources of the EN, each EN needs to satisfy

$$\sum_{i=1}^{M_j} f_{i,j}^{mec} \leq F_j, \quad (7)$$

where M_j is all users who execute tasks in EN_j. Denote $T_{i,j}^{exe}$ as the computing delay for EN_j to execute task U_i, we have

$$T_{i,j}^{exe} = \frac{C_i}{f_{i,j}^{mec}}. \quad (8)$$

Denote T_i^{exe} as the computing delay for EN to execute task U_i, we have

$$T_i^{exe} = x_{i,j}^t T_{i,j}^{exe} + \sum_{j' \in \mathcal{N}} z_{i,j,j'}^t (1 - x_{i,j}^t) T_{i,j,j'}^{exe}. \quad (9)$$

The total delay mainly consists of transmission delay and computing delay. Thus denote T_i^t as the total delay for EN to execute task U_i, we have

$$T_i^t = T_i^{trans} + T_i^{exe} \quad (10)$$

2.4 System Cost Model

User renting EN resources to execute task will incur additional costs, mainly including communication, computing, and storage costs. The communication cost is the overhead generated by the bandwidth resource used for data transmission between the user and the EN, determined by the transmission energy consumption. The computing cost is the overhead incurred in executing task in EN, determined by the amount of task data. The storage cost is the cost of using EN and remote cloud storage services, which is determined by the storage resources occupied by the services. Furthermore, denote Q_i^{avg} as the long-term average system cost for user i. Denote Q_i^t as the system cost incurred by user i renting the resource of EN in time slot t. We have

$$Q_i^t = \frac{\alpha_j p_{i,j} D_i}{B_j log_2(1 + \frac{p_{i,j} h_{i,j}}{\sigma^2})} + \beta_j C_i + \sum_{k \subseteq K_i} \nu_j c_k + \sum_{k \subseteq K_i} \mu_c c_k, \quad (11)$$

where α_j and β_j are the unit cost of energy consumption and task execution in EN_j respectively. ν_j and μ_c are the unit cost of the storage service in EN_j and cloud server respectively.

2.5 Problem Formulation

In this section, we formulate multiple ENs collaborative processing and resource allocation problem for tasks. Minimize system delay and keep cost low by designing optimal solution. Based on the previous discussion, we can express the problem as follows:

$$
\begin{aligned}
\mathbf{P1}: & \min_{x^t, z^t, f^t} \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} (\sum_{i=1}^{M} T_i^t) \\
\mathbf{s.t.}\ & C1: \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} Q_i^t \le Q_i^{avg} \\
& C2: T_i^t \le T_i^{max} \\
& C3: \sum_{k\subseteq\mathcal{K}_j} c_k \le G_j \\
& C4: f_{i,j}^{mec} > 0 \\
& C5: \sum_{i=1}^{M_j} f_{i,j}^{mec} \le F_j \\
& C6: x_i^t \in \{0,1\} \\
& C7: z_{i,j,j'}^t \in \{0,1\}
\end{aligned}
\tag{12}
$$

Constraint C1 represents the long-term average system cost threshold for each user. Constraint C2 is the maximum threshold constraint imposed on the total delay of executing tasks. Constraint C3 is the constraint that the storage space of EN is limited. Constraint C4 and C5 are the computing resource constraints allocated by EN to tasks. Constraints C6 and C7 represent binary offloading decision variable constraints that determine whether the task is executed on the associated EN or $EN_{j'}$ respectively.

Through the analysis of the above problems, we can conclude that **P1** is a mixed-integer nonlinear programming problem. It contains constraints of two discrete binary variables and continuous variables C4-C5, making the problem an NP-hard problem. Therefore, it is impractical to solve **P1** directly. In addition, solving **P1** requires a large amount of future network state information, but predicting such future information is difficult or even impossible. Therefore, in the absence of complete information, it is necessary to propose an online optimization algorithm to solve the problem.

3 An Online Optimization Algorithm

To solve **P1** efficiently, we propose an online distributed optimization algorithm (OKMTA for short). The algorithm mainly includes two parts: (1). Based on Lyapunov optimization, we propose an online algorithm to reformulate **P1**, decomposing the long-term optimization problem into a series of one-slot optimization

problems. (2). For the one-slot optimization problem, we propose a distributed solution based on Lagrange multiplier method and matching theory.

3.1 Restate the Problem by Lyapunov Optimization

The idea of Lyapunov optimization is to decompose an optimization problem with long-term constraints into each time slot. And then we can directly optimize the problem in each time slot while ensuring the stability of the system [22]. Denote $q(t) = \{q_1(t), q_2(t), ..., q_i(t)\}$ as a virtual cost queue and set the initial value to 0 for solving the long-term average cost constraint. The virtual cost queue constructed for user i can be expressed as

$$q_i(t+1) = max\{q_i(t) + Q_i^t - Q_i^{avg}, 0\}, \tag{13}$$

where $q_i(t)$ is the queue backlog of time slot t, representing the deviation between the system cost and the threshold under the current time slot t.

To stabilize $q(t)$, we denote $L(q(t)) = \frac{1}{2}\sum_{i=1}^{M} q_i^2(t)$ as quadratic Lyapunov function, representing the "congestion level" of the virtual cost queue length [23]. A small value of $L(q(t))$ means that the backlog of all virtual cost queues is small, so that the stability of all queues can be achieved. However, it is impractical to keep $L(q(t))$ small all the time. Therefore, to solve this problem, we further denote $\Delta(q(t))$ as a one-slot Lyapunov drift to keep the Lyapunov function low and thus keep the queue stable. we have

$$\Delta(q(t)) = \mathbb{E}[L(q(t+1)) - L(q(t))|q(t)], \tag{14}$$

where $\Delta(q(t))$ represents the change of the virtual cost queue within a single time slot in the Lyapunov function. We define a drift-plus-penalty function that integrates system delay and virtual cost queue stability via (15).

$$\Delta(q(t)) + V\mathbb{E}[\sum_{i=1}^{M} T_i^t|q(t)], \tag{15}$$

where V is a non-negative parameter controlling the trade-off between system delay and cost. The following **Lemma** 1 provides an upper bound on the drift plus penalty function.

Lemma 1: For all possible optimization variables satisfying C2-C7, there is an upper bound for the drift-plus-penalty function [24], as shown in (16). Where B is a constant, $B = \frac{1}{2}\sum_{i=1}^{M}(Q_i^{max} - Q_i^{avg})^2$. Where Q_i^{max} is the upper bound of the one slot virtual cost for user i.

$$\Delta(q(t)) + V\mathbb{E}[\sum_{i=1}^{M} T_i^t|q(t)] \leq B + V\mathbb{E}[\sum_{i=1}^{M} T_i^t|q(t)] + \mathbb{E}[\sum_{i=1}^{M} q_i(t)(Q_i^t - Q_i^{avg})|q(t)] \tag{16}$$

According to **Lemma** 1, our goal is to minimize the expression on the right side of (16) to determine the upper bound of the drift-plus-penalty function. This will help us transform the long-term optimization problem **P1** into a one-slot optimization problem **P2**. The offloading decision and resource allocation problems of tasks are determined by solving **P2**.

$$\mathbf{P2}: \min_{x^t, z^t, f^t} \sum_{i=1}^{M} VT_i^t + q_i(t)Q_i^t \tag{17}$$

Algorithm 1 describes the implementation process of OKMTA algorithm. The algorithm runs online, using real-time network information and the current backlog status of the queue as input, and solves **P1** by solving the approximate optimal solution of **P2**. At each time slot t, by solving for **P2** we can obtain the values of the variables x^t, z^t, f^t. At the same time, each virtual cost queue is also updated for calculation in the next time slot.

Algorithm 1: OKMTA Algorithm

Input: $\mathcal{M}$: the set of users, $\mathcal{N}$: the set of ENs, task U_i, the set of services, non-negative parameter V and q(0)=0

Output: The offloading decisions x^t, z^t and the computing resource f^t for task for each time slot t

```
for t = 0 to T-1 do
    According to Algorithm 2 and 3, the values of x^t, z^t, f^t for each time slot t are obtained.
    Update the q(t+1) according to (13).
    t = t + 1.
end
```

We can observe that (17) is a joint optimization problem. They are mutually constrained when the conditions are satisfied, so it is difficult to use mathematical methods to solve **P2**. The offloading decisions x^t and z^t are binary variables, and the computing resource f^t is a continuous variable, so **P2** is a mixed-integer nonlinear programming problem. We can decompose the original problem **P2** into two sub-problems to solve: (1). Under the given conditions of x^t and z^t, use the Lagrange multiplier method to solve the allocation problem of computing resources f^t. (2). On the basis of obtaining the f^t solution, the optimal solutions of x^t and z^t are obtained by bilaterally matching the task set and the EN set. By alternately solving these two subproblems, we can gradually optimize **P2** and find the optimal solution finally.

3.2 Distributed Optimization Algorithm in Each Time Slot

The Optimal Computing Resource Allocation. When user i sends a task U_i to EN, assuming x^t and z^t are already given, we will ignore variables

unrelated to f^t in the optimization objective function. Therefore, we can express the optimization problem for allocating computing resources to tasks as follows:

$$\begin{aligned} \mathbf{P3}: \min_{f^t} F_1(f_{i,j}^{mec}) &= \sum_{i=1}^{M} \frac{C_i}{f_{i,j}^{mec}} \\ \mathbf{s.t.}\ C4: f_{i,j}^{mec} &> 0 \\ C5: \sum_{i=1}^{M_j} f_{i,j}^{mec} &\leq F_j \end{aligned} \tag{18}$$

In order to prove that **P3** is a convex problem, we take the second derivative of $F_1(f_{i,j}^{mec})$ with respect to $f_{i,j}^{mec}$. we have

$$\frac{\partial^2 F_1(f_{i,j}^{mec})}{\partial (f_{i,j}^{mec})^2} = \frac{2C_i}{(f_{i,j}^{mec})^3}. \tag{19}$$

Due to constraint C4, we have $\frac{\partial^2 F_1(f_{i,j}^{mec})}{\partial (f_{i,j}^{mec})^2} > 0$, so **P3** is a convex problem. Therefore, the Lagrangian function can be used to solve **P3**. We have

$$L(f, \lambda) = \sum_{i=1}^{M} \frac{C_i}{f_{i,j}^{mec}} + \lambda (\sum_{i=1}^{M_j} f_{i,j}^{mec} - F_j), \tag{20}$$

where λ is a non-negative Lagrangian multiplier associated with the computing resource constraints of EN. Then we use the Karush-Kuhn-Tucker (KKT) condition to obtain the optimal computing resource allocation f^* [25]. Thus the partial derivative of (20) with respect to $f_{i,j}^{mec}$ is given as

$$\frac{\partial L(f, \lambda)}{\partial f_{i,j}^{mec}} = -\frac{C_i}{(f_{i,j}^{mec})^2} + \lambda. \tag{21}$$

Based on the KKT condition, we can obtain the approximate optimal solution of (21), we have

$$f^* = \sqrt{\frac{C_i}{\lambda}}. \tag{22}$$

Then we can obtain the optimal computing resource allocation scheme f^* based on Algorithm 2.

The Optimal Task Offloading Decisions. Then we need to further transform **P2** into a problem of how to optimize the offloading decisions of task to reduce system delay and cost when the computing resource f^* is determined. Therefore, we can express the problem as follows:

$$\begin{aligned} &\mathbf{P4}: \min_{x^t, z^t} F_2(x_{i,j}^t, z_{i,j,j'}^t) \\ &\mathbf{s.t.}\ C2 - C7 \end{aligned} \tag{23}$$

Algorithm 2: Optimal Computing Resource Allocation Algorithm

Input: $\mathcal{M}$: the set of users, $\mathcal{N}$: the set of ENs, task U_i, Maximum tolerance $\zeta = 1 \times 10^{-8}$, $\lambda_{min} = 0$ and $\lambda_{max} = 1$

Output: The optimal computing resource allocation f^*

for *all users* $i \in \mathcal{M}$ **do**
  while $\lambda_{max} - \lambda_{min} > \zeta$ **do**
    Denote $\lambda = (\lambda_{max} + \lambda_{min})/2$.
    Compute f^* according to substitute λ into (22).
    If $\sum_{i=1}^{M_j} f_{i,j}^{mec} < F_j$, update $\lambda_{max} = \lambda$, otherwise update $\lambda_{min} = \lambda$
  end
  f^* can be obtained by substituting λ into (22).
end

$$F_2(x_{i,j}^t, z_{i,j,j'}^t) = \sum_{i=1}^{M} V[T_{i,j}^{up} + x_{i,j}^t(\frac{C_i}{f_{i,j}^*} + T_{c,j}^{down}) + \sum_{j' \in \mathcal{N}} z_{i,j,j'}^t(1 - x_{i,j}^t)(\frac{C_i}{f_{i,j'}^*} + T_{i,j,j'}^{round} + T_{c,j'}^{down})] + q_i(t)Q_i^t \tag{24}$$

To effectively solve **P4**, we propose a task offloading algorithm based on matching theory. This algorithm views the problem of task offloading as a two-sided many-to-one matching problem, aiming to find a stable optimal matching between task set and EN set. By establishing a stable two-sided matching, it can ensure that the task in the current matching can obtain the maximum system utility. When no "blocking pairs" are found during the matching process, the matching is considered stable [26].

In this paper, user tends to offload task to EN with required services and sufficient computing resources, while EN tends to process tasks with less demanding computing resources. To simplify the description, we denote $\mathcal{U}$ as a set of tasks and denote $\mathcal{N}$ as a set of ENs, which are two disjoint players. Then we need to match $\mathcal{U}$ and $\mathcal{N}$. Tasks can only select one EN for processing, and each EN can process multiple tasks. Denote $H(x)$ as the matching function between $U_i \in \mathcal{U}$ and $EN_j \in \mathcal{N}$. It needs to satisfy the following conditions: (1). $H(U_i) \in \mathcal{N}, \forall U_i \in \mathcal{U}$ means that any task matches any EN in the set $\mathcal{N}$. (2). $H(EN_j) \in \mathcal{U}, \forall j \in \mathcal{N}$ means that $\mathcal{U}$ contains any task processed by any EN in the set $\mathcal{N}$. (3). $H(EN_j) = U_i \Leftrightarrow H(U_i) \in EN_j$ means that the task U_i to EN_j processing is equivalent to EN_j processing of tasks including U_i, but not only processing a task U_i. If there is a current matching H, the offloading decision of the task can be determined as

$$x_{i,j}^t = \begin{cases} 1\ , & \text{if } H(EN_j) = U_i; \\ 0\ , & \text{otherwise.} \end{cases} \tag{25}$$

$$z_{i,j,j'}^{t} = \begin{cases} 1\ , & \text{if } H(EN_{j'}) = U_i \text{ and } x_{i,j}^{t} = 0,\ j \neq j'; \\ 0\ , & \text{otherwise.} \end{cases} \tag{26}$$

Each player can build a preference list with other players by defining a utility function that is inversely proportional to the sum of the system's total delay and cost [27]. Then the utility function of EN for task U_i can be expressed as

$$SU_{U_i}(EN_j) = \frac{1}{F_2(x_{i,j}^{t}, z_{i,j,j'}^{t})}. \tag{27}$$

If there is the following definition $SU_{U_i}(EN_{j_1}) > SU_{U_i}(EN_{j_2})$, it means that the task U_i prefers to be processed in EN_{j_1} rather than in EN_{j_2}.

Algorithm 3: Based on Matching Theory Task Offloading Algorithm

Input: $\mathcal{M}$: the set of users, $\mathcal{U}$: the set of tasks, $\mathcal{N}$: the set of ENs and the set of services

Output: All tasks optimal offloading decisions x^t and z^t

Initialize Phase:
All users and associated ENs are initialized to construct a matching H that satisfies all constraints C2-C7.
Calculate the system utility under the current matching pair according to (27).
Matching Phase:
while *there exists unstable matching* **do**
  if $EN_{j'}$ *exists the services required by* U_i **then**
    Obtain the parameters of $EN_{j'}$.
    Calculate system utility of user i to $EN_{j'}$.
    if $SU_{U_i}(EN_j) < SU_{U_i}(EN_{j'})$ **then**
      $EN_{j'}$ accepts the matching request and calculates the system utility of user i according to (27).
      If constraints C2-C7 can not be satisfied, user i will be re-matched.
    end
    else
      $EN_{j'}$ rejects matching request of user i, and user i sends matching requests to other ENs.
    end
  end
end
The system unility does not change, that is, tasks and ENs achieve an optimal and stable matching H.
According to the matching results and (25)-(26), we can obtain the optimal offloading decisions x^t and z^t of all tasks.

Based on the above definition of matching theory, we propose Algorithm 3 to solve **P4**. The algorithm includes two phases: an initialization phase and a matching phase. In the initialization phase, all users are matched with associated ENs and calculate the system utility. In the matching phase, all users

continuously search for available ENs to process their tasks and establish stable matching relationships with ENs. If task U_i has low system utility with the currently matched EN, user i will send a matching request to other ENs. When other ENs receive this matching request, they calculate the system utility and make corresponding matching decisions: (1). If the EN accepts task U_i to improve the system utility, it accepts the matching request of user i. (2). Otherwise, it rejects matching request of user i. If user i's matching request is rejected by the EN, user i will send a matching request to the next available EN. When no user initiates a matching request in the matching phase, the matching phase ends. During each round of matching between tasks and ENs, both parties can achieve stable matching with the maximum system utility. Finally, according to the matching results satisfying all the constraints, the optimal offloading decisions of all tasks can be obtained.

4 Simulation Results

In this section, we evaluate the performance of the proposed OKMTA algorithm through simulation results.

4.1 Simulation Setting

In the scenario of multi-ENs collaborative task processing, we can set the following parameters for simulation experiments. The coverage area size is 500m × 500m, in which there are 10 users and 4 ENs randomly distributed. The wireless channel gain from user i to EN_j can be expressed as $h_{i,j} = d_{i,j}^{-\vartheta}$, where $d_{i,j}$ denotes the distance between the user i and the associated EN_j, the path loss factor $\vartheta = 4$, the noise power σ^2 is set to -100dBm, and the transmission power for user i is set to $p_{i,j} = 30$dBm [28]. Considering the heterogeneity of EN, the channel bandwidth B_j, storage capacity G_j, and computing resources F_j of EN_j are set to [8, 12] MHz, [5, 10]GB, [30, 60]GHz respectively. The input data size D_i of the task U_i is set to [30, 50]MB [29]. The number of CPU cycles required to execute each bit task is set to 1200. To process tasks, various resources of EN need to be rented. The communication unit cost α_j and the execution unit cost β_j of the task are set to $[2 \times 10^{-4}, 3 \times 10^{-4}]$ unit/J, $[0.4 \times 10^{-7}, 0.8 \times 10^{-7}]$ unit/bit respectively. The unit cost of the storage service on the remote cloud server and EN is set to [1.0, 1.2] unit and [0.54, 0.6] unit respectively [30].

Next, we conduct a performance research on the OKMTA algorithm proposed in this paper. We use 500 time slots for evaluation, and the time interval of each time slot is set to 10ms. Additionally, we introduce three benchmark algorithms to evaluate the algorithmic performance of OKMTA algorithm: (1). Non-Collaborative EN Processing Algorithm (NCEPA): ENs within the coverage area process tasks independently, and there is no collaborative relationship between them. Tasks can only be processed in their associated EN. (2). Optimal System Delay Processing Algorithm (OSDPA): When processing tasks, it only considers reducing system delay, without considering system cost constraints. (3). Random Processing Algorithm: tasks can be randomly offloaded to any EN for processing.

4.2 Performance Comparison

Figure 2 and 3 respectively show the average delay and cost performance of 4 ENs processing 10 tasks in the case of V = 4. As can be seen from the following two figures, the OKMTA algorithm achieves near-optimal delay performance while satisfying the long-term cost constraint. Although the OSDPA algorithm performs best in terms of delay performance, it can be clearly seen in Fig. 3 that using the OSDPA algorithm for task processing results in higher costs. In comparison, the random algorithm assigns tasks randomly to any EN without considering specific resource requirements, leading to inferior delay and cost performance. In addition, in the NCEPA algorithm, EN processes task requests sent by users independently, without considering the mutual collaboration relationship between ENs, resulting in poor overall system delay performance.

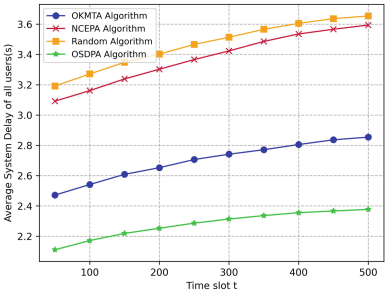

Fig. 2. Average system delay of all users

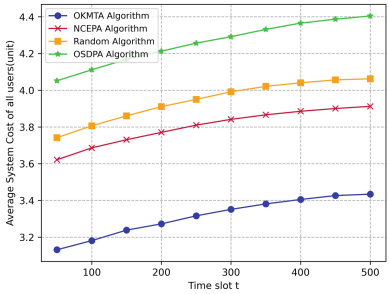

Fig. 3. Average system cost of all users

4.3 The Effect of Parameter V on System Performance

Figure 4 and 5 show the average system delay and cost of processing 10 tasks for OKMTA and OSDPA algorithms with different parameters V, respectively. It can be seen from Fig. 4 that as the parameter V increases, the average system delay of the OKMTA algorithm decreases significantly, and gradually approaches the optimal processing delay of the system. It can be seen from Fig. 5 that with the increase of the parameter V, the average system cost of the OKMTA algorithm gradually increases, and finally remains at a level lower than the cost of OSDPA algorithm. But the OSDPA algorithm achieves the best system delay at a higher system cost. The above two figures verify that the parameter V can flexibly adjust the trade-off between system delay and cost, and different parameters V can be set according to different situations to meet the needs of the system. Considering the long-term cost constraints, our algorithm outperforms the OSDPA algorithm.

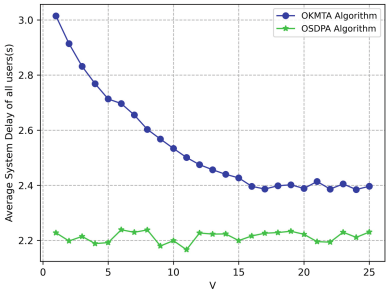

Fig. 4. The impact of different V

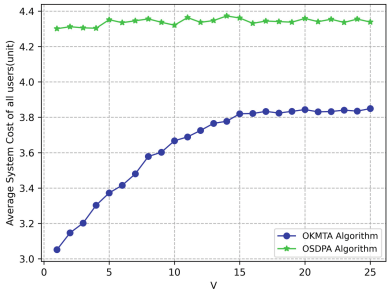

Fig. 5. The impact of different V

4.4 The Effect of the Number of Users on System Performance

Figure 6 and 7 show the total system delay and cost when using different schemes to process tasks with different numbers of users, respectively. As the number of tasks increases, system delay and cost increase for all scenarios. Specifically, the OSDPA algorithm achieves the lowest system delay, followed by our OKMTA algorithm with slightly higher system delay, while the NCEPA and random algorithms result in the highest system delay. Additionally, from Fig. 7, it can be observed that our algorithm has the lowest system cost. Although the OSDPA algorithm maintains low system delay, its cost is the highest, indicating that this algorithm consumes significant system cost to achieve the best system delay. In summary, under the condition that the constraints are satisfied, our algorithm closely approaches optimal system delay while maintaining lower system cost.

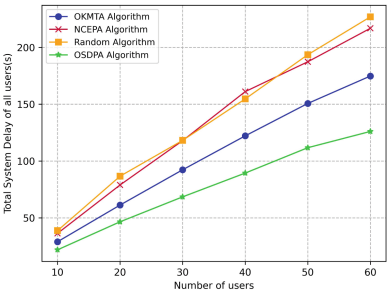

Fig. 6. System delay with different M

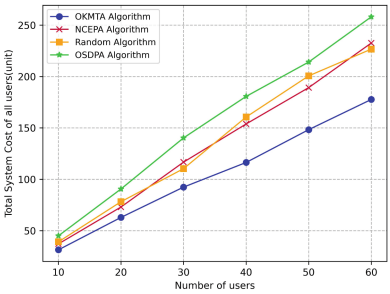

Fig. 7. System cost with different M

5 Conclusion

In this paper, we propose a multi-ENs collaborative task processing model. It mainly studies the collaborative task processing among multiple ENs, making optimal offloading decisions and computing resources for tasks to minimize system delay and cost. To solve this problem, we propose an online optimization

algorithm (i.e., OKMTA), which is executed in an online manner without predicting future information. The algorithm is mainly used to jointly optimize the offloading decision and computing resource allocation of tasks, and obtain a near-optimal solution. Finally, the simulation results show that our algorithm not only achieves the near-optimal delay effect, but also keeps the system cost low.

References

1. Elgendy, I.A., Zhang, W.-Z., Zeng, Y., He, H., Tian, Y.-C., Yang, Y.: Efficient and secure multi-user multi-task computation offloading for mobile-edge computing in mobile IoT networks. IEEE Trans. Netw. Serv. Manage. **17**(4), 2410–2422 (2020). https://doi.org/10.1109/TNSM.2020.3020249
2. El Haber, E., Nguyen, T.M., Assi, C.: Joint optimization of computational cost and devices energy for task offloading in multi-tier edge-clouds. IEEE Trans. Commun. **67**(5), 3407–3421 (2019). https://doi.org/10.1109/TCOMM.2019.2895040
3. Zhao, M., et al.: Energy-aware task offloading and resource allocation for time-sensitive services in mobile edge computing systems. IEEE Trans. Veh. Technol. **70**(10), 10925–10940 (2021). https://doi.org/10.1109/TVT.2021.3108508
4. Li, Q., Wang, S., Zhou, A., Ma, X., Yang, F., Liu, A.X.: QoS driven task offloading with statistical guarantee in mobile edge computing. IEEE Trans. Mob. Comput. **21**(1), 278–290 (2022). https://doi.org/10.1109/TMC.2020.3004225
5. Chen, Y., Zhang, N., Zhang, Y., Chen, X., Wu, W., Shen, X.S.: TOFFEE: task offloading and frequency scaling for energy efficiency of mobile devices in mobile edge computing. IEEE Trans. Cloud Comput. **9**(4), 1634–1644 (2021). https://doi.org/10.1109/TCC.2019.2923692
6. Zhou, T., Yue, Y., Qin, D., Nie, X., Li, X., Li, C.: Mobile device association and resource allocation in HCNs with mobile edge computing and caching. IEEE Syst. J. **17**(1), 976–987 (2023). https://doi.org/10.1109/JSYST.2022.3157590
7. Ren, J., Yu, G., Cai, Y., He, Y.: Latency optimization for resource allocation in mobile-edge computation offloading. IEEE Trans. Wireless Commun. **17**(8), 5506–5519 (2018). https://doi.org/10.1109/TWC.2018.2845360
8. Chen, Y., Zhang, N., Zhang, Y., Chen, X., Wu, W., Shen, X.: Energy efficient dynamic offloading in mobile edge computing for internet of things. IEEE Trans. Cloud Comput. **9**(3), 1050–1060 (2021). https://doi.org/10.1109/TCC.2019.2898657
9. Ren, J., Yu, G., He, Y., Li, G.Y.: Collaborative cloud and edge computing for latency minimization. IEEE Trans. Veh. Technol. **68**(5), 5031–5044 (2019). https://doi.org/10.1109/TVT.2019.2904244
10. Kai, C., Zhou, H., Yi, Y., Huang, W.: Collaborative cloud-edge-end task offloading in mobile-edge computing networks with limited communication capability. IEEE Trans. Cogn. Commun. Netw. **7**(2), 624–634 (2021). https://doi.org/10.1109/TCCN.2020.3018159
11. Dai, Y., Xu, D., Maharjan, S., Zhang, Y.: Joint computation offloading and user association in multi-task mobile edge computing. IEEE Trans. Veh. Technol. **67**(12), 12313–12325 (2018). https://doi.org/10.1109/TVT.2018.2876804
12. Xu, X., et al.: Secure service offloading for internet of vehicles in SDN-enabled mobile edge computing. IEEE Trans. Intell. Transp. Syst. **22**(6), 3720–3729 (2021). https://doi.org/10.1109/TITS.2020.3034197

13. Zhou, J., Zhang, X.: Fairness-aware task offloading and resource allocation in cooperative mobile-edge computing. IEEE Internet Things J. **9**(5), 3812–3824 (2022). https://doi.org/10.1109/JIOT.2021.3100253
14. Zhang, J., Guo, H., Liu, J., Zhang, Y.: Task offloading in vehicular edge computing networks: a load-balancing solution. IEEE Trans. Veh. Technol. **69**(2), 2092–2104 (2020). https://doi.org/10.1109/TVT.2019.2959410
15. Xia, X., et al.: OL-MEDC: an online approach for cost-effective data caching in mobile edge computing systems. IEEE Trans. Mob. Comput. **22**(3), 1646–1658 (2023). https://doi.org/10.1109/TMC.2021.3107918
16. Zhang, F., Han, G., Liu, L., Martinez-Garcia, M., Peng, Y.: Joint optimization of cooperative edge caching and radio resource allocation in 5G-enabled massive IoT networks. IEEE Internet Things J. **8**(18), 14156–14170 (2021). https://doi.org/10.1109/JIOT.2021.3068427
17. Song, C., Xu, W., Wu, T., Yu, S., Zeng, P., Zhang, N.: QoE-driven edge caching in vehicle networks based on deep reinforcement learning. IEEE Trans. Veh. Technol. **70**(6), 5286–5295 (2021). https://doi.org/10.1109/TVT.2021.3077072
18. Chen, J., Wu, H., Yang, P., Lyu, F., Shen, X.: Cooperative edge caching with location-based and popular contents for vehicular networks. IEEE Trans. Veh. Technol. **69**(9), 10291–10305 (2020). https://doi.org/10.1109/TVT.2020.3004720
19. Gupta, D., Moudgil, A., Wadhwa, S., Solanki, V.: Efficient data caching and computation offloading strategy for edge network. In: 2022 International Conference on Emerging Smart Computing and Informatics (ESCI), Pune, India, pp. 1–5 (2022). https://doi.org/10.1109/ESCI53509.2022.9758379
20. Ren, Q., Abbasi, O., Kurt, G.K., Yanikomeroglu, H., Chen, J.: Caching and computation offloading in high altitude platform station (HAPS) assisted intelligent transportation systems. IEEE Trans. Wireless Commun. **21**(11), 9010–9024 (2022). https://doi.org/10.1109/TWC.2022.3171824
21. Ning, Z., et al.: Intelligent edge computing in internet of vehicles: a joint computation offloading and caching solution. IEEE Trans. Intell. Transp. Syst. **22**(4), 2212–2225 (2021). https://doi.org/10.1109/TITS.2020.2997832
22. Tang, C., Zhu, C., Wu, H., Li, Q., Rodrigues, J.J.: Toward response time minimization considering energy consumption in caching-assisted vehicular edge computing. IEEE Internet Things J. **9**(7), 5051–5064 (2022). https://doi.org/10.1109/JIOT.2021.3108902
23. Xia, X., Chen, F., He, Q., Grundy, J., Abdelrazek, M., Jin, H.: Online collaborative data caching in edge computing. IEEE Trans. Parallel Distrib. Syst. **32**(2), 281–294 (2021). https://doi.org/10.1109/TPDS.2020.3016344
24. Chen, W., Wang, D., Li, K.: Multi-user multi-task computation offloading in green mobile edge cloud computing. IEEE Trans. Serv. Comput. **12**(5), 726–738 (2019). https://doi.org/10.1109/TSC.2018.2826544
25. Zhao, J., Li, Q., Gong, Y., Zhang, K.: Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks. IEEE Trans. Veh. Technol. **68**(8), 7944–7956 (2019). https://doi.org/10.1109/TVT.2019.2917890
26. Chen, D., et al.: Matching-theory-based low-latency scheme for multitask federated learning in MEC networks. IEEE Internet Things J. **8**(14), 11415–11426 (2021). https://doi.org/10.1109/JIOT.2021.3053283
27. Wu, H., et al.: Delay-minimized edge caching in heterogeneous vehicular networks: a matching-based approach. IEEE Trans. Wireless Commun. **19**(10), 6409–6424 (2020). https://doi.org/10.1109/TWC.2020.3003339

28. Feng, H., Guo, S., Yang, L., Yang, Y.: Collaborative data caching and computation offloading for multi-service mobile edge computing. IEEE Trans. Veh. Technol. **70**(9), 9408–9422 (2021). https://doi.org/10.1109/TVT.2021.3099303
29. Xu, J., Chen, L., Zhou, P.: Joint service caching and task offloading for mobile edge computing in dense networks. In: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, Honolulu, HI, USA, pp. 207–215 (2018). https://doi.org/10.1109/INFOCOM.2018.8485977
30. Zhao, J., Sun, X., Li, Q., Ma, X.: Edge caching and computation management for real-time internet of vehicles: an online and distributed approach. IEEE Trans. Intell. Transp. Syst. **22**(4), 2183–2197 (2021). https://doi.org/10.1109/TITS.2020.3012966