Journal of Systems Architecture 112 (2021) 101849

Contents lists available at ScienceDirect

Journal of Systems Architecture

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

Task migration optimization for guaranteeing delay deadline with mobility consideration in mobile edge computing

Fan Tang [a], Chubo Liu [a,*], Kenli Li [a,*], Zhuo Tang [a], Keqin Li [a,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

Keywords:
Delay deadline
Mobile edge computing
Mobility
Quality of service
Task migration

ABSTRACT

Mobile edge computing (MEC) is envisioned to integrate cloud-like capabilities into the edge of networks for improving quality of service (QoS). This makes it possible for users with resource-limited devices to execute computation-intensive tasks by offloading them to MEC nodes. Extensive works have been done for MEC. However, few of them involve user mobility. Whether to migrate task dynamically cannot be ignored when taking QoS into account. In this paper, we try to optimize task migration with user mobility consideration, in which deadlines of tasks are also involved. The problem is proved to be NP-hard. To solve it, we analyze three variants of this problem and devise a group migration (GM) algorithm with known trajectories of users. Our goal is to maximize the number of tasks whose deadlines are guaranteed. Extensive experiments are carried out, and the results confirm that GM algorithm can achieve up 35%-75% performance improvement compared three other common heuristics.

1. Introduction

1.1. Motivation

Nowadays, the explosively increasing smart phones, tablets, and other mobile devices have become an indispensable part in people's daily life [1]. With the increasing usage of mobile devices, a wide variety of mobile applications, as exemplified by autonomous driving and face recognition, augmented reality (AR), virtual reality (VR), online gaming, emerge and attract people's attention [2–5]. However, better quality of service (QoS) is always pursued by mobile users, which contradicts with running these computation-intensive and latency-sensitive applications due to mobile device's limited resources (i.e., memory, battery). To mitigate this tension, mobile edge computing (MEC) [2] has emerged as a promising technology. Related edge computing paradigms, such as fog computing, cloudlet, mist computing, are compared in [6].

Deployed at the edge of networks and in close proximity to mobile users, MEC is envisioned to deliver some cloud-like capabilities efficiently [7]. Mobile users with resource-limited devices can offload computation-intensive or latency-sensitive tasks to MEC nodes for execution. Extensive studies have been done for MEC . Nevertheless, most of them only concern about computation offloading schemes in MEC. They usually study the quasi-static scenario and overlook the user mobility that is one of the most important features in MEC.

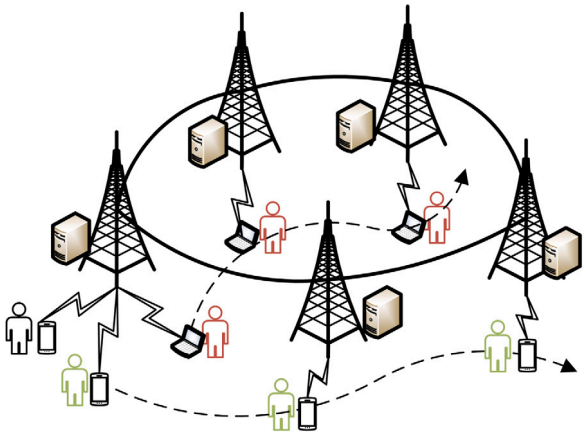

Fig. 1. An example that two mobile users move across multiple MEC nodes after offloading their heavy tasks to the MEC node on the far left.

Let us consider a scenario shown in Fig. 1, there are two mobile users endowed with resource-limited devices moving across multiple

* Corresponding authors.
E-mail addresses: fantang@hnu.edu.cn (F. Tang), liuchubo@hnu.edu.cn (C. Liu), lkl@hnu.edu.cn (K. Li), ztang@hnu.edu.cn (Z. Tang), lik@newpaltz.edu (K. Li).

https://doi.org/10.1016/j.sysarc.2020.101849
Received 12 April 2020; Received in revised form 9 July 2020; Accepted 2 August 2020
Available online 6 August 2020

MEC nodes after offloading their tasks to the MEC node on the far left. However, mobile users may move out of the coverage of the far left MEC node when their tasks are completed, thus we need to consider whether to migrate tasks dynamically with QoS (e.g., task's delay deadline) consideration.

To our knowledge, only a few works consider user mobility in MEC. Some of them assume that user mobility follows a Markovian process, but there are some cases where the Markovian assumption is not valid [24]. Some works exploit mobility information by deep learning and neural networks technology. However, none of them involves deadlines of tasks. The problem that designs an efficient task migration scheme is challenging when delay deadlines of tasks are involved. The reason lies in that completion time of task includes transmission time of input data and output data, execution time on MEC node, and migration cost among MEC nodes. All of them will be greatly influenced by different task migration schemes. Nevertheless, guaranteeing delay deadlines of the above latency-sensitive applications is important in MEC. Hence, the motivation of this paper is trying our best to make some supplementations from our perspective.

1.2. Related work

During the past several years, computation offloading in MEC has been extensively studied by numerous researchers (see [25] for a comprehensive survey). There are some researchers concerned with delay-optimal computation offloading in MEC [8–16]. Some of which assume that computing resources are infinite and overlook the congestion of computing resources among multiple tasks [8,13]. Then the computation offloading problem is simplified to a wireless resource allocation problem. In [10], Liu et al. addressed the problem of power-constrained delay minimization, and demonstrated the optimal task offloading policy can obtain by a one-dimensional search algorithm. In [11], with the assumption that arrival of task obeys the Poisson distribution, Sun et al. tried to minimize the average delay of tasks under multiple users and multiple MEC nodes environment. The authors considered delay as a partial goal in their work [12]. In [15], the authors addressed task offloading problem to maximize the weighted sum computation rate, with system resources management and task computing time allocation. Luo et al. tackled the offloading scheduling problem to minimize the weighted sum of the total latency delay and device energy consumption [16]. The problem will be trickier when delay deadlines of tasks need to be guaranteed, however, only a few works have involved delay deadlines of tasks. In [9], the authors formulated the offloading problem as a cooperative game to maximize the number of tasks whose delay deadlines are satisfied. Jiang et al. addressed the parallel real-time tasks modeled as directed acyclic graphs (DAG) with tight relative deadlines [14]. Nevertheless, all of the aforementioned works ignored user mobility which is one of the most important features in MEC.

Compared with the works which do not involve user mobility, works involving user mobility are far less. As shown in Table 1, according to the way of mobility modeling, we classify existing work into three categories, i.e., online processing [17,22], Markovian hypothesis [19, 23], and with known trajectory [18,20,21]. In the first category, lack of mobility information, the migration decision is triggered after the position of user is changed. which is a kind of delayed processing method and suitable for scenarios that mobility is hard to predict and model. The authors aimed at minimizing the average delay under long-term energy budget constraint, based on Lyapunov optimization due to the absence of mobility information [17,22]. Generally, studies for the second category assume that user mobility follows a Markovian process, and then adopt the method of Markov Decision Process. Taleb et al. proposed a framework named Follow-Me Cloud and considered user experience as a partial goal with the assumption that the user mobility follows a Markovian process [23]. The authors formulated the task migration problem as a sequential decision-making problem, in which the decision was only based on the distance between mobile user and MEC node [19]. However, there are some cases where the Markovian assumption is not valid. For example, the author in [24] showed that mobility is non-Markovian especially when the moving object tends to reach a specific destination.

Research for the last category usually devise task migration schemes with known trajectories of users. With the rapid development of deep learning and neural network technology, a short-term mobility prediction can achieve the accuracy of 95% or higher [26]. This makes it rational to exploit mobility information when designing task migration scheme. Plachy et al. tried to find a suitable communication path to further reduce offloading delay [21]. In [20], Nadembega et al. also considered quality of experience as a partial goal with a mobility-based prediction scheme DAMP [27]. The authors tried to minimize the average time cost with known trajectories of users [18]. Nevertheless, as shown in Table 1, none of the above works involves deadlines of tasks, and the problem becomes even harder when simultaneously involving mobility and deadline guarantees.

Motivated by the above facts, we try to optimize task migration with user mobility consideration under multiple users and multiple MEC nodes environment, in which deadlines of tasks are also involved.

1.3. Contributions

In this paper, we investigate the problem of task migration with user mobility consideration under multiple users and multiple MEC nodes environment, while delay deadlines of tasks are also involved. Specifically, each mobile user is associated with a computation-intensive task to be offloaded to the nearby MEC node. Each task has an expected delay deadline that needs to be guaranteed. Mobile user may move out of the coverage of the nearby MEC node when its task is completed. Thus we need to consider whether to migrate task dynamically with delay deadline of the task consideration. Our goal is to find a task migration scheme to maximize the number of tasks whose delay deadlines are guaranteed. The main contributions of this paper can be summarized as follows.

- By analyzing properties of three variants of this problem, plenty of theoretical analyzes are presented and significant algorithms have been proposed as a guideline to solve the original problem.
- We devise a group migration (GM) algorithm with known trajectories of users to maximize the number of tasks whose deadlines are guaranteed.
- Extensive experiments are carried out to evaluate the performance of GM algorithm for the number of tasks whose deadlines are guaranteed. Compared with three other common heuristics, the results confirm that GM algorithm can achieve up 35%-75% performance improvement.

The rest of this paper is organized as follows. Section 2 presents system model and definitions of notations used in this paper. Maximum optimization problem in which the number of tasks whose deadlines are guaranteed is formulated in Section 3. In Section 4, we analyze three variants of this problem and devise a group migration (GM) algorithm with known trajectories of users. Extensive experiments are carried out to evaluate the performance of GM algorithm in Section 5. Finally, the conclusion is presented in Section 6.

2. System model

2.1. System overview

In this section, we introduce the system model. As shown in Fig. 1, we consider an MEC environment with a set $\mathcal{M} = \{1, 2, \ldots, M\}$ of MEC nodes and a set $\mathcal{N} = \{1, 2, \ldots, N\}$ of mobile users. Each MEC node could be a wireless access point or a base station which is endowed with computing resources and wireless resources. Specifically, main system characteristics of this paper can be summarized as follows.

Table 1
Comparison of related work.

Properties	Without User Mobility [8–16]	With User Mobility [17–23]
Number of mobile users	Single [10,12], Multiple [8,9,11,13–16]	Single [17,19,21], Multiple [18,20,22,23]
Tasks with deadlines	[9,14], No [8,10–13,15,16]	No [17–23]
Number of MEC nodes	Single [8,10,12,15,16], Multiple [9,11,13,14]	Multiple [17–23]
Computing resource of MEC nodes	Infinite [8,13], Limited [9–12,14–16]	Infinite [19], Limited [17,18,20–23]
User mobility	No user mobility [8–16]	Online processing [17,22], Markov process [19,23], Known trajectory [18,20,21]

- Each mobile user with a computation-intensive and latency-sensitive task which needs to be executed on MEC node.
- Each task has a strict deadline that needs to be guaranteed.
- Multiple heterogeneous MEC nodes with limited computing resources.
- Considering user mobility with known trajectory.
- The objective is to maximize the number of tasks whose delay deadlines are guaranteed.

Note that each mobile user with multiple tasks can also be supported by dynamically calling our approach. We use $\mathcal{J} = \{J_1, J_2, \dots, J_N\}$ to denote task set of all mobile users. We consider a discrete time frames with $\mathcal{T} = \{1, 2, \dots, T\}$, in which T denotes the maximum number of time slots, and the length of one time slot is equal to τ. For example in Fig. 1, mobile user $n \in \mathcal{N}$ moves to the right after offloading its task $J_n \in \mathcal{J}$ to the far left MEC node for execution. MEC node executes task J_n at time slot $t-1$ is the source MEC node of task J_n at time slot $t \in \mathcal{T}$. When task J_n is completed at time slot t, MEC node which covers mobile user n at time slot t is destination MEC node of task J_n. We use $s_n^t \in \mathcal{M}$ and $h_n^t \in \mathcal{M}$ to denote source MEC node and destination MEC node of task J_n. Note that source MEC node and destination MEC node of a task may be different at different time slots. For each time slot t, controller of MEC nodes makes a task migration scheme to maximize the number of tasks whose delay deadlines are guaranteed. Migration decisions of all tasks in time slot t can be expressed as $\mathcal{A}^t = \{a_1^t, a_2^t, \dots, a_n^t\}$, where $a_n^t \in \{0, s_n^t, h_n^t\}$ denotes the MEC node which executes task J_n at time slot t. Each task can only be executed on one MEC node at each time slot. textcolorblueUncompleted task of a mobile user needs to be executed at each time slot before its completion. Specially, we have $a_n^t = 0$ if task J_n is completed before time slot t. All notations we used in this paper are summarized in Table 2.

2.2. Communication model

Without loss of generality, we adopt a widely used parameter model to describe task $J_n \triangleq (\lambda_n, c_n, d_n, O_n)$. Specifically, λ_n (in bits) denotes input data size of task J_n, c_n (in CPU cycles/bit) is workload requirement of task J_n, d_n and O_n denote delay deadline and output data size of task J_n, respectively. Given transmission power p_n of mobile user n, with reference to [1], uplink data transmission rate between mobile user n and source MEC node $m \in \mathcal{M}$ can be expressed as

$$r_{up}(m,n) = W_m \log_2\left(1 + \frac{p_n H_{m,n}}{\theta^m}\right), \tag{1}$$

where W_m is wireless channel bandwidth that MEC node m allocates for mobile user n, $H_{m,n}$ is channel gain between user n and MEC node m, θ^m is white noise power. Then we can compute transmission time of input data between mobile user n and MEC node m as

$$T_u(m,n) = \frac{\lambda_n}{r_{up}(m,n)}. \tag{2}$$

Note that transmission time of output data is not ignored in our work, the same as [1,10,11]. We assume that downlink data transmission rate $r_{down}(m,n)$ is equal to uplink data transmission rate $r_{up}(m,n)$. Thus downlink data transmission rate $r_{down}(m,n)$ can be expressed as

$$r_{down}(m,n) = W_m \log_2\left(1 + \frac{p_n H_{m,n}}{\theta^m}\right). \tag{3}$$

Table 2
Notations.

Notation	Definition
M, N	number of MEC node and user
J_n	task of mobile user n
a_n^t	MEC node that executes task J_n in time slot t
λ_n, o_n	input and output data size of task J_n
c_n	workload requirement of task J_n
d_n	delay deadline of task J_n
p_n	transmission power of mobile user n
W_m	wireless bandwidth of MEC node m
θ^m	white noise power
$H_{m,n}$	channel gain between MEC node m and user n
$r_{up}(m,n), r_{down}(m,n)$	uplink and downlink data transmission rate between MEC node m and mobile user n
$T_u(m,n)$	transmission time of input data between MEC node m and mobile user n
$T_d(h_n^t, n)$	transmission time of output data between MEC node h_n^t and mobile user n
B	wire bandwidth among MEC nodes
γ_n^t	remaining CPU cycles requirement in time slot t
f_m	maximal CPU frequency of MEC node m
$T_u(n,t)$	remaining execution time of task J_n in time slot t
$T(n,\lambda), T(n,o)$	migration time of input and output data of task J_n
τ	length of time slot
$D(n)$	completion time of task J_n

When task J_n is completed, output data of the task will be transmitted from destination MEC node h_n^t to mobile user n. We can calculate transmission time of output data between destination MEC node h_n^t and mobile user n as

$$T_d(h_n^t, n) = \frac{O_n}{r_{down}(h_n^t, n)}. \tag{4}$$

2.3. Computation model

We use f_m to describe the maximum computation capability (i.e., CPU cycles/s) that MEC node m can provide to execute task. Here we consider a heterogeneous edge computing environment that different MEC nodes may have different computation capabilities. MEC node can execute multiple tasks simultaneously by using processor sharing, to model the interference among tasks accurately is hard, which is influenced by many factors. Furthermore, we focus on optimizing task migration with user mobility consideration in this paper. Therefore, we consider a simple way of CPU cycles sharing. For each time slot t, several tasks may be executed on the same MEC node (i.e., $a_n^t = a_{n-1}^t$), thus they have to share computing resources of the MEC node. Besides, we define γ_n^t to denote how many remaining CPU cycles are required for task J_n at the beginning of time slot t, where $\gamma_n^1 = \lambda_n \cdot c_n$. Thus completion time of task J_n executed on MEC node m can be estimated as

$$T_e(n) = \frac{\gamma_n^t}{\frac{f_m}{\sum_{i \in \mathcal{N}} I\{a_i^t = a_n^t\}}}, \tag{5}$$

$$a_n^t = m, \quad \forall n, t, m. \tag{6}$$

Here $I\{x\}$ is an indicator function with $I\{x\} = 1$ if the event x is true and $I\{x\} = 0$ otherwise.

2.4. Migration model

For the presence of user mobility, controller of MEC nodes is supposed to make a task migration scheme to satisfy requirements of users (i.e., delay deadlines of tasks). Specifically, if controller decides to migrate task J_n from source MEC node to destination MEC node for execution ($a_n^t \neq a_n^{t-1}$), then extra migration cost of transmitting input data for task J_n cannot be ignored. Migration time of input data for task J_n between source MEC node and destination MEC node can be expressed as

$$T(n,\lambda) = \frac{\lambda_n}{B}, \tag{7}$$

where B is wire network bandwidth among MEC nodes. For simplicity, we assume that wire network bandwidth is same between any two MEC nodes. Overall migration is considered in our migration model at present, and partial migration of task will be considered in our future work. On the other hand, if controller decides to not migrate task J_n but mobile user n is beyond the coverage of source MEC node when task J_n is completed, then output data of task J_n needs to be migrated from source MEC node to destination MEC node. Similarly, we can calculate migration time of output data for task J_n between source MEC node and destination MEC node as

$$T(n,o) = \frac{O_n}{B}. \tag{8}$$

3. Problem formulation

In this section, we present the mathematical expression of our optimization problem. As mentioned earlier, we focus on the task migration scheme with known trajectory under multiple users and multiple MEC nodes environment, in which deadlines of tasks are involved. We try to find a task migration scheme to maximize the number of tasks whose deadlines are guaranteed. For task J_n, we have $a_n^t = 0$ if it is completed before time slot t, thus completion time of task J_n is the sum of time slots when $a_n^t > 0$. It is given by

$$D(n) = \sum_{t=1}^{t=T} I\left\{a_n^t > 0\right\}\tau, \tag{9}$$

where τ is the length of one time slot. Similarly, $I\{x\}$ is an indicator function with $I\{x\} = 1$ if the event x is true and $I\{x\} = 0$ otherwise.

Note that our goal is to maximize the number of tasks whose deadlines are guaranteed, mathematically, which can be formulated as the following task migration management (M-M) problem

$$\text{maximize} \quad \sum_{n=1}^{n=N} I\left\{D(n) \le d_n\right\}, \tag{10}$$

$$\text{s.t.} \quad a_n^t \in \left\{0, s_n^t, h_n^t\right\}, \quad \forall n \in \mathcal{N}, \forall t \in \mathcal{T}. \tag{11}$$

Unfortunately, we find that this problem is NP-hard, which means that there is no polynomial-time algorithm to obtain the optimal solution unless $P = NP$. Next, proof of NP-hardness for M-M problem is presented.

Theorem 1. *M-M problem is NP-hard.*

Proof. To proceed, we first introduce the maximum cardinality bin packing problem (MCBPP) [28]: Given N items with sizes v_i and M bins of same capacity C, and the objective is to assign a maximum number of items to the fixed number of bins within the capacity constraint, which means finding a M partition $Q_1 \cup \cdots \cup Q_m$ of items set $\{1, 2, \ldots, N\}$ to reach the above objective. Mathematically, the MCBPP problem can be formulated as

$$\text{maximize} \quad \sum_{m\in\{1,2,\ldots,M\}} |Q_m|,$$

$$\text{s.t.} \quad \sum_{i\in Q_m} v_i \le C, \forall m \in \{1, 2, \ldots, M\}.$$

Refer to [28], we can know that the maximum cardinality bin packing problem above is NP-hard.

Here we consider a special instance of M-M problem, in which there are N mobile users. Each mobile user has a task, which consists of a set of tasks $\mathcal{N} = \{J_1, J_2, \ldots, J_n\}$ with same delay deadline d. Each mobile user moves across M heterogeneous MEC nodes with same R computing resources. We use r_i to denote the allocated computing resource of task J_i. $S_m \subseteq \mathcal{N}$ denotes the set of tasks which are completed within d on MEC node m. Then, the M-M problem can be formulated as

$$\text{maximize} \quad \sum_{m\in\{1,2,\ldots,M\}} |S_m|,$$

$$\text{s.t.} \quad \sum_{i\in S_m} r_i \le R, \forall m \in \{1, 2, \ldots, M\}.$$

This special instance of M-M problem corresponds to the MCBPP with capacity R. Therefore, M-M problem is NP-hard, which achieves the proof of NP-hardness for M-M problem. □

4. Group migration algorithm

Characteristics of the M-M problem are analyzed in Section 3, in which we have proved the NP-hardness of this problem. Thus it is impossible to obtain a solution in polynomial time. To solve the problem, we analyze three variants of this problem, depending on whether destination MEC nodes of tasks are same and whether source MEC nodes of tasks are same. Specifically, three variants include **Case1:** Different Source–Different Destination, **Case2:** Different Source–Same Destination, **Case3:** Same Source–Same Destination. By analyzing characteristics of these variants, plenty of theoretical analyzes are presented and significant algorithms are developed as a guideline to solve the M-M problem. Then we devise a group migration (GM) algorithm based on these theoretical analyzes and significant algorithms. The details of which are presented as follows.

4.1. Case1: Different source–different destination

In this case, we assume that source MEC nodes of all tasks are different and destination MEC nodes of all tasks are different. Besides, there is no intersection between all source MEC nodes and all destination MEC nodes. We find that there is no congestion of computing resources among multiple tasks, thus we can obtain an optimal solution in this case. The following theorem gives an optimal task migration scheme for **Case1**.

Theorem 2. *In **Case1**, the number of tasks whose delay deadlines are guaranteed $\sum_{n=1}^{n=N} I\{D(n) \le d_n\}$ is maximized, if task J_n ($n \in \{1, 2, \ldots, N\}$) is decided to migrate only when its completion time $Z_n^{t'}$ on destination MEC node is less than completion time Z_n^t on source MEC node in time slot t.*

Proof. We use $m \in \mathcal{M}$ and $m' \in \mathcal{M}$ to denote source MEC node and destination MEC node of task $J_k \in \mathcal{J}$ at time slot t, respectively. If migration decision of task J_k is executed on source MEC node m rather than migrated to destination MEC node m' for execution, then we can estimate completion time of task J_k in time slot t as

$$Z_k^t = \tau(t-1) + \frac{\gamma_k^t}{\frac{f_m}{\sum_{i\in\mathcal{N}} I\left\{a_i^t = a_k^t\right\}}} + \frac{O_k}{B} + \frac{O_k}{r_{down}(m',k)}, \tag{12}$$

where $\tau(t-1)$ is the sum of past time slots for executing task J_k, γ_k^t denotes how many remaining CPU cycles are required to accomplish task J_k at the beginning of time slot t, f_m denotes maximum computation frequency of source MEC node m, and the denominator denotes the number of tasks executed on source MEC node m in time slot t. In

Case1, source MEC nodes of all tasks are different and destination MEC nodes of all tasks are different, and there is no intersection between all source MEC nodes and all destination MEC nodes. Thus each MEC node executes one task at most in **Case1**. If user k is beyond the coverage of source MEC node when task J_k is completed, then output data O_k of the task should be transmitted from source MEC node m to destination MEC node m' via wire network, and B is the wire network bandwidth among MEC nodes. Finally, output data O_k of the task will be transmitted from destination MEC node m' to user k via wireless channel. Thus estimated completion time of task J_k is the sum of above time period.

On the contrary, if migration decision of task J_k is migrated from source MEC node m to destination MEC node m' for execution, then the corresponding completion time of task J_k can be estimated as

$$Z_k^{t'} = \tau(t-1) + \frac{\lambda_k}{B} + \frac{\gamma_k^t}{\frac{f_{m'}}{\sum_{i\in\mathcal{N}} I\left\{a_i^t = a_k^{t'}\right\}}} + \frac{O_k}{r_{down}(m',k)}. \tag{13}$$

Similarly, completion time of the task J_k includes the sum of past time slots $\tau(t-1)$ and execution time of task J_k on destination MEC node m', and transmission time of output data O_k from destination MEC node m' to user k via wireless channel. The difference is that task J_k will be migrated to destination MEC node m', thus input data λ_k needs to be migrated from source MEC node m to destination MEC node m'. Therefore, migration time of input data λ_k should be considered into completion time of task J_k.

In **Case1**, controller of MEC nodes can make a task migration decision by minimizing completion time of tasks. For task J_k, if completion time of the task executed on destination MEC node $Z_k^{t'}$ is less than completion time of the task executed on source MEC node Z_k^t, then task J_k will be decided to migrate from source MEC node m to destination MEC node m' for execution ($a_k^t \neq a_k^{t-1}$), otherwise, task J_k will be executed on the source MEC node m till it is completed ($a_k^t = a_k^{t-1}$). As we presented before, each task cannot be migrated to other MEC nodes except its destination MEC node for each time slot in our work. Note that there is no congestion of computing resources among multiple tasks in this case, thus completion time of each task is minimal with this migration scheme. Hence the number of tasks whose delay deadlines are guaranteed is maximized. Therefore, this proves the theorem and the result follows. □

The significance of Theorem 2 is that we can try to minimize completion time of tasks when devising a migration scheme, thus the number of tasks whose delay deadlines are guaranteed is maximized to some extent.

4.2. Case2: Different source–same destination

In this case, we assume that source MEC nodes of all tasks are different, but destination MEC node of all tasks is same. Similarly, there is no intersection between all source MEC nodes and the destination MEC node. However, **Case2** is similar to the M-M problem, which is also NP-hard. To solve it, we first define a threshold th_k for task J_k, which denotes maximum number of tasks executed on destination MEC node that task J_k can accept, if task J_k will be migrated from source MEC node to destination MEC node. According to Theorem 2, task J_k will be migrated from source MEC node m to destination MEC node m', only when completion time of the task executed on destination MEC node $Z_k^{t'}$ is less than completion time executed on source MEC node Z_k^t. Thus we can calculate threshold th_k of task J_k as

$$\tau(t-1) + \frac{\lambda_k}{B} + \frac{\gamma_k^t}{\frac{f_{m'}}{th_k}} + \frac{O_k}{r_{down}(m',k)} =$$
$$\tau(t-1) + \frac{\gamma_k^t}{\frac{f_m}{\sum_{i\in\mathcal{N}} I\left\{a_i^t = a_k^t\right\}}} + \frac{O_k}{B} + \frac{O_k}{r_{down}(m',k)}, \tag{14}$$

$$th_k = \left\lfloor \frac{f_{m'}\sum_{i\in\mathcal{N}} I\left\{a_i^t = a_k^t\right\}}{f_m} + \frac{(O_k-\lambda_k)f_{m'}}{B\gamma_k^t} \right\rfloor. \tag{15}$$

Note that th_k denotes the maximum number of tasks executed on destination MEC node that task J_k can accept, if task J_k will be migrated from source MEC node to destination MEC node. Therefore, th_k should be rounded down, $th_k = \left\lfloor \frac{f_{m'}\sum_{i\in\mathcal{N}} I\left\{a_i^t = a_k^t\right\}}{f_m} + \frac{(O_k-\lambda_k)f_{m'}}{B\gamma_k^t} \right\rfloor$. Algorithm 1 gives the task migration scheme in detail for **Case2**.

Algorithm 1 Case2-Migration Algorithm

Input: $\mathcal{A}^{t-1} = \left\{a_1^{t-1}, a_2^{t-1}, \ldots, a_n^{t-1}\right\}$, $\mathcal{J}$, cm'.
Output: $\mathcal{A}^t$.
1: **for** (task k from 1 to N) **do**
2: $th_k \leftarrow 0$, $a_k^t \leftarrow a_k^{t-1}$;
3: **if** ($a_k^{t-1} \neq 0$) **then**
4: $th_k \leftarrow \left\lfloor \frac{f_{m'}\sum_{i\in\mathcal{N}} I\left\{a_i^t = a_k^t\right\}}{f_m} + \frac{(O_k-\lambda_k)f_{m'}}{B\gamma_k^t} \right\rfloor$;
5: **end if**
6: **end for**
7: Sort tasks according to the sequence α such that $th_{\alpha_1} \geq th_{\alpha_2} \geq \cdots \geq th_{\alpha_N}$;
8: **for** (k from 1 to N) **do**
9: **if** ($cm' \leq th_{\alpha_k}$) **then**
10: $a_{\alpha_k}^t \leftarrow m'$;
11: $cm'++$;
12: **else**
13: break;
14: **end if**
15: **end for**
16: **return** $\mathcal{A}^t = \left\{a_1^t, a_2^t, \ldots, a_n^t\right\}$;

As shown in Algorithm 1, input parameters include task migration scheme $\mathcal{A}^{t-1}$ in time slot $t-1$ and information of tasks $\mathcal{J}$, and the number of tasks executed on destination MEC node cm'. We assume that migration scheme of task is same with the last time slot and we set initial value of th is equal to 0 (Step 2). For tasks which are not completed, we calculate their th values based on Eq. (15) (Steps 3–5). Then we sort tasks according to the sequence α such that $th_{\alpha_1} \geq th_{\alpha_2} \geq \cdots \geq th_{\alpha_n}$ (Step 6). Based on th values of tasks and cm' of destination MEC node, migration decisions of tasks are updated (Steps 8–15). Once task J_k is decided not to migrate from source MEC node to destination MEC node, then tasks whose th values are less than or equal to th_k will also be decided not to migrate, which will be proved to support Steps 8–15 in Algorithm 1.

Theorem 3. *In **Case2**, if task J_k is decided not to migrate from source MEC node to destination MEC node, then for task J_i whose th values are less than or equal to task J_k ($th_i \leq th_k$), completion of task J_i is less when it is decided not to migrate ($Z_i^t < Z_i^{t'}$).*

Proof. According to Algorithm 1, controller of MEC nodes decides not to migrate task J_k from source MEC node to destination MEC node, if and only if th_k is less than number of tasks executed on destination MEC node ($th_k < cm'$). Then the following inequality can be established as

$$\tau(t-1) + \frac{\lambda_k}{B} + \frac{\gamma_k^t}{\frac{f_{m'}}{th_k}} + \frac{O_k}{r_{down}(m',k)} <$$
$$\tau(t-1) + \frac{\lambda_k}{B} + \frac{\gamma_k^t}{\frac{f_{m'}}{cm'}} + \frac{O_k}{r_{down}(m',k)}. \tag{16}$$

As we mentioned before, $th_k = \left\lfloor \frac{f_{m'} \sum_{i \in \mathcal{N}} I\left\{a_i^t = a_k^t\right\}}{f_m} + \frac{(O_k - \lambda_k) f_{m'}}{B \gamma_k^t} \right\rfloor$. Thus we have

$$\tau(t-1) + \frac{\gamma_k^t}{\frac{f_m}{\sum_{i \in \mathcal{N}} I\left\{a_i^t = a_k^t\right\}}} + \frac{O_k}{B} + \frac{O_k}{r_{down}(m', k)} <$$

$$\tau(t-1) + \frac{\lambda_k}{B} + \frac{\gamma_k^t}{\frac{f_{m'}}{cm'}} + \frac{O_k}{r_{down}(m', k)}, \tag{17}$$

we can find that left-hand side of the inequality is Z_k^t and the right-hand side of the inequality is $Z_k^{t'}$. That is to say if $th_k < cm'$, then we can deduce that completion time of task J_k executed on source MEC node is smaller than completion time of task J_k executed on destination MEC node ($Z_k^t < Z_k^{t'}$). Thus for task J_i whose th value is smaller than task J_k, $th_i < th_k < cm'$, we can also deduce that $Z_i^t < Z_i^{t'}$, then task J_i will be decided not to migrate from source MEC node to destination MEC node. This proves Theorem 3. □

4.3. Case3: Same source–same destination

In this case, we assume that source MEC node of all tasks is same, and destination MEC node of all tasks is also same. Besides, the source MEC node is different from the destination MEC node. We find that **Case3** is also NP-hard, and there is no polynomial-time algorithm to obtain the optimal solution unless $P = NP$. Furthermore, congestion of computing resources for multiple tasks is more intensive in **Case3**. Fortunately, we can refer to Theorem 3 and Algorithm 1 in **Case2** to solve **Case3**. Algorithm 2 gives task migration scheme for **Case3** in detail.

Algorithm 2 Case3-Migration Algorithm

Input: $\mathcal{A}^{t-1} = \left\{a_1^{t-1}, a_2^{t-1}, \ldots, a_n^{t-1}\right\}$, $\mathcal{J}$, cm'.
Output: $\mathcal{A}^t$.
1: **for** (task k from 1 to N) **do**
2: $th_k \leftarrow 0$, $a_k^t \leftarrow a_k^{t-1}$;
3: **if** ($a_k^{t-1} \neq 0$) **then**
4: $th_k \leftarrow \left\lfloor \frac{f_{m'} \sum_{i \in \mathcal{N}} I\left\{a_i^t = a_k^t\right\}}{f_m} + \frac{(O_k - \lambda_k) f_{m'}}{B \gamma_k^t} \right\rfloor$;
5: **end if**
6: **end for**
7: Sort the tasks according to the sequence α such that $th_{\alpha_1} \geq th_{\alpha_2} \geq \cdots \geq th_{\alpha_N}$;
8: **while** True **do**
9: **if** ($cm' \leq th_{\alpha_1}$) **then**
10: $a_{\alpha_1}^t \leftarrow m'$, $cm'++$, $th_{\alpha_1} \leftarrow 0$;
11: **for** (k from 1 to N) **do**
12: **if** ($th_{\alpha_k} \neq 0$) **then**
13: Update th_{α_k};
14: **end if**
15: **end for**
16: Resort the tasks according to the sequence α such that $th_{\alpha_1} \geq th_{\alpha_2} \geq \cdots \geq th_{\alpha_N}$;
17: **else**
18: break;
19: **end if**
20: **end while**
21: **return** $\mathcal{A}^t = \left\{a_1^t, , a_2^t, \ldots, a_n^t\right\}$;

As you can see, Algorithm 2 refers to main idea of Algorithm 1 to a great degree. Input parameters of Algorithm 2 are identical to Algorithm 1. Similarly, we assume that migration decision of each task is same with the last time slot and we set initial value of th is equal to 0 (Step 2). For tasks which are not completed, we calculate their th values based on Eq. (15) (Steps 3–5). Then we sort tasks according to the sequence α such that $th_{\alpha_1} \geq th_{\alpha_2} \geq \cdots \geq th_{\alpha_n}$ (Step 6). Based on th values of tasks and cm' of destination MEC node, migration decisions of tasks are updated (Steps 8–18). Note that if task J_k is decided to migrate, then tasks whose th value is less than or equal to th_k will be influenced. That is because source MEC nodes of these tasks are same. Hence, we update th values of these tasks after task J_k is decided to migrate from source MEC node to destination MEC node, and resorting tasks according to th value in a descending order. Steps 8–18 in Algorithm 2 is supported by the above-mentioned Theorem 3.

Algorithm 3 Group Migration Algorithm

Input: $\mathcal{A}^{t-1} = \left\{a_1^{t-1}, a_2^{t-1}, \ldots, a_n^{t-1}\right\}$, $\mathcal{J}$, $\mathcal{P}$, $\mathcal{W}$.
Output: $\mathcal{A}^t$.
1: **for** (task k from 1 to N and $a_k^{t-1} \neq 0$) **do**
2: $a_k^t \leftarrow a_k^{t-1}$;
3: Divide task k into corresponding groups according to its destination MEC node i.e., $group_1, group_2, \ldots, group_p$;
4: **end for**
5: **for** (group j from 1 to p) **do**
6: **if** (source MEC nodes of tasks in group j are different) **then**
7: Case2-Migration Algorithm(group j);
8: **else**
9: Further dividing group j into subgroups according to source MEC nodes of tasks, and tasks whose source MEC nodes are different consist of the last subgroup i.e., $subgroup_1, subgroup_2, \ldots, subgroup_q$;
10: Case2-Migration Algorithm(subgroup q);
11: **for** (subgroup i from 1 to $q-1$) **do**
12: Case3-Migration Algorithm(subgroup i);
13: **end for**
14: **end if**
15: **end for**
16: **return** $\mathcal{A}^t = \left\{a_1^t, a_2^t, \ldots, a_n^t\right\}$;

4.4. Group migration algorithm for M-M problem

However, as described in M-M problem, we need to consider a more realistic scenario, in which there is an intersection between all source MEC nodes and destination MEC nodes. Thus M-M problem is more complicated to solve. Fortunately, based on plenty of theoretical analyzes and algorithms presented before, we can devise a group migration (GM) algorithm to solve the M-M Problem. As shown in Fig. 2, mobile users with resource-limited devices offload their latency-sensitive tasks to MEC nodes for execution. Depending on destination MEC nodes of tasks, we divide tasks into different groups. For each group, if source MEC nodes of all tasks are different, then we can perform Algorithm 1 to solve it. For groups in which source MEC nodes of some tasks are same, we further divide these groups into subgroups depend on source MEC nodes of tasks. For subgroups in which source MEC nodes of all tasks are same and destination MEC nodes of all tasks are same, Algorithm 2 is proposed to solve them. The details of GM algorithm are presented as follows.

As shown in Algorithm 3, input parameters are similar to Algorithm 1 and Algorithm 2 we presented before. But there are two additional parameters, including a set of destination MEC nodes $\mathcal{P}$ and a set $\mathcal{W}$ which records the number of tasks executed on destination MEC nodes. Based on theoretical analyzes and algorithms we presented before, we first divide tasks into different groups according to destination MEC nodes of them. For group which source MEC nodes of all tasks are different, note that this scenario is identical to **Case2**. Thus we can perform Algorithm 1 to obtain an optimal solution for this scenario. For group which there are some tasks whose source MEC nodes are same, we can further divide the group into subgroups according to source

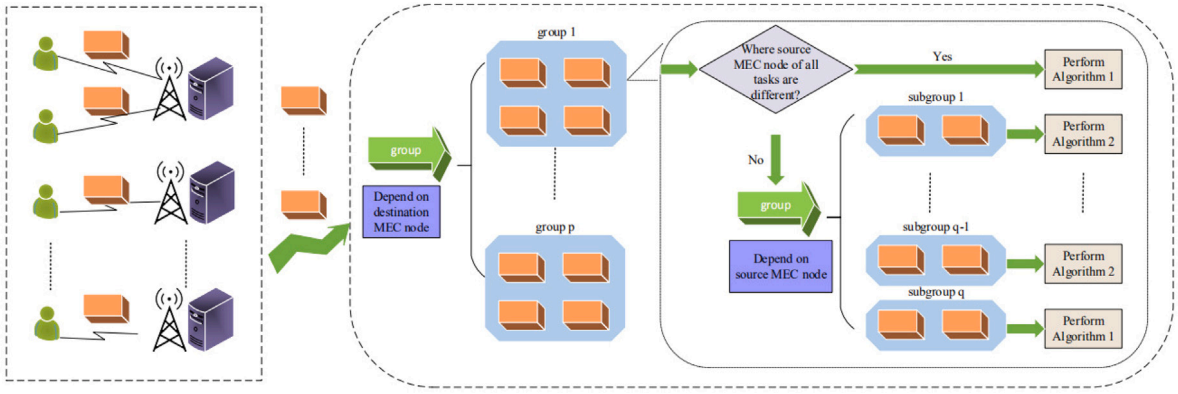

Fig. 2. An illustration of proposed GM algorithm.

MEC nodes of tasks, and tasks with different source MEC nodes consist of last subgroup. Note that the last subgroup and other subgroups are same to **Case2** and **Case3**, respectively. Thus we can perform Algorithm 1 and Algorithm 2 to solve them. As for information collection of GM algorithm, at the first time slot, MEC nodes send information of tasks executed on them to central controller, including input data size, workload, output data size, and deadline. Then MEC nodes only need to send γ_n^t that denotes how many remaining CPU cycles are required for task J_n to central controller in the next time slots. In addition, the mobility information of mobile users is obtained by an extra module that provides trajectories of mobile users.

Next, time complexity of GM algorithm is presented. As shown in Algorithm 3, we need to make a grouping operation for each task according to its destination MEC node, which requires time complexity $\Theta(N)$ (Steps 1–4). N is the number of tasks of all mobile users. Then we make migration decisions for p groups in turn, thus number of iterations of outer "for" loop (Steps 5–15) is $\Theta(p)$. p is the number of groups, and we use g to denote the number of tasks for each group. For each iteration, we perform Algorithm 1 for groups in which source MEC nodes of all tasks are different, and time complexity required by Algorithm 1 is $\Theta(2g)$. As we said before, we use g to denote the number of tasks for each group, thus the value of N in Algorithm 1 and Algorithm 2 is g. Specially in Algorithm 1, there are two "for" loops with g operations, therefore, and time complexity required by Algorithm 1 is $\Theta(2g)$. Besides, for groups in which source MEC nodes of some tasks are same, we further divide the group into q subgroups according to source MEC node of tasks. For simplification, we also use g to denote the overrated number of tasks for each subgroup. For the first $q-1$ subgroups in which source MEC nodes of all tasks are same, then we can perform Algorithm 2 to solve it. There are $q-1$ subgroups, thus the outer loop is $\Theta(q-1)$. Specially in Algorithm 2, at first, there is a "for" loop with g operations. Then migration decisions are made for each tasks in subgroups. There is a "while" loop with a nested "for" loop. Thus the time complexity required by Algorithm 2 is $\Theta(g^2)$. For the last q subgroups in which source MEC nodes of all tasks are different, then we can perform Algorithm 1 to solve it with $\Theta(2g)$ complexity. Hence we can estimate the time complexity of GM algorithm is $\Theta(N + p(2g + (q-1)\cdot g^2 + 2g))$.

5. Performance evaluation

In this section, we conduct numerical studies to illustrate performance of the proposed grouping migration algorithm.

5.1. Parameter configuration

We consider an MEC environment with 16 MEC nodes which provide wireless stations and computation services. As shown in Table 3, MEC node occupies $500 * 500$ m^2 and CPU frequency f_m of MEC node is uniformly selected from the set $\{0.1, 0.2, \ldots, 1.0\}$ GHz [29]. There are N mobile users endowed with a resource-limited device randomly moving across multiple MEC nodes, where N is varied from 20 to 120. For trajectories of mobile users, actually, there is no real moving trajectories data set that matches task data set. Thus refer to [11] and [21], trajectories of mobile users are generated by random walk model in this paper.

Table 3
Experiment parameters.

Parameters	(Fixed)–[Varied range] (Increment)
Number of mobile user (N)	(30)–[20, 100] (20)
Number of MEC node (M)	16
Input data size (λ_n)	[100,500]–[100, 250-700] (150) KB
Output data size (O_n)	[100,150]–[100, 150-600] (150) KB
Workload (c_n)	[800,2400]–[800, 1600-6400] (1600) cycles/bit
Deadline of task (d_n)	[5,50]–[5, 25-70] (15) s
MEC CPU frequency (f_m)	$\{0.1, 0.2, \ldots, 1.0\}$ GHz
Length of time slot	5 s
Wireless bandwidth W_m	1 Mbps
Channel gain $H_{m,n}$	10^{-6}
Transmission power p_n	1 W
White noise power θ^m	10^{-9} W
Wire bandwidth B	8 Mbps

As we described before, each mobile user is associated with a computation-intensive task to be offloaded to nearby MEC node. For a task, input data size λ_n is uniformly selected from $[a_1, b_1]$ KB, where a_1 is fixed at 100 and b_1 varies from 250 to 700. Similarly, output data size O_n is uniformly selected from $[a_2, b_2]$ KB, where a_2 is fixed at 100 and b_2 varies from 150 to 600. The workload requirement c_n (in CPU cycles/bit) is uniformly selected from $[a_3, b_3]$ CPU cycles/bit, where a_3 is fixed at 800 and b_3 varies from 1600 to 6400. Similarly, deadline d_n of the task also follows a uniform distribution with $d_n \in [a_4, b_4]$ s, where a_4 is fixed at 5 and b_4 varies from 25 to 70. The length of a time slot τ is 5 s. As suggested in [30], wireless channel gain $H_{m,n}$ is 10^{-6}. Besides, white noise power $\theta^m = 10^{-9}$ W [10], wireless channel bandwidth W_m=1 Mbps, transmission power of mobile user p_n=1 W, network bandwidth among MEC nodes B=8 Mbps.

5.2. Performance benchmark

To our knowledge, few works have considered user mobility in MEC. Due to different environment characteristics and performance metric, related work mentioned before is not appropriate to compare with GM algorithm(see Table 1). Hence like related work [18,19,22,31] we presented before, we also introduce three heuristics benchmark algorithms to evaluate the performance of our proposed GM algorithm.

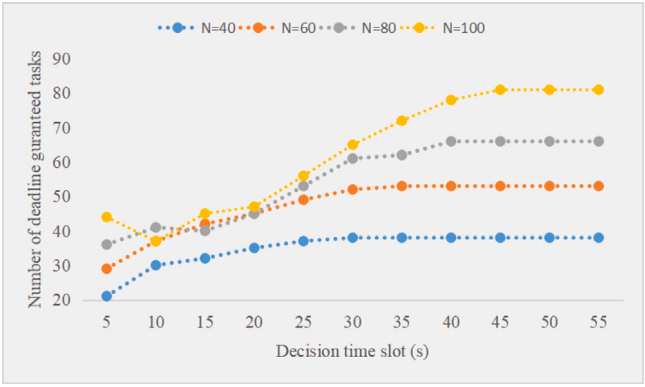

Fig. 3. Dynamics of the number of tasks whose deadlines are guaranteed with varied decision time slot.

1) **Not Migrating (NM)**: task will not be migrated wherever mobile user is moving to. In other words, task will be executed on source MEC node till it is completed. 2) **Always Migrating (AM)**: when mobile user is beyond the coverage of source MEC node, task will be migrated to destination MEC node according to the predicted mobility information of mobile user. 3) **Cold Treatment (CT)**: when mobile user is beyond the coverage of source MEC node, task's migration decision is made without considering the resource congestion among multiple tasks, specifically, each task makes a migration decision only based on other tasks' information in last time slot.

5.3. Random results

We first show dynamics of the number of tasks whose deadlines are guaranteed with increased decision time slot in Fig. 3. Specifically, we show the convergence of proposed GM algorithm with number of mobile users N varying from 40 to 100. We can find that number of tasks whose deadlines are guaranteed can converge to a stable point with increased decision time slot when GM algorithm is performed. Furthermore, the smaller number of mobile users, the faster that GM algorithm converges. It is normal that there is a performance degradation in the early stages of decision time slots. The reason lies in that proposed algorithm is a global target-oriented ground algorithm. To sum up, the proposed GM algorithm has a satisfactory convergence speed.

In the following experiments, we randomly generate 100 sets of data and show the average number, minimal number, and maximal number of tasks whose deadlines are guaranteed. We first show dynamics of number of tasks whose deadlines are guaranteed with varied input data sizes of tasks in Fig. 4. The number of mobile user is fixed at 30. With fixed the number of MEC nodes and CPU frequencies of MEC nodes, we increase input data sizes of tasks with an increment 150 KB. We can observe that the number of tasks whose deadlines are guaranteed is decreasing with increased input data sizes of tasks. That is because computation and communication time are increasing with increased input data sizes of tasks. However, deadlines of tasks are remaining unchanged. Therefore, the number of tasks whose deadlines are guaranteed is decreased. Nevertheless, compared with AM and NM and CT algorithms, the proposed GM algorithm can achieve up to 60% performance improvement.

While other variables remain unchanged, we also increase output data sizes of tasks to assess its impact on the number of tasks whose deadlines are guaranteed. As shown in Fig. 5, we can observe that the number of tasks whose deadlines are guaranteed is also decreased with increased output data sizes of tasks. The reason lies in that communication times of tasks are increasing with increased output data sizes of tasks. Nevertheless, deadlines of tasks remain unchanged.

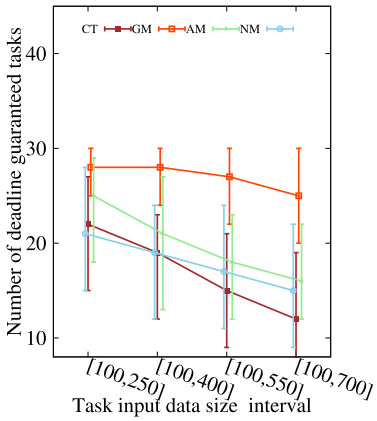

Fig. 4. The number of tasks whose deadlines are guaranteed with the variation of input data size.

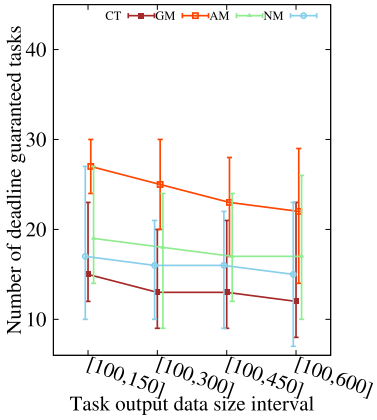

Fig. 5. The number of tasks whose deadlines are guaranteed with the variation of output data size.

Thus the number of tasks whose deadlines are guaranteed is decreased. However, the proposed GM algorithm always outperforms the other two benchmark algorithms with the variation of output data size.

Fig. 6 presents the experiment results with varied workload requirements. The number of mobile user is also fixed at 30. We increase workload requirements of tasks from $[800, 1600]$ to $[800, 6400]$ (in CPU cycles/bit) with an increment 1600. We can observe that the number of tasks whose deadlines are guaranteed is also decreased with increased workload requirements of tasks. That is because both number of MEC nodes and CPU frequencies of MEC nodes remain unchanged. Thus completion times of tasks are increased with the increase of workload requirement. However, deadlines of tasks also remain unchanged. Hence, the number of tasks whose deadlines are guaranteed is decreased with increased workload requirements of tasks. As the picture shows, our proposed GM algorithm always outperforms AM and NM and CT algorithms.

On the premise of other variables remain unchanged, we also change deadlines of tasks to assess the impact on the number of tasks whose deadlines are guaranteed. We increase deadlines of tasks from $[5, 25]$ to $[5, 70]$ s with an increment 15. Fig. 7 presents experimental results with varied deadlines of tasks. From the figure, we can observe that the number of tasks whose deadlines are guaranteed tends to increase with increased deadlines of tasks. However, no matter how

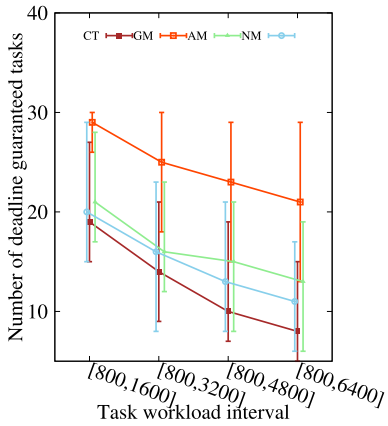

Fig. 6. The number of tasks whose deadlines are guaranteed with the variation of workload requirement.

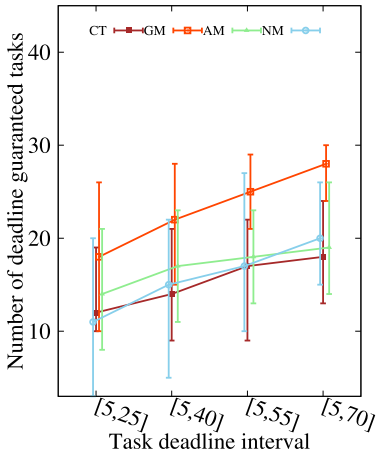

Fig. 7. The number of tasks whose deadlines are guaranteed with the variation of deadline.

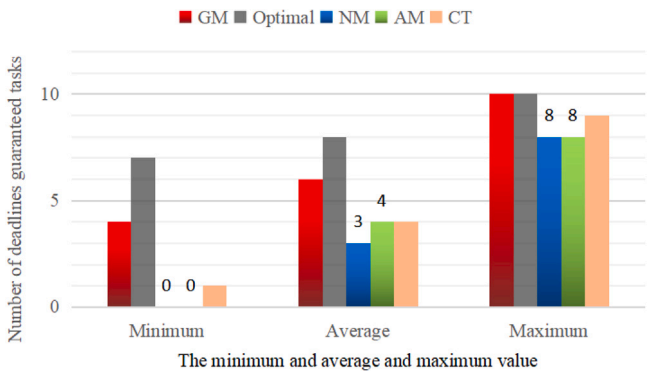

Fig. 8. The number of tasks whose deadlines are guaranteed.

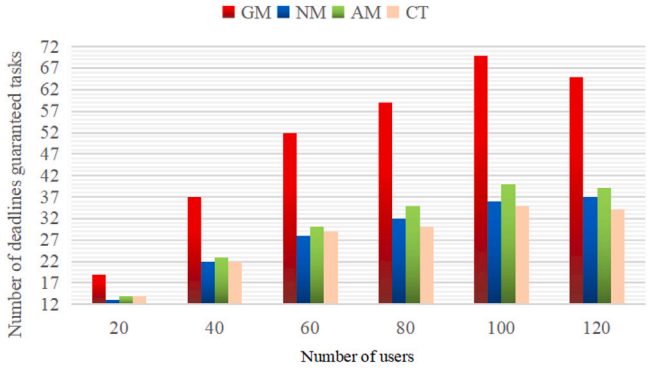

Fig. 9. The number of tasks whose deadlines are guaranteed with varied number of users.

interval of deadline varies, the proposed GM algorithm always outperforms AM and NM and CT algorithms, which illustrates the advantage of our algorithm to satisfy delay deadlines of tasks.

Besides, we present the performance gap between the proposed GM algorithm and optimal solution. The optimal solution is obtained by brute-force method, and the time complexity is 2^{NT}, which increases exponentially with the increased number of task and time slot. Thus we only compare the results of small scale problem when number of user is 10, specifically, we present the minimum and average and maximum value of the number of tasks whose deadlines are guaranteed for running 100 dataset. And CPU frequency set of MEC node is $\{0.1, 0.2, \ldots, 0.5\}$ GHz. From Fig. 8, we find that results of proposed GM algorithm are near the optimal solution, compared to other algorithms.

Furthermore, Fig. 9 presents experimental results with different numbers of mobile user and fixed CPU frequency set $\{0.1, 0.2, \ldots, 1.0\}$ GHz of MEC node. The number of mobile users varies from 20 to 120 with an increment 20. In this figure, we only present the average number of tasks whose deadlines are guaranteed with running 100 task datasets. With increased number of mobile users, congestion for computing resources among mobile users becomes more intensive. However, both number of MEC nodes and CPU frequencies of MEC nodes remain unchanged, thus it is reasonable that performance improvement of proposed GM algorithm with more number of mobile users is lower than less number of mobile users. Similarly, compared with AM and NM and CT algorithms, the proposed GM algorithm can make a pretty high performance improvement no matter how the number of mobile users varies.

On the other hand, we also change number of mobile users under varied CPU frequencies of MEC nodes to observe their common influences on number of deadlines guaranteed tasks. Number of mobile users also varies from 20 to 120 with an increment 20. CPU frequency set of MEC node varies from $[0.1, 0.5]$ to $[0.6, 1.0]$ GHz. From Fig. 10, we find that the number of tasks whose deadlines are guaranteed tends to increase with increased CPU frequencies of MEC node. The reason lies in that completion times of tasks are decreased with increased CPU frequencies of MEC nodes. However, deadlines of tasks remain unchanged. Therefore, the number of tasks whose deadlines are guaranteed tends to increase with increased CPU frequencies of MEC nodes. As shown in Fig. 10(a), GM algorithm has a few performance improvements when number of mobile users is 20. That is because computing resources are abundant for in this case. However, we can observe that performance improvement of GM algorithm grows with increased number of mobile users and increased CPU frequencies of MEC nodes in Fig. 10(b)–10(f).

Fig. 11 presents the average number of tasks whose deadlines are guaranteed under different CPU frequency sets of MEC nodes. As shown in Fig. 11(a), the number of tasks whose deadlines are guaranteed presents a trend of parabola with increased number of mobile users when GM algorithm is performed. The reason lies in that extremely

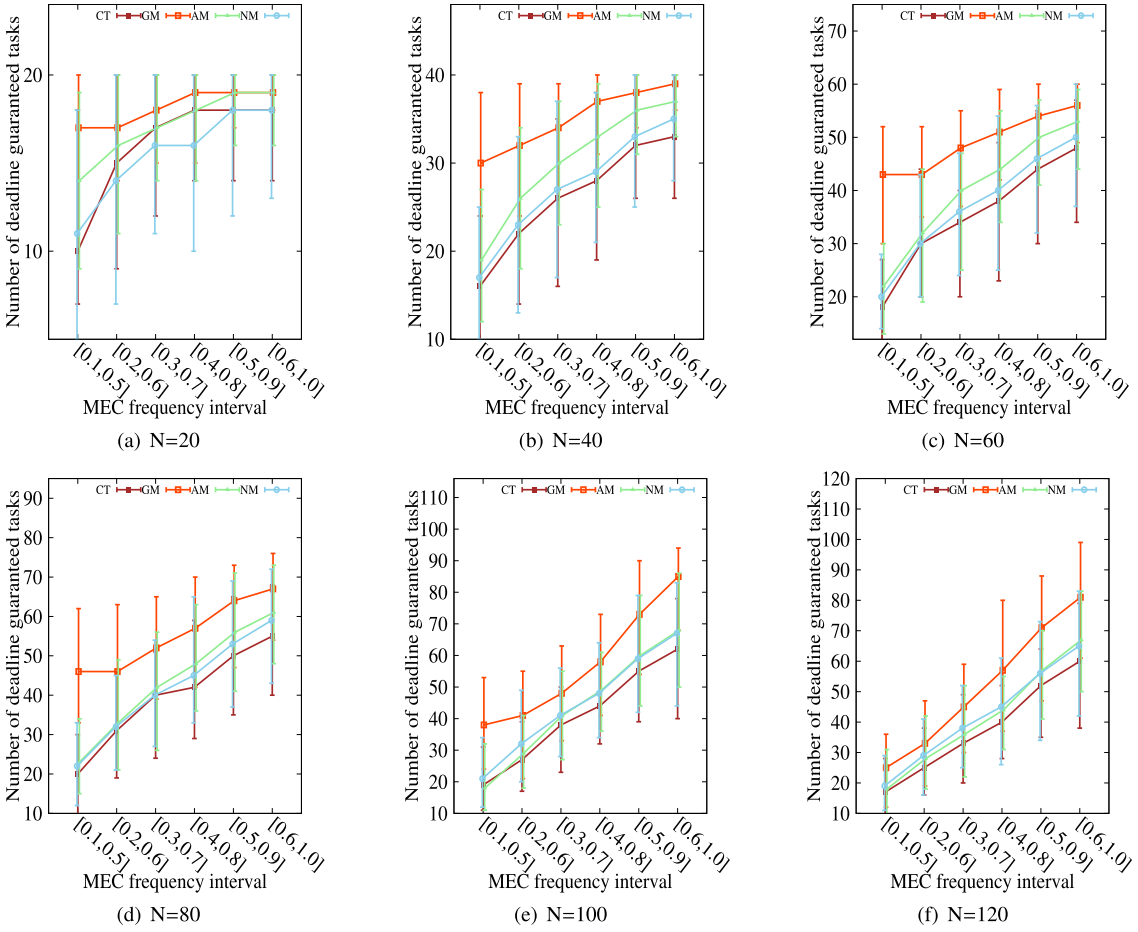

Fig. 10. The number of tasks whose deadlines are guaranteed with varied CPU frequencies of MEC nodes under different numbers of users.

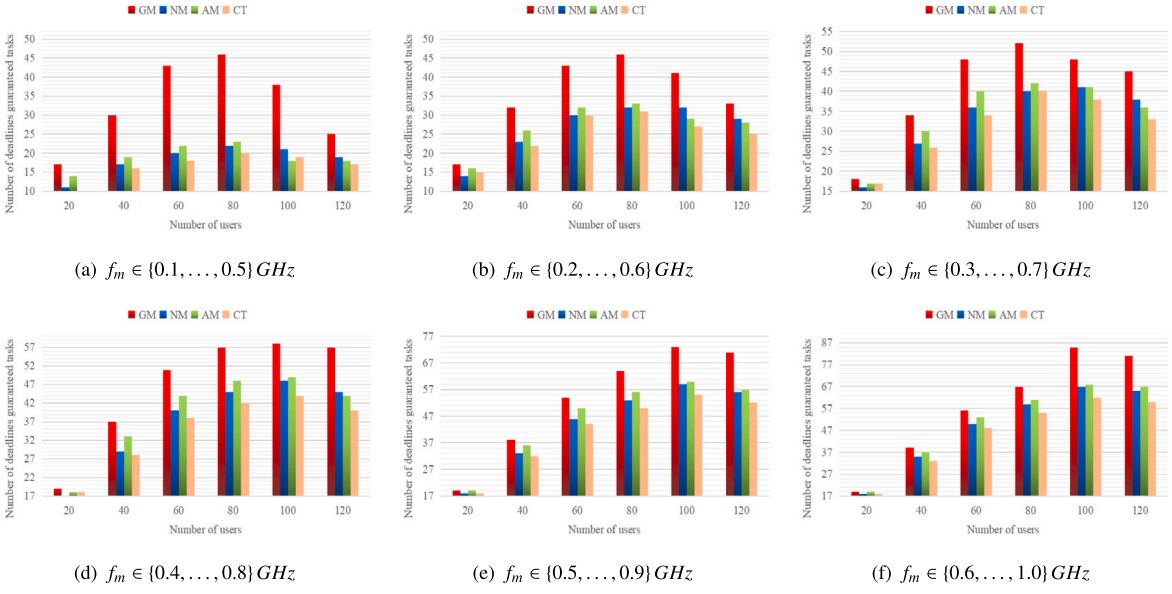

Fig. 11. The number of tasks whose deadlines are guaranteed with varied number of users under different CPU frequency sets of MEC nodes.

limited computing resources are not enough to execute too many tasks of mobile users. In Fig. 11(b)–11(f), the trend of parabola is gone be disappeared with increased higher CPU frequencies of MEC nodes. As shown in Fig. 11(f), compared with other two benchmark algorithms, performance improvement of the proposed GM algorithm is low when number of mobile users is small. That is because computing resources are pretty enough to execute corresponding tasks. Task migration scheme has no obvious effect on performance improvement in this case. However, the proposed GM algorithm always outperforms AM and NM and CT algorithms in the above-mentioned simulations. To sum up, all experimental results that verify advantages of the proposed GM algorithm.

6. Conclusion and future work

In this paper, we addressed the task migration problem with user mobility consideration under multiple users and multiple MEC nodes environment, in which deadlines of tasks are involved. By analyzing properties of three variants of this problem, plenty of theoretical analyzes were presented and significant algorithms had been proposed as a guideline to solve it. We proposed a group migration (GM) algorithm with known trajectories of users. Our goal was to maximize the number of tasks whose deadlines are guaranteed. Extensive experiments were carried out to evaluate the performance of GM algorithm. Compared with always migrating (AM) and not migrating (NM) and cold treatment (CT) algorithms, the results confirmed that GM algorithm can achieve up 35%-75% performance improvement.

As part of future direction, we plan to extend the existing work by considering computation offloading decision-making simultaneously in current scenario.

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

The authors would like to express their gratitude to the anonymous reviewers whose constructive comments have greatly helped to improve this manuscript. This work was partially supported by the National Key Research and Development Program of China (2018YFB1701403), Programs of National Natural Science Foundation of China (Grant Nos. U19A2058, 61702170). This work was also sponsored by Zhijiang Lab, China (No. 2020KE0AB01).

References

[1] T.Q. Dinh, J. Tang, Q.D. La, T.Q. Quek, Offloading in mobile edge computing: Task allocation and computational frequency scaling, IEEE Trans. Commun. 65 (8) (2017) 3571–3584.

[2] M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta, A. Neal, et al., Mobile-edge computing introductory technical white paper, in: White Paper, Mobile-Edge Computing (MEC) Industry Initiative, 2014, pp. 1089–7801.

[3] F. Wang, J. Xu, X. Wang, S. Cui, Joint offloading and computing optimization in wireless powered mobile-edge computing systems, IEEE Trans. Wireless Commun. 17 (3) (2017) 1784–1797.

[4] 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.

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

[6] A. Yousefpour, C. Fung, T.T. Nguyen, K.P. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, J.P. Jue, All one needs to know about fog computing and related edge computing paradigms: A complete survey, J. Syst. Archit. 98 (2019) 289–330.

[7] N. Abbas, Y. Zhang, A. Taherkordi, T. Skeie, Mobile edge computing: A survey, IEEE Internet Things J. 5 (1) (2017) 450–465.

[8] X. Chen, L. Jiao, W. Li, X. Fu, Efficient multi-user computation offloading for mobile-edge cloud computing, IEEE/ACM Trans. Netw. 24 (5) (2015) 2795–2808.

[9] C. Liu, K. Li, J. Liang, K. Li, COOPER-MATCH: Job offloading with a cooperative game for guaranteeing strict deadlines in MEC, IEEE Trans. Mob. Comput. (2019).

[10] 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), IEEE, 2016, pp. 1451–1455.

[11] X. Sun, N. Ansari, Latency aware workload offloading in the cloudlet network, IEEE Commun. Lett. 21 (7) (2017) 1481–1484.

[12] 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), IEEE, 2017, pp. 1–6.

[13] 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.

[14] X. Jiang, N. Guan, X. Long, Y. Tang, Q. He, Real-time scheduling of parallel tasks with tight deadlines, J. Syst. Archit. 108 (2020) 101742.

[15] C. Li, M. Song, H. Tang, Y. Luo, Offloading and system resource allocation optimization in TDMA based wireless powered mobile edge computing, J. Syst. Archit. 98 (2019) 221–230.

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

[17] Y. Sun, S. Zhou, J. Xu, EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks, IEEE J. Sel. Areas Commun. 35 (11) (2017) 2637–2646.

[18] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, K.K. Leung, Dynamic service placement for mobile micro-clouds with predicted future costs, IEEE Trans. Parallel Distrib. Syst. 28 (4) (2016) 1002–1016.

[19] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, K.K. Leung, Dynamic service migration in mobile edge-clouds, in: 2015 IFIP Networking Conference (IFIP Networking), IEEE, 2015, pp. 1–9.

[20] A. Nadembega, A.S. Hafid, R. Brisebois, Mobility prediction model-based service migration procedure for follow me cloud to support QoS and QoE, in: 2016 IEEE International Conference on Communications (ICC), IEEE, 2016, pp. 1–6.

[21] J. Plachy, Z. Becvar, E.C. Strinati, Dynamic resource allocation exploiting mobility prediction in mobile edge computing, in: 2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), IEEE, 2016, pp. 1–6.

[22] T. Ouyang, Z. Zhou, X. Chen, Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing, IEEE J. Sel. Areas Commun. 36 (10) (2018) 2333–2345.

[23] T. Taleb, A. Ksentini, P.A. Frangoudis, Follow-me cloud: When cloud services follow mobile users, IEEE Trans. Cloud Comput. 7 (2) (2019) 369–382.

[24] M. Srivatsa, R. Ganti, J. Wang, V. Kolar, Map matching: Facts and myths, in: Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM, 2013, pp. 484–487.

[25] P. Mach, Z. Becvar, Mobile edge computing: A survey on architecture and computation offloading, IEEE Commun. Surv. Tutor. 19 (3) (2017) 1628–1656.

[26] A. Hadachi, O. Batrashev, A. Lind, G. Singer, E. Vainikko, Cell phone subscribers mobility prediction using enhanced Markov chain algorithm, in: 2014 IEEE Intelligent Vehicles Symposium Proceedings, IEEE, 2014, pp. 1049–1054.

[27] A. Nadembega, A. Hafid, T. Taleb, A destination and mobility path prediction scheme for mobile networks, IEEE Trans. Veh. Technol. 64 (6) (2014) 2577–2590.

[28] K.-H. Loh, B. Golden, E. Wasil, Solving the maximum cardinality bin packing problem with a weight annealing-based algorithm, in: Operations Research and Cyber-Infrastructure, Springer, 2009, pp. 147–164.

[29] C. You, K. Huang, H. Chae, B.-H. Kim, Energy-efficient resource allocation for mobile-edge computation offloading, IEEE Trans. Wireless Commun. 16 (3) (2016) 1397–1411.

[30] J. Guo, Z. Song, Y. Cui, Z. Liu, Y. Ji, Energy-efficient resource allocation for multi-user mobile edge computing, in: GLOBECOM 2017-2017 IEEE Global Communications Conference, IEEE, 2017, pp. 1–7.

[31] S. Wang, R. Urgaonkar, T. He, M. Zafer, K. Chan, K.K. Leung, Mobility-induced service migration in mobile micro-clouds, in: 2014 IEEE Military Communications Conference, IEEE, 2014, pp. 835–840.

Fan Tang received the B.S. degree in computer science and technology from Huaqiao University, China, in 2018. She is currently working toward the M.S. degree at Hunan University, China. Her research interests are mainly in cloud computing, edge computing, parallel computing, reinforcement learning, and game theory.

Chubo Liu received the B.S. degree and Ph.D. degree 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 at Hunan University. His research interests are mainly in game theory, approximation and randomized algorithms, cloud and edge computing. He has published over 25 papers in journals and conferences such as the *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Industrial Informatics*, *IEEE Internet of Things Journal*, *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, *Theoretical Computer Science*, ICPADS, HPCC, and NPC. He won the Best Paper Award in IFIP NPC 2019 and the IEEE TCSC Early Career Researcher(ECR) Award in 2019. He is a member of IEEE and CCF.

Kenli Li received the Ph.D. degree in computer science from Huazhong University of Science and Technology, China, in 2003. He was a visiting scholar at University of Illinois at Urbana-Champaign from 2004 to 2005. He is currently the dean and a full professor of computer science and technology at Hunan University and deputy 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 230 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 Computing*, ICPP, and CCGrid. He is an outstanding member of CCF. He is a senior member of the IEEE and serves on the editorial board of *IEEE Transactions on Computers*.

Zhuo Tang received the Ph.D. in computer science from Huazhong University of Science and Technology, China, in 2008. He is currently a full professor of the College of Computer Science and Electronic Engineering at Hunan University, and is the associate chair of the department of computing science. His majors are distributed computing system, cloud computing, and parallel processing for big data, including distributed machine learning, security model, parallel algorithms, and resources scheduling and management in these areas. He is a member of ACM and CCF.

Keqin Li is a SUNY Distinguished Professor of computer science. His current research interests include parallel computing and high performance computing, distributed computing, energy-efficient computing and communication, heterogeneous computing systems, cloud computing, big data computing, CPU–GPU hybrid and cooperative computing, multicore computing, storage and file systems, wireless communication networks, sensor networks, peer-to-peer file sharing systems, mobile computing, service computing, Internet of things and cyber–physical systems. He has published over 630 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He is currently serving 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 a fellow of the IEEE.