Contents lists available at ScienceDirect

Future Generation Computer Systems

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

System delay optimization for Mobile Edge Computing

Surong Xiao [a], Chubo Liu [a], Kenli Li [a,*], Keqin Li [b]

[a] College of Information Science and Engineering, Hunan University, Hunan 410082, China
[b] Department of Computer Science, State University of New York, New Paltz, NY 12561, USA

ARTICLE INFO

Article history:
Received 22 April 2019
Received in revised form 7 February 2020
Accepted 10 March 2020
Available online 16 March 2020

Keywords:
Mobile edge computing
Matching theory
Resource allocation
Task offloading scheduling

ABSTRACT

Mobile edge computing (MEC) has emerged as an effective paradigm that delivers cloud services and functions to edge devices, with the objective to further enhance quality of service (QoS) of terminal users by offloading their computation-intensive tasks. In this article, a multi-user and multi-server MEC system is considered and each user can choose one MEC server to execute its computation task. We try to minimize the system delay (i.e., the maximum server delay). The problem is decomposed into task offloading problem and transmit power allocation problem which are solved by matching theory and a heuristic idea, respectively. The experimental results show that the proposed algorithm can not only obtain less delay, but also generate less energy consumption compared with the decomposed computation offloading and resource allocation algorithm, the shortest distance based scheduling algorithm and the random scheduling algorithm, especially when the data amount of tasks is the same but the workload is random.

1. Introduction

Nowadays, the ubiquitous smart phones, tablets, and other mobile devices have become a necessity for people's daily life [1–3]. As a result, all kinds of mobile applications are springing up including online gaming, image processing [4], augmented reality [5,6], and so on [7–9]. Usually, these applications are computation-intensive and energy-intensive for mobile devices. However, to improve computing services of users, satisfying their computation (or energy) demands and addressing them with low latency are necessarily pursued. MEC is an expected technique to mitigate this problem, which tries to enhance service performance by delivering cloud services [10–13] to the proximity of the internet edge devices [14–16]. With MEC, users can offload computation-intensive or energy-intensive tasks to MEC servers for execution.

The efficiency of MEC systems is greatly affected by task offloading decision and resource allocation. Therefore, many researchers have worked on offloading in MEC. Based on the number of servers involved in MEC, the existing researches can be divided into studies on single-server systems and multi-server systems. The single-server system accounts for the majority [17–20]. In [17], the authors considered the resource allocation problem of multiple users with different computing loads sharing one MEC server. The MEC system considered in [18] consists of one MEC server and multiple IoT devices. The authors designed a perturbed Lyapunov function to maximize the network utility. The optimal scheduling problem per slot is solved as a knapsack problem. In [19], mobile applications are executed locally or transmitted to the MEC server with the objective to conserve energy for the mobile device. The delay-optimal computation task scheduling problem of single-server MEC systems is handled by adopting a Markov decision process approach in [20]. However, as mentioned, all the above listed works focus on one MEC server.

There are also some works on multi-server MEC systems [21–23]. In [21], a multi-cell and multi-server system is considered. The joint task offloading and resource allocation problem is studied with the objective of minimizing the task execution delay and energy consumption of users. The authors decomposed the problem into task offloading problem which is settled by convex and quasi-convex optimization techniques, and resource allocation problem which is solved by a heuristic algorithm. In [22], users can offload its task to one MEC server through a heterogenous network. The authors tried to optimize the offloading decisions of users, the power of users and the computation frequency of servers to minimize system overhead. The problem of joint optimization of the radio resources and the computational resources in an MIMO multi-server system is considered in [23].

However, both the studies of single-server and multi-server systems tend to consider the system efficiency issue from the perspective of users [24–26]. Few studies consider the system efficiency from the operators' perspective. Nevertheless, in some cases, we want to complete all the computation tasks in the system with less time. In this way, the servers' computing resources can be devoted to the other computing tasks earlier, and,

* Corresponding author.
E-mail addresses: surong@hnu.edu.cn (S. Xiao), liuchubo@hnu.edu.cn (C. Liu), lkl@hnu.edu.cn (K. Li), lik@newpaltz.edu (K. Li).

https://doi.org/10.1016/j.future.2020.03.028
0167-739X/© 2020 Published by Elsevier B.V.

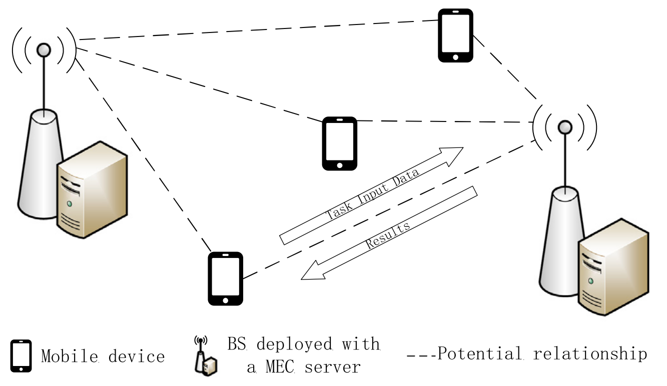


Fig. 1. A MEC system with multiple MEC servers and multiple mobile devices.

naturally, the MEC system can serve more users. Our study is to optimize the latency of MEC system.

In our work, we consider a MEC system with multiple servers and mobile devices (users). Each user comes with a computation task that needs to be offloaded to a MEC server for execution. The system is considered using orthogonal frequency division multiple access (OFDMA) as the access technique. Our problem is to find a task offloading and transmit power allocation scheme for each user, with the objective to minimize the maximum server delay of the system. This problem is found to be a mixed integer non-linear problem (MINLP), which is NP-hard. To solve this problem, we propose two sub-optimal algorithms. The task offloading part adopts the method of matching theory and the transmit power allocation part adopts a heuristic idea.

2. System model

We consider a multi-user and multi-server MEC system. In the system, there are multiple mobile devices such as mobile phone. Each mobile device is regarded as a user. Each user has one computation task, which is expected to be offloaded to a MEC server for processing. There are also multiple MEC servers which are deployed by telecom operators. With a certain amount of memory capacity and computing power, MEC servers can store task input data of users and compute the tasks. Fig. 1 shows a MEC system with two servers and three mobile devices, where the three mobile devices can choose any server to offload their computing tasks. We summarize the primary notations used throughout this paper in Table 1.

2.1. Network model

The MEC servers are deployed at different base stations (BSs). Each server can receive data wirelessly from mobile devices through corresponding BS. As a MEC server usually serves multiple users, the entire spectrum is reused by every BS. We use OFDMA as multiple access scheme. The whole spectrum is divided into N subchannels. Each task is assigned to one subchannel so that uplink transmissions among users who offloaded its task to the same MEC server are orthogonal. The operational frequency band is represented as B. The band of a subchannel denoted as W can be calculated by $W = B/N$. The subchannel set of each BS is represented as $\mathbf{N} = \{1, 2, \dots, N\}$, and we use n stands for the nth subchannel. We suppose the network is quasi-static, that means users do not quit or join midway through the offloading period.

Table 1
The summary of constants and variables in the model.

Expression	Physical meaning and/or reference.
$\mathbf{U}$	The set of users.
U	The number of users.
u	The index of a user.
d_u	The task input data size of task u.
c_u	The amount of CPU circles needed to compute per unit data of task u.
p_u	The transmit power of mobile device u.
$\mathbf{S}$	The set of MEC servers.
$\mathbf{S}_u$	The MEC servers list that can be chosen by user u.
S	The number of MEC servers.
s	The index of a MEC server.
f_s	The work frequency of MEC server s.
$\mathbf{N}$	The set of subchannels for each BS.
N	The number of subchannels for each BS.
n	The index of a subchannel.
$\mathbf{U}_s$	The set of users choosing MEC server s.
$\mathbf{U}^n$	The set of users choosing subchannel n.
B	The operational frequency band.
W	The band of a subchannel.
h_{us}^n	The channel gain for transmitting task u to MEC server s through subchannel n.
p^{max}	The maximum of transmission power.
$\mathbf{A}$	The offloading decision profile of all users.
a_{us}^n	The offloading decision of user u related to MEC server s and subchannel n.
a_{us}	The offloading decision between user u and MEC server s.
$\mathbf{P}$	The transmit power profile of all users.
σ^2	The power spectral density of the background noise.
Φ_s	The sequence of tasks in $\mathbf{U}_s$ by their arrival order.
ϕ_s^i	The task index number of the ith task on MEC server s.
$\mathcal{P}$	The set of transmission power options for users.
L	The number of options in $\mathcal{P}$.

2.2. Computation task and MEC server model

The number of users and MEC servers are represented by U and S respectively. We use set $\mathbf{U} = \{1, 2, \dots, U\}$ to collect all the mobile devices (i.e., U tasks of the devices) in the system and u to denote the uth device (i.e., uth task). For task u, we use d_u for its input data size and c_u for its workload which represents the CPU circles needed to compute per unit data. The value of c_u reflects the nature of the computation task data and can be measured offline. The set of S MEC servers is written as $\mathbf{S} = \{1, 2, \dots, S\}$ and we use s stand for the sth server. For server s, we use f_s to denote its work frequency. Each MEC server can accept multiple tasks and execute the tasks by the order of their arrival time (first come, first execute).

2.3. Communication model

In the MEC system, each user can choose one MEC server to offload its computation task via one of the subchannels of the corresponding BS. We denote the offloading decision profile of users as $\mathbf{A} = \{a_{us}^n | u \in \mathbf{U}, s \in \mathbf{S}, n \in \mathbf{N}\}$. The value of a_{us}^n can only be 0 or 1. When a_{us}^n equals to 1, it means user u chooses to transmit its computation task to MEC server s through subchannel n and a_{us}^n equals to 0 means task u is not transmitted to MEC server s or/and is not transmitted through subchannel n. Since a user can only select one MEC server and one subchannel, we can derive

$$\sum_{s=1}^{S}\sum_{n=1}^{N} a_{us}^{n} = 1,\ \forall u \in \mathbf{U}. \tag{1}$$

For each MEC server, a subchannel carries only one task at a time, so

$$\sum_{u=1}^{U} a_{us}^{n} = 1,\ \forall s \in \mathbf{S},\ n \in \mathbf{N}. \tag{2}$$

Sometimes we use a_{us} to represent the offloading relationship between user u and MEC server s. Likewise, $a_{us} = 1$ means user u offloads its task to MEC server s and $a_{us} = 0$ means user u does not offload its task input data to MEC server s. The value of a_{us} equals to $\sum_{n=1}^{N} a_{us}^{n}$. The number of users that a MEC server can serve at the same time is constrained by

$$\sum_{u=1}^{U} a_{us} \le N,\ \forall s \in \mathbf{S}. \tag{3}$$

The set of users choosing MEC server s is denoted by $U_s = \{u|u \in \mathbf{U}, a_{us} = 1\}$, and the set of users choosing subchannel n by $U^n = \{u|u \in \mathbf{U}, \sum_{s=1}^{S} a_{us}^{n} = 1\}$.

2.4. Task offloading and mobile-edge execution model

To utilize the computation resource of the MEC servers, each user needs to transmit its task to one of the servers. After receiving the data, each MEC server accomplishes the computational process for the users who offload its task to it. The computing order of the tasks is the same as their arrival order. After computing, the MEC servers send the results back to the corresponding mobile devices. As the data size of computation results is relatively smaller, downloading the results can be fast, the time for downloading results is neglected in this research. We focus on the uplink transmission time which is related to the data rate and the data size, and the execution time which has a correlation with task data size, task workload and the execution frequency of the MEC server. Here we give the derivation of the data rate below. For user u, h_{us}^{n} is the channel gain for transmitting its task to MEC server s through subchannel n. The transmit power of mobile device u is written as p_u and $\mathbf{P} = \{p_u|u \in \mathbf{U}\}$ is the transmit power profile of all users. The transmission rate of task u offloaded to MEC server s through subchannel n is

$$R_{us}^{n}(\mathbf{A}, \mathbf{P}) = W \log_2\left(1 + \frac{p_u h_{us}^{n}}{\sigma^2 + \sum_{k\in\mathbf{S},k\neq s}\sum_{j\in U_k} a_{jk}^{n} p_j h_{js}^{n}}\right), \tag{4}$$

where σ^2 represents the power spectral density of the background noise and the second term in the denominator stands for the accumulated inter-channel interference. The transmission rate of user u to MEC server s is given by $R_{us} = \sum_{n=1}^{N} a_{us}^{n} R_{us}^{n}, \forall u \in \mathbf{U}, s \in \mathbf{S}$.

The handling capacity of a MEC system refers to the amount of task data processed per unit time in the system and it reflects the ability of the entire system to handle user tasks. To increase the handling capacity, we need to minimize the system delay caused by completing all tasks. In the next section, we formulate an optimization problem to minimize the system delay by allocating the computation tasks to the MEC servers and adjusting the transmitting power of mobile devices.

3. Problem formulation

In this section, we first analyze the completion process of each task using two time slots and one time point. Then we manage to calculate the delay of the MEC servers. At last we formulate an optimization problem for joint task offloading and transmit power allocation (JTOTPA), with the objective of minimizing the maximum MEC server delay.

The completion of each task goes through three phases: input data transmission, queue wait and server execution. According to the three phases, we define transmission time, execution time (two time slots) and ready time (one time point) for each task. The transmission time is the time needed for the task being transmitted from the mobile device to the MEC server. We use $t_{trans}^{u,s}$ to represent the transmission time for task u offloaded to MEC server s, it can be calculated as

$$t_{trans}^{u,s} = \frac{a_{us} d_u}{R_{us}}. \tag{5}$$

The execution time refers to the duration that the task is executed at the MEC server. $t_{exe}^{u,s}$ represents the execution time for task u processed at MEC server s and is given by

$$t_{exe}^{u,s} = \frac{a_{us} d_u c_u}{f_s}. \tag{6}$$

We refer to the time at which the MEC server begins to execute a task as the task's ready time. A task is ready when (1) its input data has been transmitted to the chosen MEC server (2) the prior tasks on the same MEC server have been completed. The execution order of each MEC server is fixed by the arrival order of the tasks. For MEC server s, we need to work out the transmission time of tasks in U_s, and sort them from small to large. Consequently, we get a sequence of ordered tasks for MEC server s represented as $\Phi_s = [\phi_s^1, \phi_s^2, \dots, \phi_s^{|U_s|}]$, where $|U_s|$ is the number of tasks allocated to MEC server s. We use ϕ_s^i representing the index (in set $\mathbf{U}$) of the ith task executed on MEC server s. The ready time of task u on MEC server s is denoted as $t_{ready}^{u,s}$. So that we can put the ready time of task ϕ_s^i ($\phi_s^i \in U_s$) as

$$t_{ready}^{\phi_s^i,s} = \begin{cases} t_{trans}^{\phi_s^i,s} & i = 1; \\ max\{t_{trans}^{\phi_s^{i-1},s} + t_{exe}^{\phi_s^{i-1},s},\ t_{trans}^{\phi_s^i,s}\} & 1 < i \le |U_s|. \end{cases} \tag{7}$$

Server latency is equal to the summation of the ready time of the last task and its execution time. So the delay of MEC server s is represented as

$$t_{comp}^{s} = t_{ready}^{\phi_s^{|U_s|},s} + t_{exe}^{\phi_s^{|U_s|},s},\ \forall s \in \mathbf{S}. \tag{8}$$

Our optimization goal is to minimize the maximum server latency, so we formulate the optimization problem as

$$P_1 : \min_{\mathbf{A},\mathbf{P}} \max_{s\in\mathbf{S}} t_{comp}^{s}(\mathbf{A}, \mathbf{P}), \tag{9}$$

$$s.t.\quad a_{us} = \{0, 1\},\ \forall u \in \mathbf{U},\ s \in \mathbf{S}, \tag{10}$$

$$\sum_{s=1}^{S}\sum_{n=1}^{N} a_{us}^{n} = 1,\ \forall u \in \mathbf{U}, \tag{11}$$

$$\sum_{u=1}^{U} a_{us}^{n} \le 1,\ \forall s \in \mathbf{S},\ n \in \mathbf{N}, \tag{12}$$

$$\sum_{u=1}^{U} a_{us} \le N,\ \forall s \in \mathbf{S}, \tag{13}$$

$$0 < p_u \le p^{max},\ \forall u \in \mathbf{U}. \tag{14}$$

The constraints (10) and (11) make sure a task can only be offloaded to one MEC server using one of its subchannels. We can also learn from constraint (12) that each BS can at most serve one user per subchannel. The constraint (13) tells that a MEC server cannot accept tasks exceeding the subchannel number N.

The constraint (14) specifies the maximum of transmission power for each mobile device which is represented as p^{max}.

This problem is an mixed-integer nonlinear programming (MINLP) problem because it has integer variables in **A** and continuous variables in **P**. Logically speaking, the best solution of P_1 can be get by exhaustive search. If the MEC system contains 20 mobile devices and 8 MEC servers each with 4 subchannels, the exhaustive search time of offloading decision can be as high as $\frac{32!}{(32-20)!} \approx 5.49 \times 10^{26}$ and for each scheduling scheme the optimal transmit power needs to be settled. As we can see, finding the optimal result can be costly so we propose a low complexity sub-optimal method and compare the method with other approaches in the following sections.

Here we summarize the difficult points of the considered problem:

- The task offloading problem and the transmit power allocation problem are related to each other. It is necessary to know the transmit power of each mobile device in order to calculate the transmission time of each task, thereby making a task offloading scheme. On the other hand, we must know the task offloading decisions so that we can adjust the devices' transmit power.
- The accumulated inter-channel interference in the denominator in (4) makes the calculation of data rate extremely complicated. Once a user changes its offloading decision or transmit power, some of the other users' data rate will be affected and so as the object of the problem formulated in P_1.

In the next section, we will analyze the difficulties above and then propose our solutions.

4. Sub-optimal algorithm for joint task offloading and transmit power allocation

The proposed problem is NP-hard and it is difficult to solve the objective function directly. There is a binary vector **A** and a continuous vector **P** which make the problem complicated, so we want to solve the binary part and the continuous part separately. We observe that the constraints (10)–(13) in P_1 are used to constrain **A**, and the constraint (14) is to constrain **P** and they are decoupled from each other. Thus, we decompose the JTOTPA problem into two subproblems: one for the task offloading (TO) decision and the other for transmit power allocation (TPA). The two subproblems are going to be optimized alternately. Here we form the task offloading subproblem as

$$P_2 : \min_{\mathbf{A}} \max_{s \in \mathbf{S}} t^s_{comp}(\mathbf{A}), \quad (15)$$
$$s.t.\ (10),\ (11),\ (12),\ (13),$$

and the transmit power allocation subproblem as

$$P_3 : \min_{\mathbf{P}} \max_{s \in \mathbf{S}} t^s_{comp}(\mathbf{P}), \quad (16)$$
$$s.t.\ (14).$$

We will perform the two subproblems in sequence and then iterate the process to get the sub-optimal result for P_1.

4.1. Task offloading

A user's task offloading decision is made of two parts: server selection and subchannel selection. In a MEC system with 5 users and 2 servers (Fig. 2(a)), users' choosing results of servers are shown in Fig. 2(b) and their choosing decisions of subchannels are depicted in Fig. 2(c). Afterwards, the execution order and ready time of the tasks are displayed in Fig. 2(d). As we can see, the maximum server delay equals to the completion time of task 3 on server 1. Obviously, the delay will change if these users have different server choices. And from expressions (4) and (5) we know that different subchannel choices make the arrival time of tasks different, therefore make the tasks' execution order and the gaps between tasks changed. So, making a proper task offloading scheme is critical to minimize the system delay.

In this part, we further divide the TO problem into two subproblems: (1) server allocation problem that decide to which server each user offloads its task, and (2) subchannel allocation problem that decide through which subchannel each user transmits its task input data.

The two allocation problems both can be posed as user–resource matching problems [27]. The purpose of a matching problem is to get a mutually beneficial combination for two-sided agents according to their preferences. Servers and subchannels can be unified as resources. As each user/resource can only match with one resource/user, our problem boils down to a *one-to-one matching* problem.

One-to-one matching: Each agent can at most match to one agent in the other set.

We use *deferred acceptance (DA) algorithm* to find a *stable matching* for each problem. Here we first introduce the concept of stable matching, then we give an overview of *DA* algorithm in the next section.

Stable matching: A matching has no blocking pair (*BP*).

In a user–resource matching problem, if a user and a resource are not matched with each other but prefer each other than their current partner, they make up a *BP*.

4.1.1. Deferred acceptance algorithm

The *DA* algorithm is an iterative procedure. We summarize it into the following four stages.

Initialize: For each user, initialize its selection list of resources. At start, each user can select any resources, so every selection list contains all the resources.

Propose: Each user (who is not kept by any resources) proposes to its favorite resource in its selection list and delete it from the list.

Reject/keep: Each resource rejects all but its favorite proposer. Here the resource does not accept this favorite user but keeps it as a candidate in case a better choice may come along later.

Terminate: If there are no users rejected by any resources, the iteration ends. If not, these rejected users go to propose stage.

The matching obtained by *DA* algorithm is proved to be stable [28] and the procedure takes polynomial time in one-to-one matching problem.

4.1.2. Server allocation

In the user–server matching part, the servers are the resources in *DA* algorithm. For subchannel n, the users in $\mathbf{U}^n$ and the MEC servers match one to one. The user–server matching game is formally defined as follow.

Definition 4.1. Given two disjoint nonempty sets of players, $\mathbf{U}^n$ and $\mathbf{S}$, a one-to-one matching function $\Phi\colon \mathbf{U}^n \to \mathbf{S}$ is defined such that for all $u \in \mathbf{U}^n$ and $s \in \mathbf{S}$

(1) $\Phi(u) \subseteq \mathbf{S}$ and $|\Phi(u)| = 1$;
(2) $\Phi(s) \subseteq \mathbf{U}^n$ and $|\Phi(s)| \in \{0, 1\}$;
(3) $u = \Phi(s) \leftrightarrow s = \Phi(u)$.

The first two conditions imply that each user can only choose one MEC server to offload its task and each MEC server can accept at most one task on the subchannel. The third condition means if user u chooses MEC server s, MEC server s chooses user u too. That means a two-way match between user u and MEC server s.

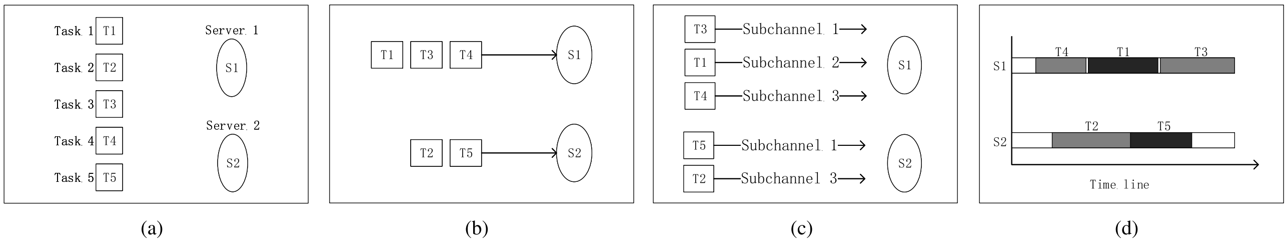


Fig. 2. The tasks' offloading scheduling and execution. 2(a) shows a system with 5 users and 2 MEC servers. 2(b) represents the server selection part of TO scheduling. 2(c) shows the subchannel selection part of TO scheduling. 2(d) reveals the execution order and time consuming of users on MEC servers.

After defining the matching game for server allocation, we then define $\alpha_u(s)$ representing the preference of user u to MEC server s. If $\alpha_u(s_1)$ is greater than $\alpha_u(s_2)$, we consider that user u prefers server s_1 to server s_2. In the same way, we define $\alpha_s(u)$ to measure the preference of MEC server s to user u. When $\alpha_s(u_1)$ is less than $\alpha_s(u_2)$, we think that MEC server s prefers user u_1 to user u_2.

We define the preference value of user u to server s as

$$\alpha_u(s) = R_{us}^n - \omega_1 \sum_{i \in \mathbf{S}, i \neq s} p_u h_{ui}^n, \tag{17}$$

where ω_1 is an equilibrium parameter. The first term is the data rate of user u transmitting its task to server s on subchannel n. It is reasonable because the higher the user's data rate, the shorter the transfer time. The second term is the sum of the interference of user u to the other users on the same subchannel n. The smaller the value of the second term is, the less user u will interfere with the other users. Thus, the transmission rate of the other co-subchannel users will increase.

The preference of MEC server s to user u is

$$\alpha_s(u) = \sum_{k \in \mathbf{S}, k \neq s} \sum_{j \in U_k} a_{jk}^n p_j h_{js}^n + \omega_2 \left| \frac{d_u c_u}{\theta_0} / \frac{f_s}{f_0} - 1 \right|, \tag{18}$$

where $\theta_0 = \frac{\sum_{u \in \mathbf{U}^n} d_u c_u}{|\mathbf{U}^n|}$ is the average amount of CPU circles needed for every task, $f_0 = \frac{\sum_{s=0}^{S} f_s}{S}$ is the average work frequency of all the MEC servers, ω_2 is also an equilibrium parameter. $\frac{d_u c_u}{\theta_0}$ reflects the level of CPU circles required to process task u among all the tasks on subchannel n. If $\frac{d_u c_u}{\theta_0} > 1$, it means finishing task u needs more CPU circles than the average level, we can think of task u as computationally intensive. And if $\frac{d_u c_u}{\theta_0} < 1$, it means task u is computationally sparse. The denominator $\frac{f_s}{f_0}$ reflects the level of work frequency of server s among all the servers. If $\frac{f_s}{f_0} > 1$, it represents the work frequency of MEC server s is greater than the average level. Thus, we believe that MEC server s has a strong computing capacity. If $\frac{f_s}{f_0} < 1$, it signifies that MEC server s is relatively weak in terms of computing capacity. We can see from the formula, the closer $\frac{d_u c_u}{\theta_0} / \frac{f_s}{f_0}$ is to 1, the greater the sth MEC server's preference to user u. That is because we want to match the tasks with the same level servers. The MEC server with more powerful computing capacity prefers the user whose task is computationally intensive, and the MEC server with weak computing power prefers computationally sparse task. This avoids the situation that the server with weak computing power processes computationally intensive tasks while other servers with strong computing power stay idle.

We design Algorithm 1 to perform the matching of users and MEC servers. Before matching, we initialize the to-be-matched server list of each user $\mathbf{S}_u$ ($\forall u \in \mathbf{U}^n$) to $\mathbf{S}$, the unmatched set of users $\mathbf{U}_{unmatched}$ to $\mathbf{U}^n$. For every MEC server, there is a requesting list of users applying for matching which is given as U_s^{req} ($\forall s \in \mathbf{S}$). These requesting lists are set to be empty in initialization.

Algorithm 1 User-server matching algorithm

```
Require:
    The set of to-be-matched servers of each user $u$, $\mathbf{S}_u = \mathbf{S}, u \in \mathbf{U}^n$;
    The set of unmatched users on subchannel $n$, $\mathbf{U}_{unmatched} = \mathbf{U}^n$;
    The set of users requesting MEC server $s$, $U_s^{req} = \emptyset, s \in \mathbf{S}$.
Ensure:
    Find a one-to-one matching $\Phi^*$ for users and servers;
 1: while $\mathbf{U}_{unmatched} \neq \emptyset$ do
 2:     for all $u \in \mathbf{U}_{unmatched}$ do
 3:         Construct the preference of user $u$ by (17);
 4:         Find $s \leftarrow \arg\max_{s \in \mathbf{S}_u} \alpha_u(s)$;
 5:         Remove $s$ from $\mathbf{S}_u$;
 6:         Put $u$ into set $U_s^{req}$;
 7:     end for
 8:     $\mathbf{U}_{unmatched} \leftarrow \emptyset$
 9:     for all $s \in \mathbf{S}$ do
10:         Construct the preference of MEC server $s$ by (18);
11:         Find $u \leftarrow \arg\min_{u \in U_s^{req}} \alpha_s(u)$;
12:         Put the rejected users into unmatched user set
            $\mathbf{U}_{unmatched} \leftarrow \mathbf{U}_{unmatched} \cup U_s^{req} \setminus u$;
13:         MEC server $s$ keeps user $u$ by setting $U_s^{req} \leftarrow \{u\}$;
14:     end for
15: end while
16: Each MEC server matches to the user in its request list.
17: return $\Phi^*$.
```

After that, we proceed the matching part. For u in $\mathbf{U}_{unmatched}$, we calculate its preference to MEC servers in $\mathbf{S}_u$ according to (17), select the MEC server with the largest preference value and delete it from $\mathbf{S}_u$, and then we add user u to the requesting set of the chosen server. For each MEC server we construct its preference to users in its requesting list via (18), keep its favorite user and reject the rest in its request set. Then all the rejected users continue to propose to their next favorite MEC server. The matching steps are repeated until $\mathbf{U}_{unmatched}$ is empty.

4.1.3. Subchannel allocation

For a user who offloads its task to MEC server s, we say that the user is associated with MEC server s. We need to assign every user associated with MEC server s a subchannel to transmit its task input data. Here, we will match each user with a subchannel in the subchannel allocation part.

Definition 4.2. Given two disjoint nonempty sets of players, $\mathbf{U}_s$ and $\mathbf{N}$, a one-to-one matching function Ψ: $\mathbf{U}_s \to \mathbf{N}$ is defined such that for all $u \in \mathbf{U}_s$ and $n \in \mathbf{N}$

(1) $\Psi(u) \subseteq \mathbf{N}$ and $|\Psi(u)| = 1$;
(2) $\Psi(n) \subseteq \mathbf{U}_s$ and $|\Psi(n)| \in \{0, 1\}$;
(3) $u = \Psi(n) \leftrightarrow n = \Psi(u)$.

Algorithm 2 Transmit power allocation algorithm

```
Require:
    The transmission time of user $u$, $t^u_{trans}(p_u)$
    The ready time of user $u$, $t^u_{ready}$
    The subchannel on which user $u$ transmits data, $n$
    The profile of candidate powers, $\mathcal{P}$
    The set of users transmitting data on subchannel $n$, $U^n$.
Ensure:
    Find a transmission power $p^*_u$ for user $u$.
1: if $t^u_{trans}(p_u) < t^u_{ready}$ then
2:     Find $p^l$ satisfies $t^u_{trans}(p^l) \le t^u_{ready}$
3:             and $t^u_{trans}(p^{l+1}) > t^u_{ready}$;
4:     Update $p_u \leftarrow p^l$;
5: end if
6: return $p^*_u$.
```

The first two conditions tell that a user can only select one subchannel to complete data transmission, and a subchannel can only serve one user at most for each server. The third condition means if user u is matched with subchannel n, then subchannel n is matched with user n. In other words, the users and subchannels are in a one-to-one relationship.

Likewise, we define the preference of user u to subchannel n as $\beta_u(n)$. If $\beta_u(n_1)$ is greater than $\beta_u(n_2)$, we say that user u prefers subchannel n_1 to subchannel n_2. And $\beta_n(u)$ represents the preference of subchannel n to user u. We think of subchannel n prefers user u_1 to user u_2 if $\beta_n(u_1)$ is greater than $\beta_n(u_2)$. The preference of user u to subchannel n is defined as

$$\beta_u(n) = W\log_2(1+\gamma^n_{us}) - \omega_3 \sum_{i\in\mathbf{S}, i\neq s} p_u h^n_{ui}, \tag{19}$$

where ω_3 is a weighted parameter, and $\gamma^n_{us} = \frac{p_u h^n_{us}}{\sigma^2+\sum_{k\neq s} max\{p_j h^n_{js} | j\in \mathbf{U}_k\}}$. The preference in (19) implies that users tend to choose subchannels offering higher transmit rate and it also controls users' interference to other users on the same subchannel. To sum up, users want their own transmission rate to be high, and they want to bring as little interference as possible to other users.

Subchannel selections mainly influence the transmission rate and the interference within the same subchannel. So we set the preference of subchannel n to user u the same as (19). That is

$$\beta_n(u) = W\log_2(1+\gamma^n_{us}) - \omega_3 \sum_{i\in\mathbf{S}, i\neq s} p_u h^n_{ui}. \tag{20}$$

And the procedure of user–subchannel matching is the same as user–server matching except that the resource is changed from servers to subchannels and the preference formulas are replaced from (17) and (18) to (19) and (20). The description of user–subchannel matching algorithm is omitted here.

4.2. Transmit power allocation

We notice that the change of a user's transmit power will not only change its own data rate, but also affect the data rate of other users on the same subchannel. Setting the transmit power to maximum for all the users is not a good choice. If the transmit power p_u of user u is increased, the task will arrive at the target MEC server faster. However, at the same time, the interference of user u on other users over the same subchannel will increase, thus will reduce the data transmit speed of these users. In this section, we propose a discretization method to allocate the users' transmit power.

Algorithm 3 JTOTPA algorithm

```
Require:
    U, S, N, $\mathcal{P}$, $maxIter$.
Ensure:
    $A^*$ and $P^*$.
1: Initialization and preprocessing before computation
2: Set $loop \leftarrow 0$;
3: Set the transmit power of all users in U to $p^{max}$;
4: Randomly select a MEC server for each user and make sure
   (13) is satisfied;
5: Task offloading scheduling and transmit power allocation
6: repeat
7:     $loop \leftarrow loop + 1$;
8:     $Obj_{old} \leftarrow Obj_{new}$;
9:     Users–subchannels matching in a single MEC server
10:    for $s = 1$ to $S$ do
11:        Obtain the optimal matching $\Phi^*$ via Algorithm 1;
12:    end for
13:    Users–servers matching over a single subchannel
14:    for $n = 1$ to $N$ do
15:        Obtain the optimal matching $\Psi^*$ via Algorithm 1;
16:    end for
17:    Transmit power allocation for each user
18:    for $u = 1$ to $U$ do
19:        Obtain the optimal transmit power $p^*_u$ via Algorithm 2;
20:    end for
21:    Evaluate the target function in (9);
22:    $Obj_{new} \leftarrow \max_{s\in\mathbf{S}} t^s_{comp}(\mathbf{A}^*, \mathbf{P}^*)$;
23: until $Obj_{old} - Obj_{new} > \epsilon$ and $loop \le maxIter$.
```

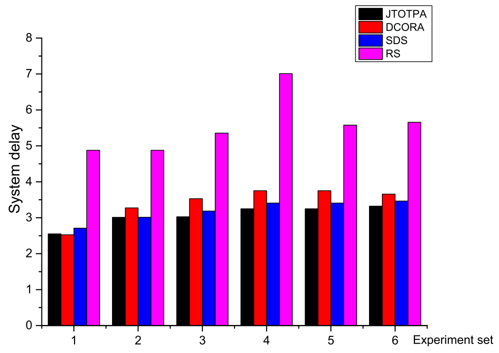
4.2.1. Method

By observing, we find out that not all tasks are immediately executed when they arrive at the MEC server. Some tasks, especially those that arrive relatively late at the MEC server, need to wait for the early arrival tasks to complete before they can be computed. For this part of users, we can reduce their transmit power appropriately, as long as they can arrive at the MEC server before the completion of the previous tasks. In this way, other users in the same subchannel will have a larger transmission rate due to the reduction of inter-channel interference. We set a series of power values $\mathcal{P} \triangleq [p^1, p^2, \dots, p^L]$ as the transmit power choices. L stands for the number of the options and l is an integer which satisfies $1 \le l \le L$. The relation of these values is constrained by $p^{max} = p^1 > p^2 > \cdots > p^L > 0$. If the power p^l meets conditions $t^u_{trans}(p^l) \le t^u_{ready}(p_u)$ and $t^u_{trans}(p^{l+1}) > t^u_{ready}(p_u)$, it is selected as the transmit power of user u.

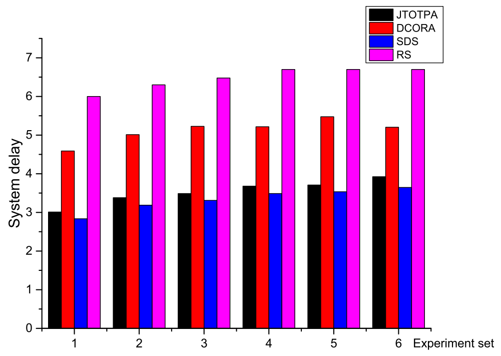
In principle, user u can adjust the transmission power to p^{eq} which satisfies $t^u_{trans}(p^{eq}) = t^u_{ready}(p_u)$. Intuitively, this allows task u to be executed as soon as it arrives at the MEC server. But the value is so tight. Since we adjust the transmit power of users one by one, after the power adjustment of user u, the transmit time of user u may still be affected when other users on the same server make the adjustment. Our method leaves some room for such variations.

4.2.2. Algorithm

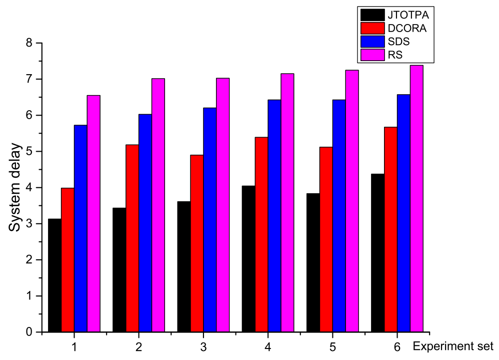
In this part we propose Algorithm 2 for transmit power allocation. For user u, we keep the original transmit power if $t^u_{trans} = t^u_{ready}$ is satisfied. As for $t^u_{trans} < t^u_{ready}$, we find a transmit power $p^l (1 \le l \le L)$ in $\mathcal{P}$ as the new transmit power for user u if $t^u_{trans}(p^l) \le t^u_{ready} < t^u_{trans}(p^{l+1})$. Since the adjustment of user u affects the data rate of all the inter-subchannel users, we need to update the correspondingly changed transmission time of these users.



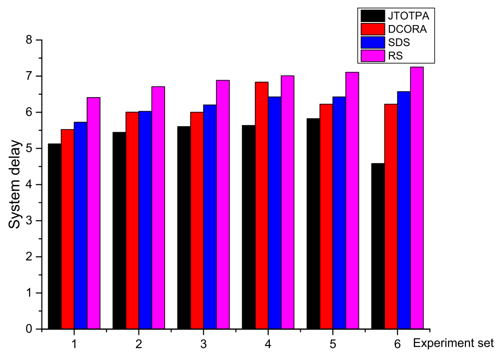
(a) 10 users



(b) 20 users



(c) 30 users



(d) 40 users

Fig. 3. The effect of task input data size on the system delay.

4.3. *The alternating minimization algorithm*

The task offloading decision and the transmit power allocation will be performed in an alternating way and the key steps are summarized in Algorithm 3. The proposed algorithm consists of two parts. Since the matching game between users and subchannels is based on known transmit power of users and a given user–server scheduling, phase one performs the initialization and preprocessing part. The second phase performs task offloading and transmit power allocation iteratively. The algorithm terminates until the object error between two consecutive iterations is less than ϵ or until the number of iterations reaches the maximum *maxIter*.

5. Experimental evaluation

In this section we first investigate JTOTPA's performance in terms of system delay, and then assess its performance in terms of energy expenditure. In our experiments, the area of the MEC system is zoned within a 200×200 m^2 square. The mobile devices and servers are randomly distributed in the region, subject to uniform distribution. The path loss is $-140.7 - 36.7lg(dis)$, in which dis means the distance between the user and the server. All else being equal, the further away a mobile device is from the server, the lower its data rate. Each mobile device has an image processing task. The image number of each task is randomly selected from $\{1, 2, 3, 4, 5, 6, 7\}$. The size of each image is 420 KB. Therefore the task input data size d_u is randomly chosen from $\{420, 840, 1260, 1680, 2100, 2520, 2940\}$ KB. The task workload c_u is chosen from the closed interval $[1, 1595]$ by random. The maximum transmit power of all the mobile devices is 20 dBm. The work frequency of each server is a random value in $\{1, 2, 3, 4, 5\}$ GHz. We use the following methods as comparative experiments:

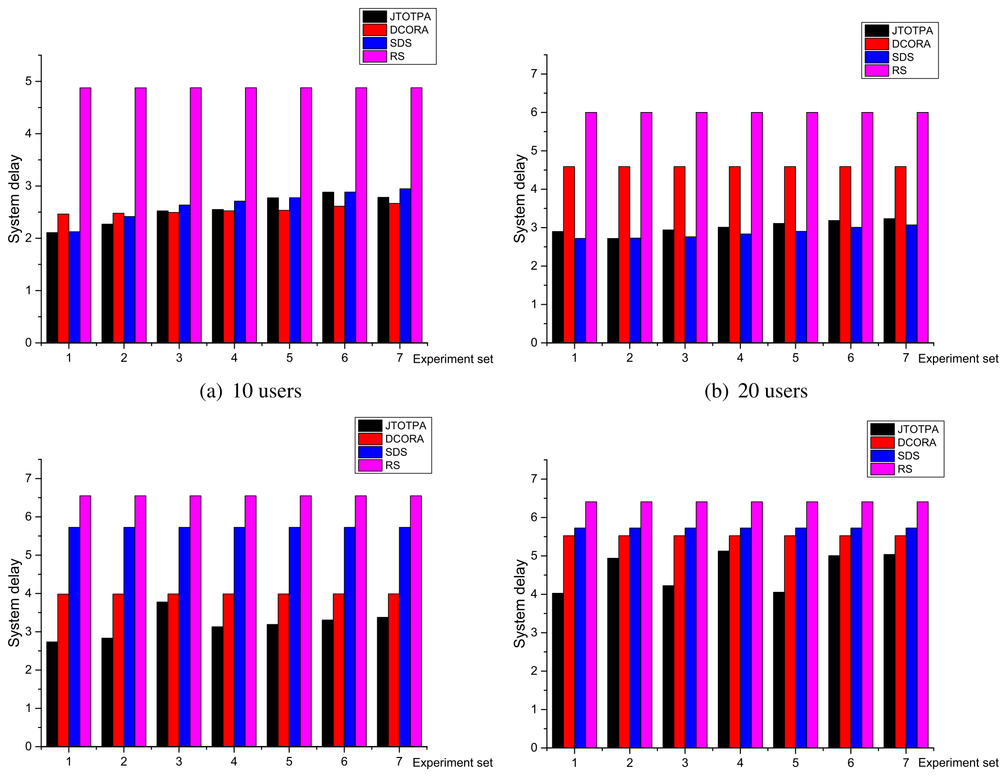
- The decomposed computation offloading and resource allocation approach (DCORA) [29]: the task offloading scheme is confirmed by a many-to-one matching and a one-to-one matching while the transmit power allocation is founded by a bisection method.
- The shortest distance based scheduling approach (SDS): all the tasks are offloaded to the nearest MEC server for computation. All the mobile devices adopt the max transmit power.
- The random scheduling approach (RS): the task offloading decision of each user is chosen randomly. All the mobile devices adopt the max transmit power.

The time complexity of RS is the least, which is $O(U)$, followed by SDS, which is $O(SU)$. For JTOTPA and DCORA, the time complexity of user–subchannel matching process is $O(SN^3)$, and that of transmit power allocation part is $O(U)$. The difference lies in the user–server matching process. JTOTPA's time complexity is $O(S^3N)$ and DCORA's time complexity is $O(SU^2)$. If the number of users in the MEC system reaches the state of full saturation, that is, when $U = S * N$, the time complexity of DCORA is $O(S^3N^2)$. At this time, JTOTPA has more advantages in time expenditure.

We have summarized the parameters in Table 2.

5.1. *Evaluation on system delay*

In this section we explore the relationship between the system delay and the features of task input data. Task input data size has an impact on the transmit time and the execution time of the task. Task workload influences the execution time of the task too. So here we take these two factors into consideration. In all of the experiments, the value of S is 10, the value of N is 4 and the geographical position of MEC servers and mobile devices stays unchanged.



(a) 10 users (b) 20 users

(c) 30 users (d) 40 users

Fig. 4. The effect of task workload on the system delay.

Table 2
The summary of parameter settings.

Expression	Value
S	10
N	4
U	{10, 20, 30, 40}
d_u	{420, 840, 1260, 1680, 2100, 2520, 2940} KB
c_u	[1, 1595] cir/bit
f_s	{1, 2, 3, 4, 5} GHz
W	5 MHz
p^{max}	20 dBm

5.1.1. *Task input data size*

In this part, the effect of task input data size on the system delay is discussed. The task workload of each task is fixed to 800 r/bit. We perform the experiment under different user saturations (*user number/user capacity*), which are 25% (10 users), 50% (20 users), 75% (30 users) and 100% (40 users). For each experiment, the total task input data size in the system gradually increased, each experiment was divided into 6 sets: for $\forall u \in \mathbf{U}$, (1) $d_u = 420$ KB, (2) $d_u \in \{420, 840, 1260\}$ KB, (3) $d_u = 1260$ KB, (4) $d_u \in \{1260, 1680, 2100\}$ KB, (5) $d_u = 2100$ KB, (6) $d_u \in \{2100, 2520, 2940\}$ KB.

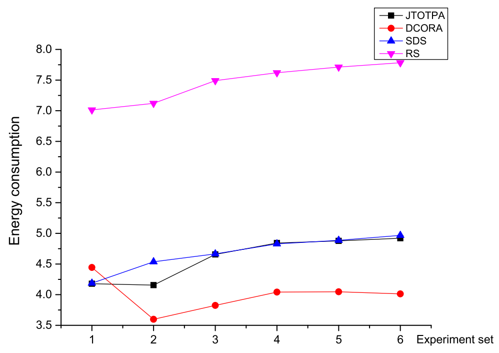
From the experimental results shown in Fig. 3, we can see that, the system delay obtained by JTOTPA algorithm is much lower than the other methods in most cases. And SDS algorithm has good performance at low user saturation (25%, 50%), while has relatively bad performance at high user saturation (75%, 100%). That is because as the user saturation increases the influence among users grows enormous and complicated, choosing the nearest MEC server for users can make the inter-channel interference extremely large. In this case, DCORA has considered the interference so it has better performance than SDS in high user saturation groups.

For JTOTPA algorithm, the system delay only increases with the increase of task input data size in the cases of 25% and 50% user saturation, while there is no unified upward trend in the cases of 75% and 100% user saturation. As can be seen from Figs. 3(c) and 3(d), sets 1, 3 and 5 show an upward trend, while sets 2, 4 and 6 do not follow this trend. This is because the input data size of every task in 1, 3 and 5 sets is exactly the same, while the input data size of tasks in 2, 4 and 6 sets is random within a certain range. According to our matching game for TO, the scheduling scheme of task offloading in 1, 3, 5 experiments remains unchanged, so the system delay naturally increases with the increase of transmission time and execution time. However, when the task input data size is in a random state, the MEC server will adjust its user selection accordingly, and when the user saturation is high (75%, 100%), this adjustment forces some mobile users to choose another MEC server, which may extend the system delay, may shorten the delay.

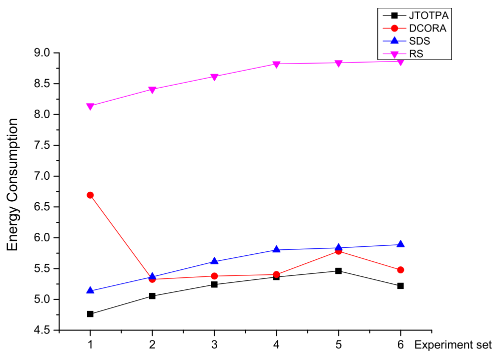
5.1.2. *Task workload*

For different images (even with the same size), due to their different picture quality, the processing consumes different computing resources, so it is necessary to take task workload, which describes the amount of CPU circles needed per unit data, into consideration. We talk about the effect of task workload on the system delay in this part. The input data size of each task is fixed to 420 KB. Same as in the previous part, we set four levels of the user saturation of the system, and conducted four experiments. For each experiment, the total task workload in the system gradually increased, which was graded into 7 sets: (1) $c_u \in [1, 400]$ r/bit, (2) $c_u = 400$ r/bit, (3) $c_u \in [400, 800]$ r/bit, (4) $c_u = 800$ r/bit, (5) $c_u \in [800, 1200]$ r/bit, (6) $c_u = 1200$ r/bit, (7) $c_u \in [1200, 1595]$ r/bit.

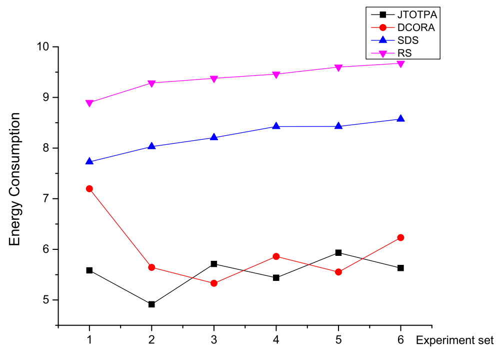
From the four bar charts in Fig. 4, the system delay of the 7 sets of experiments changes slightly for DCORA, SDS and RS



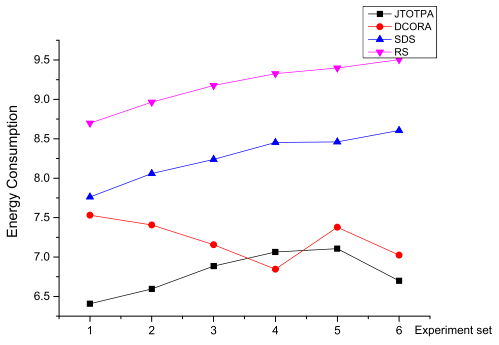
(a) 10 users



(b) 20 users

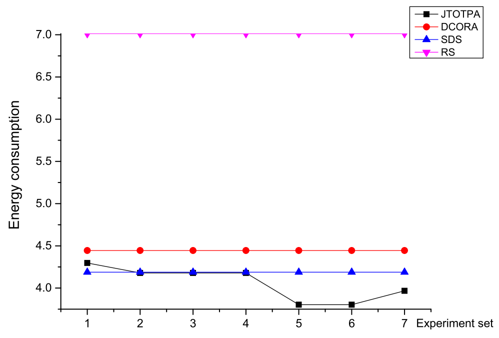


(c) 30 users

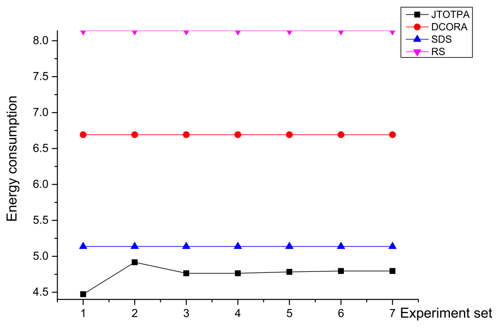


(d) 40 users

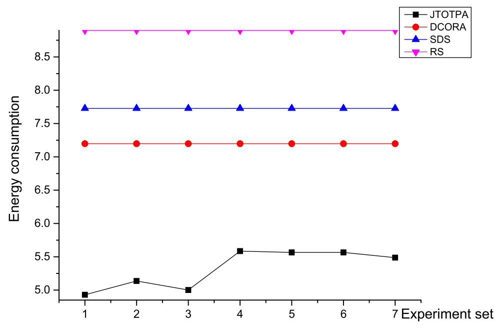
Fig. 5. The effect of task input data size on the energy consumption.



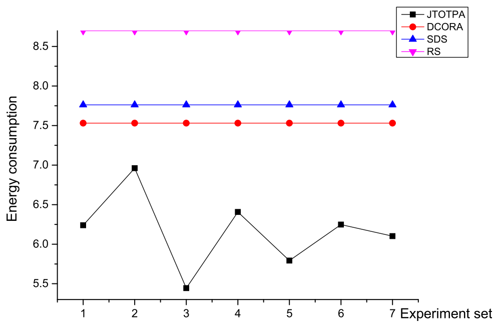
(a) 10 users



(b) 20 users



(c) 30 users



(d) 40 users

Fig. 6. The effect of task workload on the system energy consumption.

algorithms. On the one hand, the task offloading scheme stays unchanged because these three methods have not taken the task workload into account when making offloading decisions. On the other hand, the decision making of the transmit power has not taken into account the task workload factor either. Thus the increase of task workload only affects the users' execution time on the MEC servers which is relatively short as the MEC servers operate at a high frequency.

As for JTOTPA algorithm, we have considered the task workload factor in Eq. (18). The system delay of the 7 sets of experiments in Figs. 4(a) and 4(b) shows an upward trend, while it has some ups and downs in Figs. 4(c) and 4(d). This is caused by the combination of user saturation and tasks' geographical distribution. When the system's user saturation is high (75%, 100%), the randomness of users' task workload distribution has a stronger impact on the system delay.

When it comes to the comparison of these methods, we can see that SDS has good performance in the case of low user saturation. The proposed method JTOTPA obtains the least system delay in high user saturation (75%, 100%). And the system delay of JTOTPA is very close to that of SDS in low user saturation (25%, 50%). The position of method DCORA lies between JTOTPA and SDS in high user saturation (75%, 100%). As for RS, the system delay is the longest in all sets of the four experiments which shows the necessity to develop an offloading strategy rather than using a stochastic strategy.

5.2. Evaluation on energy consumption

In this section, we use energy consumption as the assessment criteria of these four algorithms. Here we only consider the power consumption generated by the users transferring the tasks, not the consumption of the MEC servers processing the data. We define the energy cost of the system as

$$\mathbf{E}(\mathbf{A}, \mathbf{P}) = \sum_{u \in \mathbf{U}} \sum_{s \in \mathbf{S}} a_{us} p_u t^{u,s}_{trans}. \tag{21}$$

Energy consumption is not only related to the time it takes each user to transmit its task, but also related to the transmit power of the mobile devices. In this part, we explore the relationship between energy cost and the features of task data like the previous part.

We mainly examine the advantages and disadvantages of the four methods in terms of the amount of task input data size and task workload. First of all, both of the two factors have an impact on the task offloading scheme and the execution time of the tasks. Second, the task input data size will also affect the task transfer time. At last, the transit power is also influenced by the ready time and the transmit time of users. Consequently, these effects will be reflected in the energy cost. All experimental settings are the same as in Section 5.1.

5.2.1. Task input data size

The experimental results are shown in Fig. 5. Among the four groups of experiments, the performance of the RS algorithm is the worst, because the energy cost generated by RS is much higher than that of the other three methods. In the experiments of the system delay part, it is not difficult to find that the system delay obtained by RS is also the maximum. In addition, this algorithm employs the max transmit power (20 dBm) for users, so it generates greater energy consumption.

In the cases of low user saturation (25%, 50%), SDS algorithm has a small energy consumption, and in the cases of high user saturation (75%, 100%), the energy consumption generated by SDS algorithm increases a lot. This shows that the algorithm is not suitable for high saturation system. When the user saturation is low, the influence between users is small, SDS algorithm can better play its advantages.

With the increase of task data size, the energy consumption obtained by RS and SDS methods gradually increases. Different from RS and SDS, the energy consumption of JTOTPA and DCORA fluctuates irregularly with the increase of task data size. This is because the latter two methods change their task offloading schemes and adjust the transmit power of users according to the amount of data of the tasks. It can be seen from Fig. 5 that these changes and adjustments make the latter two methods better than RS and SDS in terms of energy consumption.

Table 3
The G value of the four methods.

Method	Average value of G
JTOTPA	3.1402
DCORA	3.0161
SDS	2.9691
RS	0.8327

In the first group of experiments, we can see that DCORA produces less energy consumption, while in the second group of experiments, JTOTPA obtains better performance. In the third and fourth group of experiments, the best performance is generated by the two methods alternately. It is hard to say which method is better as the geographical distribution of data size plays an important role. Both approaches have their own focus and would produce different results for different geographical distributions of data size.

5.2.2. Task workload

The effect of task workload on the system energy consumption is shown in Fig. 6. Obviously, of the four algorithms we compare, three of them (DCORA, SDS, RS) are insensitive to the workload of the tasks, so their line graphs show horizontal curves. The worst performer is still the RS algorithm. And the energy consumption of SDS and DCORA algorithms is much lower than that of RS, while SDS is better than DCORA in the cases of low user saturation (25%, 50%), and worse than DCORA in the cases of high user saturation (75%, 100%).

In these four groups of experiments, we can see that JTOTPA performs best in most cases by getting the least energy cost. When the data amount of every task in the system is the same but their workload is different, JTOTPA can effectively adjust the transmit power and the task offloading scheme to keep the delay and the energy consumption of the system at a low level. With the increase of task workload, energy consumption of JTOTPA does not show a consistent rule. This is due to the complex changes in the geographic distribution of task workload, which we will not analyze in detail in this paper.

5.3. Comprehensive evaluation

In the last two parts, we performed experiments about the system delay and the energy consumption of the mobile devices, and compared the advantages and disadvantages of the four methods. In this part, we will further consider the tradeoff between delay and energy [30,31]. We use a proportion G [32] to evaluate the four methods. The proportion G is defined as

$$G = G_1^{\rho} G_2^{1-\rho}, \tag{22}$$

where G_1 refers to the ratio of the execution time (local/remote), G_2 is the ratio of energy consumption (local/remote) and ρ is the weight coefficient ($0 \leq \rho \leq 1$).

Both time and energy saving is considered in G. As a performance indicator, a larger G means a better offloading system [32]. We have calculated the G value of each user when there are 40 users in the system and averaged the value. The results obtained by the four methods are as Table 3.

Considering the experimental performance of both system delay and energy consumption, JTOTPA still has certain advantages compared to the other three methods, while the RS method has the smallest G value and a larger gap with the other three methods, which again illustrates the necessity of developing an offload strategy. DCORA also has a better performance than SDS which indicates that choosing the nearest server for each user is not a good choice either.

6. Conclusion

In this paper, we settled the problem of joint task offloading and transmit power allocation of multiple mobile users in MEC systems with multiple servers. Based on matching theory and a heuristic approach, we proposed two sub-optimal algorithms to alternately minimize the maximum server delay. The proposed method mainly has the following two innovations: (1) considering the balance between the total computation workload of tasks and the computing capacity of the MEC servers; (2) using power discretization for the allocation of users' transmit power. The experimental results show that JTOTPA cannot only obtain less delay, but also generate less energy consumption when the data amount of tasks is the same but the workload is random in the MEC system.

During the experiments, we found that the geographical distribution of task data in the system also had an important impact on the system delay. Therefore, we want to further study and adopt different task offloading and power allocation methods to optimize the system delay under different geographical distribution modes. In addition, we also consider the inclusion of task deadline in the system for research.

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.

Acknowledgments

This work was partially supported by the National Key Research and Development Program of China (2018YFB1003401, 2018YFB0203804), the Program of National Natural Science Foundation of China (Grant No. 61702170), and the Fundamental Research Funds for the Central Universities.

References

[1] S. Tripathi, Group interaction with smart phones at work place, in: International Conference on Human Computer Interaction with Mobile Devices & Services, 2007.

[2] A. Puder, I. Yoon, Smartphone cross-compilation framework for multiplayer online games, in: International Conference on Mobile, 2010.

[3] W. Jiang, J. Wu, F. Li, G. Wang, H. Zheng, Trust evaluation in online social networks using generalized flow, IEEE Trans. Comput. 65 (3) (2016) 952–963.

[4] H. Nejati, V. Pomponiu, T. Do, Y. Zhou, S. Iravani, N. Cheung, Smartphone and mobile image processing for assisted living: Health-monitoring apps powered by advanced mobile imaging algorithms, IEEE Signal Process. Mag. 33 (4) (2016) 30–48, http://dx.doi.org/10.1109/MSP.2016.2549996.

[5] L.W. Chen, Y.H. Peng, Y.C. Tseng, An augmented reality based group communication system for bikers using smart phones, in: 2011 IEEE International Conference on Pervasive Computing and Communications Workshops, PERCOM Workshops, 2011, pp. 325–327, http://dx.doi.org/10.1109/PERCOMW.2011.5766896.

[6] T.L. Chou, L.J. Chanlin, Augmented reality smartphone environment orientation application: A case study of the Fu–Jen university mobile campus touring system, Proc. Soc. Behav. Sci. 46 (2012) 410–416, http://dx.doi.org/10.1016/j.sbspro.2012.05.132, 4th World Conference on Educational Sciences (WCES-2012) 02-05 February 2012 Barcelona, Spain.

[7] W. Jiang, J. Wu, G. Wang, H. Zheng, Forming opinions via trusted friends: Time-evolving rating prediction using fluid dynamics, IEEE Trans. Comput. 65(4) (2016) 1211–1224.

[8] W. Jiang, G. Wang, M.Z.A. Bhuiyan, J. Wu, Understanding graph-based trust evaluation in online social networks: Methodologies and challenges, ACM Comput. Surv. 49 (1) (2016) 10:1–10:35, http://dx.doi.org/10.1145/2906151, URL http://doi.acm.org/10.1145/2906151.

[9] W. Jiang, J. Wu, G. Wang, On selecting recommenders for trust evaluation in online social networks, ACM Trans. Internet Technol. 15 (4) (2015) 14:1–14:21, http://dx.doi.org/10.1145/2807697, URL http://doi.acm.org/10.1145/2807697.

[10] V. Namboodiri, T. Ghose, To cloud or not to cloud: A mobile device perspective on energy consumption of applications, in: 2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM, 2012, pp. 1–9, http://dx.doi.org/10.1109/WoWMoM.2012.6263712.

[11] C. Liu, K. Li, C. Xu, K. Li, Strategy configurations of multiple users competition for cloud service reservation, IEEE Trans. Parallel Distrib. Syst. 27 (2) (2016) 508–520, http://dx.doi.org/10.1109/TPDS.2015.2398435.

[12] K. Li, C. Liu, K. Li, A.Y. Zomaya, A framework of price bidding configurations for resource usage in cloud computing, IEEE Trans. Parallel Distrib. Syst. 27 (8) (2016) 2168–2181, http://dx.doi.org/10.1109/TPDS.2015.2495120.

[13] C. Liu, K. Li, K. Li, Minimal cost server configuration for meeting time-varying resource demands in cloud centers, IEEE Trans. Parallel Distrib. Syst. 29 (11) (2018) 2503–2513, http://dx.doi.org/10.1109/TPDS.2018.2836452.

[14] Y. Mao, C. You, J. Zhang, K. Huang, K.B. Letaief, A survey on mobile edge computing: The communication perspective, IEEE Commun. Surv. Tutor. 19 (4) (2017) 2322–2358, http://dx.doi.org/10.1109/COMST.2017.2745201.

[15] S.S.D. Ali, H. Ping Zhao, H. Kim, Mobile edge computing: A promising paradigm for future communication systems, in: TENCON 2018 - 2018 IEEE Region 10 Conference, 2018, pp. 1183–1187, http://dx.doi.org/10.1109/TENCON.2018.8650169.

[16] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: Vision and challenges, IEEE Internet Things J. 3 (5) (2016) 637–646, http://dx.doi.org/10.1109/JIOT.2016.2579198.

[17] C. You, K. Huang, Multiuser resource allocation for mobile-edge computation offloading, in: 2016 IEEE Global Communications Conference, GLOBECOM, 2016, pp. 1–6, http://dx.doi.org/10.1109/GLOCOM.2016.7842016.

[18] X. Lyu, W. Ni, H. Tian, R.P. Liu, X. Wang, G.B. Giannakis, A. Paulraj, Optimal schedule of mobile edge computing for internet of things using partial information, IEEE J. Sel. Areas Commun. 35 (11) (2017) 2606–2615, http://dx.doi.org/10.1109/JSAC.2017.2760186.

[19] W. Zhang, Y. Wen, K. Guan, K. Dan, D.O. Wu, Energy-optimal mobile cloud computing under stochastic wireless channel, IEEE Trans. Wireless Commun. 12 (9) (2013) 4569–4581.

[20] J. Liu, Y. Mao, J. Zhang, K.B. Letaief, Delay-optimal computation task scheduling for mobile-edge computing systems, in: 2016 IEEE International Symposium on Information Theory, ISIT, 2016, pp. 1451–1455, http://dx.doi.org/10.1109/ISIT.2016.7541539.

[21] T.X. Tran, D. Pompili, Joint task offloading and resource allocation for multi-server mobile-edge computing networks, IEEE Trans. Veh. Technol. 68 (1) (2019) 856–868, http://dx.doi.org/10.1109/TVT.2018.2881191.

[22] X. Chen, Decentralized computation offloading game for mobile cloud computing, IEEE Trans. Parallel Distrib. Syst. 26 (4) (2015) 974–983, http://dx.doi.org/10.1109/TPDS.2014.2316834.

[23] S. Sardellitti, G. Scutari, S. Barbarossa, Joint optimization of radio and computational resources for multicell mobile-edge computing, IEEE Trans. Signal Inf. Process. Netw. 1 (2) (2015) 89–103, http://dx.doi.org/10.1109/TSIPN.2015.2448520.

[24] C. You, Z. Yong, Z. Rui, K. Huang, Asynchronous mobile-edge computation offloading: Energy-efficient resource management, IEEE Trans. Wireless Commun. (2018) 1.

[25] H.Q. Le, H. Al-Shatri, A. Klein, Efficient resource allocation in mobile-edge computation offloading: Completion time minimization, in: 2017 IEEE International Symposium on Information Theory, ISIT, 2017, pp. 2513–2517, http://dx.doi.org/10.1109/ISIT.2017.8006982.

[26] Y. Mao, J. Zhang, K.B. Letaief, Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems, in: 2017 IEEE Wireless Communications and Networking Conference, WCNC, 2017, pp. 1–6, http://dx.doi.org/10.1109/WCNC.2017.7925615.

[27] Y. Gu, W. Saad, M. Bennis, M. Debbah, Z. Han, Matching theory for future wireless networks: fundamentals and applications, IEEE Commun. Mag. 53 (5) (2015) 52–59, http://dx.doi.org/10.1109/MCOM.2015.7105641.

[28] D. Gale, L. S. Shapley, College admissions and stability of marriage, Amer. Math. Monthly 69 (2013) 9–15, http://dx.doi.org/10.4169/amer.math.monthly.120.05.386.

[29] Q. Pham, T. Leanh, N.H. Tran, B.J. Park, C.S. Hong, Decentralized computation offloading and resource allocation for mobile-edge computing: A matching game approach, IEEE Access 6 (2018) 75868–75885, http://dx.doi.org/10.1109/ACCESS.2018.2882800.

[30] Z. Jiang, S. Mao, Energy delay tradeoff in cloud offloading for multi-core mobile devices, IEEE Access 3 (2015) 2306–2316, http://dx.doi.org/10.1109/ACCESS.2015.2499300.

[31] H. Wu, K. Wolter, Stochastic analysis of delayed mobile offloading in heterogeneous networks, IEEE Trans. Mob. Comput. 17 (2) (2018) 461–474, http://dx.doi.org/10.1109/TMC.2017.2711014.

[32] H. Wu, Q. Wang, K. Wolter, Tradeoff between performance improvement and energy saving in mobile cloud offloading systems, in: 2013 IEEE International Conference on Communications Workshops, ICC, 2013, pp. 728–732, http://dx.doi.org/10.1109/ICCW.2013.6649329.

Surong Xiao is currently studying for a master's degree at Hunan University in Changsha, China. Her research interest is in modeling and scheduling of mobile edge computing systems.

Chubo Liu received the BS and Ph.D. degrees in computer science and technology from Hunan University, China, in 2011 and 2016, respectively. He is currently an associate professor of computer science and technology with Hunan University in Changsha. His research interests are mainly in modeling and scheduling of distributed computing systems, approximation and randomized algorithms, game theory, grid and cloud computing. He has published over 10 papers in journals such as the *IEEE Transactions on Parallel and Distributed Systems*, the *IEEE Transactions on Cloud Computing*, the *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, the *Future Generation Computer Systems*, and the *Theoretical Computer Science*.

Kenli Li received the Ph.D. degree in computer science from the Huazhong University of Science and Technology, China, in 2003. He was a visiting scholar with University of Illinois at Urbana-Champaign from 2004 to 2005. He is currently the dean and a full professor of computer science and technology with Hunan University and the director of National Supercomputing Center in Changsha. His major research areas include parallel computing, high-performance computing, grid and cloud computing. He has published more than 160 research papers in international conferences and journals such as the *IEEE Transactions on Computers*, the *IEEE Transactions on Parallel and Distributed Systems*, the *IEEE Transactions on Signal Processing*, the *Journal of Parallel and Distributed Systems*, ICPP, and CCGrid. He serves on the editorial board of the *IEEE Transactions on Computers*. He is an outstanding member of CCF. He is a senior member of the IEEE.

Keqin Li is a SUNY Distinguished Professor of computer science with the State University of New York. He is also a Distinguished Professor at Hunan University, China. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and cyber–physical systems, heterogeneous computing systems, big data computing, high-performance computing, CPU–GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, intelligent and soft computing. He has published over 630 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He currently serves or has served on the editorial boards of the *IEEE Transactions on Parallel and Distributed Systems*, the *IEEE Transactions on Computers*, the *IEEE Transactions on Cloud Computing*, the *IEEE Transactions on Services Computing*, and the *IEEE Transactions on Sustainable Computing*. He is an IEEE fellow.