Task Scheduling in Deadline-Aware Mobile Edge Computing Systems

Tongxin Zhu, Tuo Shi, Jianzhong Li, Zhipeng Cai, *Senior Member, IEEE*, and Xun Zhou

***Abstract*—Mobile edge computing (MEC) is a new computing approach in which computation tasks carried by mobile devices (MDs) can be offloaded to MEC servers or computed locally. Since the MDs are always battery limited and computation tasks have strict deadlines, how to schedule the execution of each task energy effectively is important. Comparing with existing works, we consider a much more complexed scenario, in which multiple moving MDs sharing multiple heterogeneous MEC servers, and a problem named as minimum energy consumption problem in deadline-aware MEC system is formulated. Such problem is proved to be NP-hard, and two approximation algorithms are proposed focusing on single and multiple MD scenarios, respectively. The performances of these algorithms are varied by theoretical analysis and simulations.**

***Index Terms*—Edge computing, schedules.**

I. Introduction

With the rapid development of mobile devices (MDs), more and more computation-intensive and latency-sensitive mobile applications are popular [1], such as augmented reality [2], speech recognition [3], [4], face recognition [5]–[7] and some crowdsensing applications [8], etc. These applications require intensive computation and energy resources to provide services. However, the computation and energy resources of an MD are constrained by its physical size and battery life. As a consequence, the resource-constrained MDs cannot satisfy the high quality of service requirement of these applications.

Mobile cloud computing [9] is an efficient approach to overcome this challenge by offloading some computation tasks of an MD to resource-rich cloud infrastructures. Nevertheless, the geographical distances between MDs and cloud infrastructures are quite far in general, which leads to huge communication latency. Since most mobile applications are latency-sensitive, the huge communication latency of mobile cloud computing will reduce the quality of services of these applications and is intolerant for application users. To reduce the communication latency, enormous edge devices with relatively less resources but shorter distances compared with cloud infrastructures in the network are utilized. That is, MDs can offload their latency-sensitive computation tasks to nearby edge devices through wireless communication. Then, edge devices perform the received computation tasks with their own constrained resources or transmit these tasks to resource-rich cloud infrastructures by fast and low-latency connection (e.g., via fiber transmission). Finally, edge devices transmit the computation results to MDs through wireless communication. This paradigm is called as the mobile edge computing (MEC) [10], and the edge devices are called as MEC servers.

In an MEC system, the foundational problem is to design the offloading mechanism, which makes schedule for each computation task in the MEC system, including the MEC server selection and time selection for the MD to offload the computation task. The offloading mechanism for all tasks of MDs is called as a task schedule of all tasks in an MEC system. In the following, we use offloading mechanism and task schedule interchangeably. The designed task schedule should both satisfy the latency constraint of each computation task and reduce the energy consumption for processing and communication, which is significantly challenging.

There are many existing works focusing on task scheduling [11]–[24], however, these works do not consider the mobility of the MDs. In this paper, we focus on the complexed scenario, in which moving MDs sharing multiple heterogeneous MEC servers. The main challenges are summarized as follows.

1) Since MDs and MEC servers communicate with each other through wireless communication, an MD and an MEC server can communicate with each other only when the MD is in the communication range of the MEC server. Besides, the distance between an MD and an MEC server will affect the communication rate between them. As an MD moves, the set of MEC servers that can communicate with the MD and the communication rates between the MD and these MEC servers will change.
2) The diversity of MEC servers should be considered. The MEC servers are edge devices in the network, e.g., tablet PC, laptop and desktop computer, etc., which are heterogeneous in computation and communication capacity. Accordingly, the energy consumption and latency of computation and communication for offloading the same

Manuscript received July 20, 2018; revised September 1, 2018; accepted October 1, 2018. Date of publication October 9, 2018; date of current version June 19, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61632010, Grant 61502116, Grant U1509216, and Grant 61370217 and in part by the National Science Foundation under Grant 1252292, Grant 1741277, and Grant 1704287. *(Corresponding author: Jianzhong Li.)*

T. Zhu, T. Shi, and J. Li are with the Department of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China (e-mail: zhutongxin@stu.hit.edu.cn; shituo@hit.edu.cn; lijzh@hit.edu.cn).

Z. Cai is with the Department of Computer Science, Georgia State University, Atlanta, GA 30303 USA (e-mail: zcai@gsu.edu).

X. Zhou is with the Department of Management Sciences, University of Iowa, Iowa City, IA 52242 USA (e-mail: xun-zhou@uiowa.edu).

Digital Object Identifier 10.1109/JIOT.2018.2874954

task to distinct MEC servers are different. Therefore, the diversity of heterogeneous MEC servers should be considered in server selection of the offloading mechanism.

3) The latency sensitivities and computing demands of distinct computation tasks are diverse. In consequence, offloading different computation tasks to the same MEC server will have different influences on the performance of an offloading mechanism. Therefore, the diversity of latency sensitivity and computing demand of computations tasks should be considered in the processing order of tasks in an offloading mechanism.
4) The simultaneously wireless communications between MDs and MEC servers may interfere with each other. Besides, the computation ability of MEC servers are heterogeneous and limited. Therefore, the communication time of different pairs of MDs and MEC servers and the computation time of different tasks on each MEC server should be elaborately designed in the task schedule to avoid interferences.

To address these challenges, a task schedule indicating when and which computation tasks are offloaded to MEC servers is studied in this paper. Interference-free task schedules with minimum energy consumption under the deadline constrains of all computation tasks are designed by jointly considering the mobility of MDs, the resources of heterogeneous MEC servers, and computation demands of tasks. Both scenarios of single MD and multiple MDs are considered in this paper. Specially, we optimize the scheduling of all computation tasks to minimize the maximum energy consumption of both MDs and MEC servers and satisfy the deadline of each task as well. The main contributions of this paper are as follows.

1) We first consider the mobility of MDs and formally define the minimum energy consumption problem in deadline-aware MEC system problem (MEC-DMEC), which aims at minimizing the energy consumption of both MDs and MEC servers with the deadline constraint of each computation task.
2) The NP-hardness of MEC-DMEC has been proved, and two approximation algorithms are proposed for the scenarios of single MD and multi-MDs, respectively, i.e., the local-based partial reasonable task schedule construction algorithm (LoPRTC) and the LoPRTC for multi-MDs (LoPRTC-MMD).
3) The performances of the proposed algorithms are analyzed theoretically and evaluated experimentally. Both theoretical analyses and experimental results show that the proposed algorithms can significantly reduce the energy consumption of both MDs and MEC servers.

The rest of this paper is organized as follows. The related works are discussed in Section II. System model and problem definition are presented in Section III. Approximation algorithms and performance analyses for MEC-DMEC system with single MDs and multiple MDs are given in Sections IV and V, respectively. The experimental results are illustrated in Section VI. Finally, the conclusion of this paper is provided in Section VII.

II. Related Work

The task scheduling is an important problem in distributed network systems, such as sensor networks [25] and wireless networks [26], [27]. However, since the topology in MEC systems and the computational capability of MDs are different from these networks, the task scheduling problem in MEC systems is unique. Many previous work have investigated the offloading mechanism in MEC systems with two optimization objectives, latency and energy consumption. The related work can be divided into three categories.

The first category of previous work investigated the single MD and single MEC server computation offloading problem [11], [12]. Wang *et al.* [11] investigated partial computation offloading by jointly optimizing the computational speed, offloading ratio and transmit power of the MD with two objectives, energy consumption of MD minimization, and latency of application execution minimization. Lorenzo *et al.* [12] studied the optimal partition of a complexed task jointly with the selection of transmit power and constellation size to minimize the energy consumption of the MD under the latency constraint of the task.

The second category of previous work studied the computation offloading problem in case of multiple MDs sharing a single MEC server in MEC systems [13]–[20]. You *et al.* [13] studied a consumed energy minimization problem in a multi-MDs MEC system based on time-division multiple access and orthogonal frequency-division multiple access to minimize the weighted sum mobile energy consumption under the constraint on computation latency. The energy-latency tradeoff was investigated in [14], where computation tasks arrived at multiple MDs in a stochastic manner. An iterative algorithm based on a novel successive convex approximation technique was proposed in [15] to minimize the energy consumption of all MDs while meeting latency constraints. The distributed computation offloading decision making problem among multiple MDs was formulated as a multi-MDs computation offloading game in [16]. The authors proposed a distributed computation offloading algorithm to achieve a Nash equilibrium. Molina *et al.* [17] studied the server scheduling problem using queuing theory to achieve a low latency in the execution of applications. Yu *et al.* [18] also intended to reduce the energy consumption of all MDs. Except for competitive relation, multiple MDs can also cooperate with each other. The cooperative computing was proposed in [19] to improve the computation capability of the MEC system. Furthermore, a joint computation-and-communication cooperation protocol was proposed in [20], where the helper (another MD) of an MD can both compute partial offloaded tasks and relay the offloaded tasks to MEC server.

The third category of the previous work considered the computation offloading problem of multiple heterogeneous MEC servers in MEC systems [21]–[24]. An computation framework of offloading from a single MD to multiple MEC servers was proposed in [21]. They proposed algorithms for offloading decisions and CPU frequency scaling to minimize both latency of all tasks and energy consumption of the MD. The scenario of multiple MDs and multiple MEC servers is

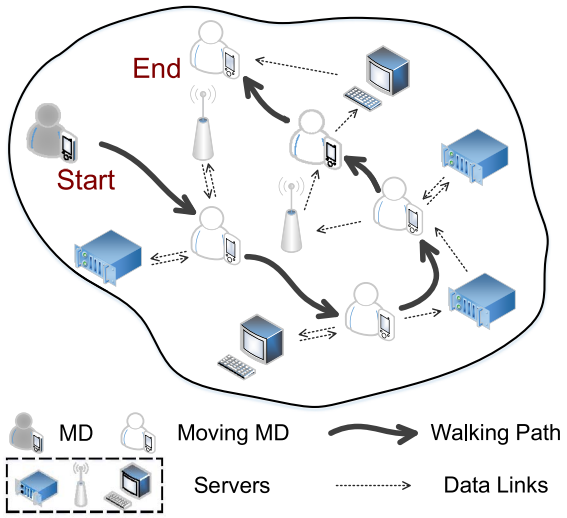

Fig. 1. System model.

TABLE I
SYMBOL TABLE

Symbol	Description
$\mathcal{A}$	$\mathcal{A} = \{A_1, ..., A_K\}$ is the area MEC servers deployed in.
A_j	$A_j (1 \le j \le K)$ is a sub-block of $\mathcal{A}$ divided by MEC servers.
$\mathcal{S}$	$\mathcal{S} = \{S_1, ..., S_M\}$ is the set of MEC servers.
$\mathbb{S}_j$	$\mathbb{S}_j$ is the coverage server set of sub-block A_j
$\mathcal{S}_0$	$\mathcal{S}_0 = \{S_{M+1}, \cdots, S_{M+L}\}$ is the set of MDs.
t_s, t_e	The start and end time of the walking of all MDs.
$\mathcal{P}_l$	$\mathcal{P}_l = \{(A^l_{u_1}, t^l_{u_1}), ..., (A^l_{u_p}, t^l_{u_p})\}$ is the walking path of S_{M+l}.
P^t_i, P^r_i	The transmission and receiving power of MEC server S_i.
p^t_l, p^r_l	The transmission and receiving power of MD S_{M+l}.
R_i	The maximum communication range of MEC server S_i.
L_{ik}	The k-th level of MEC server S_i.
r^U_{ik}, r^D_{ik}	The up-link and down-link rates in level L_{ik}.
f_i	The CPU frequency of MEC server S_i.
f_{M+l}	The CPU frequency of MD S_{M+l}.
$\mathcal{T}_l$	$\mathcal{T}_l = \{T_{l1,\cdots,T_{lN_l}}\}$ is the set of tasks in MD S_{M+l}.
$\mathcal{T}$	$\mathcal{T} = \{T_1, ..., T_N\}$ is the set of tasks in all MDs.
T_j	$T_j = \{\alpha_j, \beta_j, \omega_j, d_j\}$ is the j-th task.
α_j, β_j	The size of input and output of task T_j.
ω_j	Required CPU cycles for the computation workload of T_j.
d_j	The deadline of task T_j.

considered in [22]. MDs and MEC servers are clustered by spatial proximity and mutual interests in popular tasks. The computation latency of tasks was reduced through proactively caching popular and cacheable computing tasks in cooperative MEC servers. Computation migration over multiple MEC servers was investigated in [23] and [24]. That is, when an MD moves closer to a new MEC server, it can migrate the computation of a task that has been offloaded to another MEC server to this server or compute the task in the original MEC server. This process was modeled as a Markov decision process (MDP) in [23] and an optimal threshold policy was proposed to solve for the optimal action of the MDP. Furthermore, the authors extended this paper in [24]. Lyapunov optimization techniques are applied to minimize the migration cost.

However, above works do not consider the mobility of the MDs, which is an important property in the MEC system. In this paper, we focus on the complexed scenario where multiple MDs sharing multiple heterogeneous MEC servers in the MEC systems. Besides, we take into account the mobility of MDs in the task scheduling, which was ignored in most of the previous work. Since MEC servers are also resource-constrained compared with cloud infrastructures, the energy consumption of both MDs and MEC servers are considered in this paper while the previous work ignored the energy consumption of MEC servers.

III. Problem Description

A. System Model

As illustrated in Fig. 1, M heterogeneous MEC servers, $\mathcal{S} = \{S_1, S_2, \ldots, S_M\}$, have been deployed in an area $\mathcal{A}$, and each MEC server S_i has a CPU frequency f_i. There are L MDs walking through area $\mathcal{A}$ in time interval $[t_s, t_e]$, which are denoted as $\mathcal{S}_0 = \{S_{M+1}, \ldots, S_{M+L}\}$. The CPU frequency of MD S_{M+l} is f_{M+l}, where $1 \le l \le L$. Each MD S_{M+1} contains N_l computation tasks, i.e., $\mathcal{T}_l = \{T_{l1}, \ldots, T_{lN_l}\}$. The set of all tasks in all MDs are denoted as $\mathcal{T} = \{T_1, T_2, \ldots, T_N\}$, where $\mathcal{T} = \bigcup_{l=1}^{L} \mathcal{T}_l$ and $N = \sum_{l=1}^{L} N_l$.

Each task T_j can be represented by a quadruple, $T_j = \{\alpha_j, \beta_j, \omega_j, d_j\}$, where α_j is the input data size of task T_j (in bits), β_j is the output data size (in bits), ω_j is the number of CPU cycles that is required to process task T_j, and d_j is the deadline of task T_j, respectively. Without loss of generality, we suppose that $t_s \le d_{l1} \le \cdots \le d_{lN_l} \le t_e$ for tasks in MD S_{M+l} and $t_s \le d_1 \le \cdots \le d_N \le t_e$ for tasks in all MDs. We assume that each task is highly integrated or relatively simple which is indivisible or atomic. Therefore, each MD has to decide whether to process a task as a whole locally or offload it to an MEC server. Since all tasks are generated locally by the MDs, we can further assume that all tasks in $\mathcal{T}_l = \{T_{l1}, \ldots, T_{lN_l}\}$ can be processed locally under all deadline constraints. Besides, all tasks are independent of each other. The symbols used in this section is illustrated in Table I.

In the offloading process, the task profile has to be transmitted to the MEC server, and the MEC server will transmit the outputs back to the MD after finishing the computation of the task. Suppose that the transmitting and receiving powers of MEC server S_i are P^t_i and P^r_i, and let p^t_l and p^r_l be the transmitting and receiving powers of MD S_{M+l}. Thus for each MEC server S_i, there is a communication range R_i, which means if the distance between an MD and S_i is less than R_i, they can communicate with each other freely. That is, the area of the circle centered at S_i with radius R_i is covered by MEC server S_i. Since MDs are walking around in $\mathcal{A}$, the increasing distance between an MD and an MEC server will slowly decrease the uplink and downlink rates.

Therefore, as illustrated in Fig. 2, using $l_i - 1$ circles centered at S_i and radii $R_{i1}, R_{i2}, \ldots, R_{i(l_i-1)}$, we partition the original circle into l_i annuli, which are presented as l_i levels $L_{i1}, L_{i2}, \ldots, L_{il_i}$. We assume that, in each level L_{ik} $(1 \le k \le l_i)$, the uplink rate r^U_{ik} and downlink rate r^D_{ik} are constants. For each pair of $1 \le k < k' \le l_i$, $r^U_{ik} > r^U_{ik'}$, $r^D_{ik} > r^D_{ik'}$, which implies that the inner level always has higher rates. Thus, area $\mathcal{A}$ can be divided into K sub-blocks by different levels of MEC servers, and $\mathcal{A}$ can be represented by $\mathcal{A} = \{A_1, A_2, \ldots, A_K\}$. Each sub-block belongs to a level of

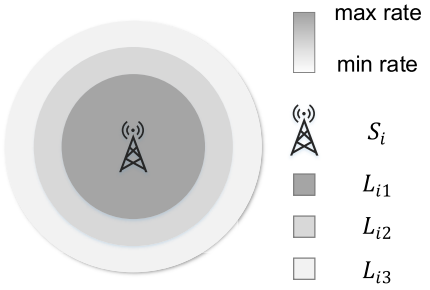

Fig. 2. Communication radius of the MEC server is divided into three levels.

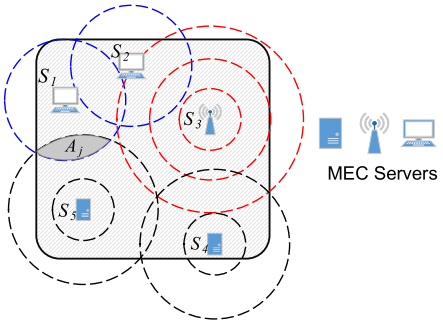

Fig. 3. Area $\mathcal{A}$ can be divided into sub-blocks by multiple levels of MEC servers.

different MEC servers, and a sub-block A_j can also be represented by a set of server-level pairs, $A_j = \{(S_i, L_{ik}) | S_i \in \mathcal{S} \wedge \forall a \in A_j, R_{i(k-1)} \leq \text{distance}(a, S_i) \leq R_{ik}\}$, where a is a spot in A_j. We define the coverage server set of sub-block A_j as $\mathbb{S}_j$, where $\mathbb{S}_j = \{S_i | (S_i, L_{ik}) \in A_j\}$.

An example is shown in Fig. 3, sub-block A_j can be expressed as $\{(S_1, L_{11}), (S_5, L_{52})\}$ and the coverage server set of A_j is $\mathbb{S}_j = \{S_1, S_5\}$. Based on the partition of area $\mathcal{A}$, the MD's walking path $\mathcal{P}$ can be described by a list of sub-block and timestamp pairs, i.e., $\mathcal{P} = \{(A_{u_1}, t_{u_1}), (A_{u_2}, t_{u_2}), \ldots, (A_{u_p}, t_{u_p})\}$. If the MD is in A_{u_j}, then for each $(S_i, L_{ik}) \in A_{u_j}$, the MD can communicate with S_i with uplink and downlink rates r_{ik}^U and r_{ik}^D, respectively.

B. Energy Consumption and Latency Model

1) Local Computing Cost: The power consumption of a CPU is modeled as $P = \kappa f^3$ [28], where f denotes the CPU frequency and κ is a coefficient depending on chip architecture, respectively. The execution latency $D_{j,M+l}$ for MD S_{M+l} to process task $T_j = \{\alpha_j, \beta_j, \omega_j, d_j\} \in \mathcal{T}_l$ locally is

$$D_{j,M+l} = \frac{\omega_j}{f_{M+l}} \tag{1}$$

where f_{M+l} is the CPU frequency of MD S_{M+l}. The energy consumption $E_{j,M+l}$ for MD S_{M+l} to process task T_j is estimated as follows:

$$E_{j,M+l} = D_{j,M+l} \times P = \frac{\omega_j}{f_{M+l}} \times \kappa f_{M+l}^3 = \omega_j \kappa f_{M+l}^2. \tag{2}$$

The energy consumption of any MEC server S_k when task T_j is processed locally by MD S_{M+l} is zero, i.e., $E_{j,k} = 0$ $(1 \leq k \leq M)$. Besides, the energy consumption of another MD S_{M+k} when task T_j is processed locally by MD S_{M+l} is zero, i.e., $E_{j,M+k} = 0$ $(1 \leq k \leq L$ and $k \neq l)$.

2) Offloading Cost: When task $T_j = \{\alpha_j, \beta_j, \omega_j, d_j\} \in \mathcal{T}_l$ is offloaded to MEC server S_i by MD S_{M+l}, let $r_i^U \in \{r_{ik}^U | 1 \leq k \leq l_i\}$ and $r_i^D \in \{r_{ik}^D | 1 \leq k \leq l_i\}$ be the uplink and downlink rates, respectively. Thus, the latency of processing T_j is

$$D_{j,i} = D_{j,i}^U + D_{j,i}^P + D_{j,i}^D \tag{3}$$

where $D_{j,i}^U = (\alpha_j / r_i^U)$ is the uplink transmission latency, $D_{j,i}^D = (\beta_j / r_i^D)$ is the downlink transmission latency, and $D_{j,i}^P = (\omega_j / f_i)$ is the processing latency of MEC server S_i.

The energy consumption $E_{j,i}$ of MEC server S_i when task T_j is offloaded to S_i consists of three parts: 1) receiving energy $E_{j,i}^R$; 2) processing energy $E_{j,i}^P$; and 3) transmitting energy $E_{j,i}^T$. The receiving and transmitting energy consumption are estimated as follows:

$$E_{j,i}^R = P_i^r \times D_{j,i}^U = P_i^r \times \frac{\alpha_j}{r_i^U} \tag{4}$$

$$E_{j,i}^T = P_i^t \times D_{j,i}^D = P_i^t \times \frac{\beta_j}{r_i^D}. \tag{5}$$

The processing energy consumption is

$$E_{j,i}^P = \omega_j \kappa f_i^2. \tag{6}$$

Therefore, the energy consumption $E_{j,i}$ of MEC server S_i is estimated as the following formula:

$$\begin{aligned} E_{j,i} &= E_{j,i}^R + E_{j,i}^P + E_{j,i}^T \\ &= P_i^r \times \frac{\alpha_j}{r_i^U} + \omega_j \kappa f_i^2 + P_i^t \times \frac{\beta_j}{r_i^D}. \end{aligned} \tag{7}$$

The energy consumption $E_{j,M+l}$ of MD S_{M+l} when task T_j is offloaded to MEC server S_i by MD S_{M+l} consists of two parts: 1) uplink energy $E_{j,M+l}^U$ and 2) downlink energy $E_{j,M+l}^D$, which is estimated as follows:

$$\begin{aligned} E_{j,M+l} &= E_{j,M+l}^U + E_{j,M+l}^D \\ &= p_l^t \times \frac{\alpha_j}{r_i^U} + p_l^r \times \frac{\beta_j}{r_i^D}. \end{aligned} \tag{8}$$

The energy consumption $E_{j,k}$ of another MEC server S_k when task T_j is offloaded to MEC server S_i by MD S_{M+l} is zero, i.e., $E_{j,k} = 0$ $(1 \leq k \leq M$ and $k \neq i)$. Besides, the energy consumption $E_{j,M+k}$ of another MD S_{M+k} when task T_j is offloaded to MEC server S_i by MD S_{M+l} is zero, i.e., $E_{j,M+k} = 0$ $(1 \leq k \leq L$ and $k \neq l)$.

C. Problem Definition

For each task, its execution process can be defined as follows.

Definition 1 (Task Execution Process): A task execute process of task T_j in MD S_{M+l} can be represented as a quadruple $\Gamma_j = (T_j, \Delta_j, t_j^S, t_j^R)$, and it satisfies the following conditions.

1) $\Delta_j \in \mathcal{S}_0 \cup \mathcal{S}$ indicates whether task T_j is offloaded to sever S_i $(\Delta_j = S_i)$ or processed locally $(\Delta_j = S_{M+l})$.

2) If $\Delta_j = S_i$, S_{M+l} starts to offloaded task T_j to MEC server S_i at time t_j^S and S_i transmits the result of task T_j at time $t_j^R \geq t_j^S + D_{j,i}^U + D_{j,i}^P$.
3) If $\Delta_j = S_{M+l}$, the MD start to process task T_j locally at time t_j^S and the processing is finished at time $t_j^R \geq t_j^S + D_{j,M+l}$.

Considering the communication radii of MEC servers and interferences between MEC servers and MDs, a *task execution process* maybe infeasible. Therefore, a reasonable offloading set is defined as follows.

Definition 2 (Reachability of MEC Server): MEC server S_i is reachable by task T_j in MD S_{M+l} at time t if S_{M+l} is in the communication range of S_i. That is, according to the definition of the walking path $\mathcal{P}_l = \{(A_{u_1}^l, t_{u_1}^l), \ldots, (A_{u_p}^l, t_{u_p}^l)\}$ of S_{M+l}, if $t_{u_k}^l \leq t < t_{u_{(k+1)}}^l$ $(1 \leq k < p)$, i.e., S_{M+l} is in sub-block $A_{u_k}^l$, there exists $(S_i, L_{il}) \in A_{u_k}^l$, where $1 \leq l \leq l_i$.

Definition 3: (Conflict-Free Execution Process): Two execution processes $\Gamma_j = (T_j, \Delta_j, t_j^S, t_j^R)$ and $\Gamma_{j'} = (T_{j'}, \Delta_{j'}, t_{j'}^S, t_{j'}^R)$ where $j \neq j'$ are conflict-free if they satisfy one of the following cases.

1) $\Delta_j \in \mathcal{S}_0$, $\Delta_{j'} \in \mathcal{S}$ or $\Delta_{j'} \in \mathcal{S}_0$, $\Delta_j \in \mathcal{S}$ which means one of the task in $\{T_j, T_{j'}\}$ is processed locally, and the other one is computed by a reachable MEC server.
2) $\Delta_j = \Delta_{j'} \in \mathcal{S}_0$ and $[t_j^S, t_j^R] \cap [t_{j'}^S, t_{j'}^R] = \emptyset$, which means these two tasks are both processed locally in the same MD but their processing periods have no overlaps.
3) $\Delta_j \in \mathcal{S}_0$, $\Delta_{j'} \in \mathcal{S}_0$ but $\Delta_j \neq \Delta_{j'}$ which means these two tasks are processed locally in different MDs.
4) $\Delta_j = S_i \in \mathcal{S}$ and $\Delta_{j'} = S_{i'} \in \mathcal{S}$ which means these two tasks are offloaded to two reachable MEC servers, and the following two conditions should be held.
   a) The uplink transmissions are conflict-free, i.e., $t_j^S \geq t_{j'}^S + D_{j',i'}^U$ or $t_{j'}^S \geq t_j^S + D_{j,i}^U$.
   b) The downlink transmissions are conflict-free, i.e., $t_j^R \geq t_{j'}^R + D_{j',i'}^D$ or $t_{j'}^R \geq t_j^R + D_{j,i}^D$.

Definition 4 (Reasonable Task Schedule): A reasonable task schedule Π is denoted as $\Pi = \{(T_j, \Delta_j, t_j^S, t_j^R) | 1 \leq j \leq N, t_s \leq t_j^S < t_e, t_s \leq t_j^R \leq t_e\}$. $(T_j, \Delta_j, t_j^S, t_j^R) \in \Pi$ satisfies the following three conditions.

1) Δ_j is reachable by task T_j in MD S_{M+l} in time interval $[t_j^S, t_j^S + D_{j,i}^U]$ and $[t_j^R, t_j^R + D_{j,i}^D]$, where $T_j \in \mathcal{T}_l$.
2) T_j is timely finished at time $t_j^S + D_{j,i}^U + D_{j,i}^P + D_{j,i}^D = t_j^R + D_{j,i}^D$, i.e., $t_j^R + D_{j,i}^D \leq d_j$ if $\Delta_j = S_i \in \mathcal{S}$, or at time $t_j^S + D_{j,M+l} = t_j^R$, i.e., $t_j^R \leq d_j$ if $\Delta_j = S_{M+l} \in \mathcal{S}_0$.
3) $(T_j, \Delta_j, t_j^S, t_j^R)$ is conflict-free with other elements in Π.

Condition 1) implies that the connectivity between Δ_j and MD S_{M+1} is guaranteed during their communication time $[t_j^S, t_j^S + D_{j,i}^U]$ and $[t_j^R, t_j^R + D_{j,i}^D]$. In fact, the velocity of an MD's walking is relatively slow so that the distance between the MD and an MEC server shows little change which will not affect the transmission rate between them much. Therefore, we assume that the uplink rate r_i^U and downlink rate r_i^D maintain their values during one communication.

Obviously, all tasks in $\mathcal{T}$ can be executed by execution processes in each reasonable task schedule. In order to reduce the energy consumption in the MEC system, we propose algorithms for the MEC-DMEC to minimize the energy consumption of both MDs and MEC servers with deadline constraint of each task. On account of the diversity in energy resources of the MD and MEC servers, the energy consumption of each is assigned a scalar weight λ_i $(1 \leq i \leq M + l)$, which is set according to reality. The inputs and outputs of the MEC-DMEC problem is given as follows.

Inputs:

1) The set of MEC servers, $\mathcal{S} = \{S_1, \ldots, S_M\}$, which are deployed in area $\mathcal{A}$.
2) The set of MDs, $\mathcal{S}_0 = \{S_{M+1}, \ldots, S_{M+L}\}$.
3) The set of tasks of all MDs, $\mathcal{T} = \{T_1, \ldots, T_N\}$, where $\mathcal{T} = \bigcup_{l=1}^{L} \mathcal{T}_l$.
4) The walking path of each MD S_{M+l}, $\mathcal{P}_l = \{(A_{u_1}^l, t_{u_1}^l), \ldots, (A_{u_p}^l, t_{u_p}^l)\}$, where $1 \leq l \leq L$.
5) The communication range R_i of each MEC server S_i $(1 \leq i \leq M)$.
6) The CPU frequency of each MEC server f_i $(1 \leq i \leq M)$ and the MD f_{M+l} $(1 \leq l \leq L)$.
7) The transmitting and receiving powers of each MEC server, P_i^t, P_i^r, $(1 \leq i \leq M)$.
8) The transmitting and receiving powers of each MD S_{M+l}, p_l^t, p_l^r, $(1 \leq l \leq L)$.
9) The scalar wight λ_i $(1 \leq i \leq M + L)$ for MEC servers and the MDs.

Outputs: A reasonable task schedule Π satisfies the following formula:

$$
\begin{aligned}
\min \quad & \max_{1 \leq i \leq M+L} \lambda_i \sum_{j=1}^{N} E_{j,i} \\
\text{s.t.} \quad & \Pi \text{ is a reasonable execution schedule} \\
& E_{j,i} = E_{j,i}^R + E_{j,i}^P + E_{j,i}^T, \quad (T_j, S_i, t_j^S, t_j^R) \in \Pi \\
& E_{j,i} = 0, \quad (T_j, S_k, t_j^S, t_j^R) \in \Pi \wedge i \neq k \\
& E_{j,M+l} = \omega_j \kappa f_{M+l}^2, \quad (T_j, S_{M+l}, t_j^S, t_j^R) \in \Pi \\
& E_{j,M+l} = E_{j,M+l}^U + E_{j,M+l}^D, \ (T_j, S_i, t_j^S, t_j^R) \in \Pi \wedge T_j \in \mathcal{T}_l \\
& 0 \leq \lambda_i \leq 1, \quad \forall 1 \leq i \leq M + L. \qquad (9)
\end{aligned}
$$

Theorem 1: The MEC-DMEC problem is at least NP-Hard.

Proof: Construct the following sub-MEC-DMEC problem.

Inputs:

1) The set of MEC servers, $\mathcal{S} = \{S_1, \ldots, S_M\}$, which are deployed in the same location A_u of area $\mathcal{A}$.
2) The set of MDs, $\mathcal{S}_0 = \{S_{M+1}\}$.
3) The set of tasks belong to MD S_{M+1}, $\mathcal{T}_1 = \{T_1, \ldots, T_N\}$.
4) The walking path of S_{M+1} is $\mathcal{P} = \{(A_u, t_u)\}$.
5) The communication range R_i of each MEC server S_i $(1 \leq i \leq M)$ is infinite.
6) The CPU frequency of each MEC server f_i $(1 \leq i \leq M)$ and the MD f_{M+1} have same valued, i.e., $f_1 = \cdots = f_M = f_{M+1} = f$.
7) The transmission and receiving powers of each MEC servers, $P_i^t = 0, P_i^r = 0$, $(1 \leq i \leq M)$, and MD S_{M+1}, $p_1^t = p_1^r = 0$. The transmission and receiving rates are infinite.

8) The scalar wight $\lambda_i = 1$ $(1 < i \leq M+1)$ for MEC servers and MD.

Outputs: A reasonable task schedule Π satisfies the following formula:

$$\begin{aligned}
\min \quad & \max_{1\leq i\leq M+1} \lambda_i \sum_{j=1}^{N} E_{j,i} \\
\text{s.t.} \quad & \Pi \text{ is a reasonable execution schedule} \\
& E_{j,i} = E^P_{j,i}, \quad (T_j, S_i, t^S_j, t^R_j) \in \Pi \\
& E_{j,i} = 0, \quad (T_j, S_k, t^S_j, t^R_j) \in \Pi \wedge i \neq k \\
& E_{j,M+1} = \omega_j \kappa f^2_{M+1}, \quad (T_j, S_{M+l}, t^S_j, t^R_j) \in \Pi \\
& E_{j,M+1} = 0, \quad (T_j, S_i, t^S_j, t^R_j) \in \Pi \wedge T_j \in \mathcal{T}_l \\
& \lambda_i = 1, \quad \forall 1 \leq i \leq M+L. \qquad (10)
\end{aligned}$$

Based on the inputs of sub-MEC-DMEC for each task T_j, $E_{j,1} = \cdots = E_{j,M+1}$. Let $E_j = E_{j,M+1}$, then (10) can be reformulated as

$$\begin{aligned}
\min \quad & \max_{1\leq i\leq M+1} \sum_{j=1}^{N} x_{i,j} E_j \\
\text{s.t.} \quad & x_{i,j} = \begin{cases} 1 & D_j = S_i \\ 0 & \text{otherwise} \end{cases} \quad 0 \leq i \leq M, 1 \leq j \leq N. \qquad (11)
\end{aligned}$$

In the following, we will reduce the minimum makespan schedule (MMS) to the sub-MEC-DMEC problem.

Given an instance of the MMS problem, $[J, P, m]$, where $J = \{J_0, J_1, \ldots, J_{n-1}\}$ indicates n jobs, $P = \{p_0, p_1, \ldots, p_{n-1}\}$ is set of processing times of each job, and m is the number of identical machines. An instance of sub-MEC-DMEC problem can be reduced as follows.

1) Construct $m-1$ MEC servers, $\mathcal{S} = \{S_1, \ldots, S_m\}$, an MD S_0, and an area $\mathcal{A} = \{A_u\}$.
2) Let the CPU frequency f for every $S_i \in \{S_i | 1 \leq i \leq m+1\}$ be a positive value and satisfy $0 \leq f \leq \min\{p_0, \ldots, p_{n-1}\}$.
3) Generate n tasks for MD S_{m+1}, $\mathcal{T}_1 = \{T_1, \ldots, T_n\}$. For each task $T_j = \{\alpha_j, \beta_j, \omega_j, d_j\}$, α_j, β_j are arbitrary positive numbers, $\omega_j = (p_j/\kappa f^2)$ where $p_j \in P$, κ is a coefficient described in the above section, and d_j is an infinite large number.
4) Let the walking path of S_{m+1} be $\mathcal{P} = \{(A_u, t_u)\}$ where t_u is an arbitrary time spot.

According to above four steps, processing task T_j in any device in $\{S_i | 1 \leq i \leq m+1\}$ will consume p_j energy. Let the optimal solution for MMS problem be $X = \{X_j | 0 \leq j \leq n-1,\ 1 \leq X_j \leq m+1,\ X_j \in Z^+\}$ where X_j indicates the index of machine that job J_j will be processed in. Then, the following formula must be minimized:

$$\begin{aligned}
& \max_{1\leq i\leq M+1} \sum_{j=1}^{N} x_{i,j} p_j \\
& \text{s.t.} \quad x_{i,j} = \begin{cases} 1 & X_j = S_i \\ 0 & \text{otherwise} \end{cases} \quad 1 \leq i \leq M+1, 1 \leq j \leq N. \qquad (12)
\end{aligned}$$

TABLE II
SYMBOL TABLE

Symbol	Description
$D_{j,i}$	The execution latency for S_i to process T_j locally.
$D^U_{j,i}$	The uplink transmission latency.
$D^D_{j,i}$	The downlink transmission latency.
$D^P_{j,i}$	The processing latency.
q	$q \in \mathbb{Z}^+$ is a parameter of the algorithm.
CS_j	Candidate offloading MEC server set for task T_j.

For each task $T_j \in \mathcal{T}$, let $\Delta_j = S_{X_j}$, then it is easy to see that, the reasonable $\Pi = \{(T_j, \Delta_j, j, j+0.1) | 1 \leq j \leq N\}$ is an optimal solution of the generated sub-MEC-DMEC problem.

For the same reason, for an optimal solution of the generated sub-MEC-DMEC problem, Π^{opt}, it can be easily transformed into an optimal schedule of the MMS problem. ■

IV. APPROXIMATION ALGORITHM FOR MEC-DMEC SYSTEM WITH SINGLE MD

In this section, we focus on MEC-DMEC system with single MD S_{M+1} first, which is a special case of multi-MDs. The task set of MD S_{M+1} is $\mathcal{T} = \{T_1, \ldots, T_N\}$. Besides, the walking path of the only MD S_{M+1} is denoted as $\mathcal{P} = \{(A_{u_1}, t_{u_1}), (A_{u_2}, t_{u_2}), \ldots, (A_{u_p}, t_{u_p})\}$.

We propose a reasonable task schedule for all tasks $\mathcal{T} = \{T_1, \ldots, T_N\}$ to approximately solve the MEC-DMEC problem. The LoPRTC is illustrated in Algorithm 2, which has two main phases. First, we construct a reasonable local-based task schedule (RLTS) where each task is processed locally by MD S_{M+1}. Then, we balance the energy consumption among MD and MEC servers through offloading some tasks to MEC servers. The symbols used in this section is illustrated in Table II.

A. Reasonable Local-Based Task Schedule Generation

An RLTS is a reasonable task schedule where all tasks in $\mathcal{T} = \{T_1, \ldots, T_N\}$ are processed by MD S_{M+1} locally. In this section, we compute an RLTS Π_0, where each task is put off until the latest start time of its execution that guarantees the constraints of a reasonable task schedule. The computation of Π_0 consists of the following two steps, and the RLTS algorithm is illustrated in Algorithm 1.

Step 1: Determine the execution order of all tasks in $\mathcal{T}$. All tasks are scheduled in the nondescending order of their deadlines. Since $d_1 \leq d_2 \leq \cdots \leq d_N$, tasks $T_1, T_2, \ldots, T_N$ are executed in sequence.

Step 2: For $N \geq j \geq 1$, compute the execution time $[t^{S'}_j, t^{R'}_j]$ of task T_j. First, the latest finish time of T_N's execution is $t^{R'}_j = d_N$ and the latest start time of T_N's execution is $t^{S'}_j = t^{R'}_j - D_{N,M+1}$, where $D_{N,M+1}$ is the execution latency for MD S_{M+1} to process task T_N locally computed in (1). Then, for each task T_j from $j = N-1$ to 1, the latest finish time of T_j's execution is $t^{R'}_j = \min\{d_j, t^{S'}_{j+1}\}$. That means, T_j has to be finished before its deadline d_j and the start time of next task's execution. The latest start time of T_j's execution is $t^{S'}_j = t^{R'}_j - D_{j,M+1}$.

Algorithm 1: RLTS Algorithm

Input: $\mathcal{T} = \{T_1, \ldots, T_N\}$ where $T_j = \{\alpha_j, \beta_j, \omega_j, d_j\}$, $1 \le j \le N$, the CPU frequency f_{M+1} of S_{M+1}
Output: A reasonable local-based task schedule Π_0

$t_N^{R'} = d_N$;
$t_N^{S'} = t_N^{R'} - D_{N,M+1} = t_N^{R'} - \frac{\omega_N}{f_{M+1}}$;
$\Gamma_N = (T_N, S_{M+1}, t_N^{S'}, t_N^{R'})$;
$\Pi_0 = \{\Gamma_N\}$;
for $j = N-1$ *to 1* **do**
  $t_j^{R'} = \min\{d_j, t_{j+1}^{S'}\}$;
  $t_j^{S'} = t_j^{R'} - D_{j,M+1} = t_j^{R'} - \frac{\omega_j}{f_{M+1}}$;
  $\Gamma_j = (T_j, S_{M+1}, t_j^{S'}, t_j^{R'})$;
  $\Pi_0 = \Pi_0 \cup \{\Gamma_j\}$;
Return Π_0;

Theorem 2: The local-based task schedule Π_0 computed by Algorithm 1 is a reasonable task schedule.

Proof: We need to prove that each execution process $\Gamma_j = (T_j, S_{M+1}, t_j^{S'}, t_j^{R'})$ in Π_0 satisfies the following three conditions of a reasonable task schedule.

1) Apparently, MD S_{M+1} is reachable by all tasks all the time.
2) Each task T_j $(1 \le j \le N)$ is timely finished before deadline d_j since the finish time $t_j^{R'} = \min\{d_j, t_{j+1}^{S'}\} \le d_j$.
3) Tasks are processed in the nondescending order of their deadlines $(d_1 \le \cdots \le d_N)$ one by one. Besides, the finish time of task T_j is earlier than the start time of task T_{j+1}, i.e., $t_j^{R'} = \min\{d_j, t_{j+1}^{S'}\} \le t_{j+1}^{S'}$ for all $1 \le j \le N-1$. Therefore, each two execution processes $\Gamma_j = (T_j, S_{M+1}, t_j^{S'}, t_j^{R'})$ and $\Gamma_{j'} = (T_{j'}, S_{M+1}, t_{j'}^{S'}, t_{j'}^{R'})$ in Π_0 are conflict-free, i.e., $[t_j^{S'}, t_j^{R'}] \cap [t_{j'}^{S'}, t_{j'}^{R'}] = \emptyset$.

Therefore, the local-based task schedule Π_0 computed by Algorithm 1 is a reasonable task schedule. ■

B. Partial Reasonable Task Schedule Optimization

In the RLTS Π_0 computed by Algorithm 1, all energy consumptions are from MD S_{M+1} itself. In this section, some tasks are offloaded to MEC servers, where MEC servers can share the energy consumptions of S_{M+1} to minimize the maximum weighted energy consumption of all MEC servers and the MD, i.e., $\max_{1 \le i \le M+1} \lambda_i \sum_{j=1}^{N} E_{j,i}$.

According to the construction of Π_0, the schematization of time interval $[t_s, t_e]$ is shown in Fig. 4. The shaded time interval $[t_j^{S'}, t_j^{R'}]$ presents the execution latency $D_{j,M+1}$ for MD S_{M+1} to process task T_j, where $1 \le j \le N$.

In this section, MD offloads as many tasks as possible in time interval $[t_s, t_e]$ to reduce the maximum weighted energy consumption of all MEC servers and the MD. For each task T_j $(1 \le j \le N)$, the MD should make a decision, whether it offloads T_j to an MEC server. If so, the MD should choose an MEC server to offload and decide the offloading time. The decision on one task may influences the decisions on other tasks. An approximation algorithm for minimizing the maximum weighted energy consumption of all MEC servers and the MD is proposed in this section. The approximation algorithm for partial reasonable task schedule optimization is illustrated in Algorithm 2 from lines 2 to 19, and it consists of the following three phases.

1) Phase I [Tasks Partition (Lines 2–5)]: Divide N tasks $T_1, T_2, \ldots, T_N$ into $\lceil (N/q) \rceil$ groups, i.e., $\{T_1, \ldots, T_q\}, \ldots, \{T_{\lfloor (N/q) \rfloor q+1}, \ldots, T_N\}$, where $1 \le q \le N$ is an integer.

2) Phase II [Local Task Schedule Optimization (Lines 6–17)]: For $0 \le k \le \lfloor (N/q) \rfloor$, compute the local optimal task schedule $\Pi_k = \{(T_j, \Delta_j, t_j^S, t_j^R) | kq+1 \le j \le (k+1)q\}$ for each group of tasks $\{T_{kq+1}, \ldots, T_{(k+1)q}\}$. The details of this phase are described as follows.

First, for each task schedule Π_k, we denote the execution time interval as $[\tau_k^s, \tau_k^e]$, where τ_k^s and τ_k^e denote the start time and end time of Π_k, i.e.,

$$\tau_k^s = \min_{(T_j, \Delta_j, t_j^S, t_j^R) \in \Pi_k} \{t_j^S\} \tag{13}$$

$$\tau_k^e = \max_{(T_j, S_i, t_j^S, t_j^R) \in \Pi_k} \{ \max_{i=M+1} \{t_j^R\}, \max_{1 \le i \le M} \{t_j^R + D_{j,i}^D\} \}. \tag{14}$$

Second, for each $0 \le k \le \lfloor (N/q) \rfloor$, the task schedule of the kth group of tasks $\{T_{kq+1}, \ldots, T_{(k+1)q}\}$ is constructed as the following steps.

Step 1: Compute the candidate execution time interval $[\tau_k^{s'}, \tau_k^{e'}]$ of the kth group of tasks $\{T_{kq+1}, \ldots, T_{(k+1)q}\}$, where $\tau_k^{s'}$ is the earliest possible start time for the execution of all tasks in this group and $\tau_k^{e'}$ is the latest possible end time to complete the execution of all tasks in this group.

According to the definition of $\tau_k^{s'}$, there are two cases for the assignment of $\tau_k^{s'}$.

1) When $k = 0$, $\tau_k^{s'}$ is initialized as the start time when MD S_{M+1} enters area $\mathcal{A}$, i.e., $\tau_1^{s'} = t_s$.
2) When $1 \le k \le \lfloor (N/q) \rfloor$, $\tau_k^{s'}$ is initialized as the end time of task schedule of the $(k-1)$th group Π_{k-1}, i.e., $\tau_k^{s'} = \tau_{k-1}^e$.

According to the definition of $\tau_k^{e'}$, $\tau_k^{e'}$ is assigned as the start time of the first task in the $(k+1)$th group in RLTS Π_0, i.e., $\tau_k^e = t_{(k+1)q+1}^{S'}$.

Step 2: Compute the candidate offloading MEC server set CS_j for each task T_j in the kth group, where $kq+1 \le j \le (k+1)q$.

Each task T_j in $\{T_{kq+1}, \ldots, T_{(k+1)q}\}$ can be offloaded to MEC servers that cover its walking path in time interval $[\tau_k^{s'}, \min\{d_j, \tau_k^{e'}\}]$. Time interval $[\tau_k^{s'}, \min\{d_j, \tau_k^{e'}\}]$ can be divided into several subtime intervals according to S_{M+1}'s walking path $\mathcal{P} = \{(A_{u_1}, t_{u_1}), \ldots, (A_{u_p}, t_{u_p})\}$. An example is shown in Fig. 5, where time interval $[\tau_k^{s'}, \min\{d_j, \tau_k^{e'}\}]$ is divided into b subtime intervals corresponding to b sub-blocks $\{A_{u_a}, \ldots, A_{u_{a+b}}\}$ in area $\mathcal{A}$, where $t_{u_a} \le \tau_k^{s'} \le \min\{d_j, \tau_k^{e'}\} \le t_{a+b}$, $1 \le a \le a+b \le p$. As Fig. 5 shows, each sub-block has a representative color. For example, *yellow* is the representative color of sub-block A_{u_a}, *blue* is the representative color of sub-block $A_{u(a+1)}$, etc.

Each sub-block A_{u_l} is covered by MEC servers in coverage server set $\mathbb{S}_{u_l}$. That means, tasks can be offloaded to these MEC servers when S_{M+1} is at sub-block A_{u_l}. Therefore, the candidate offloading MEC server set for task T_j is the union

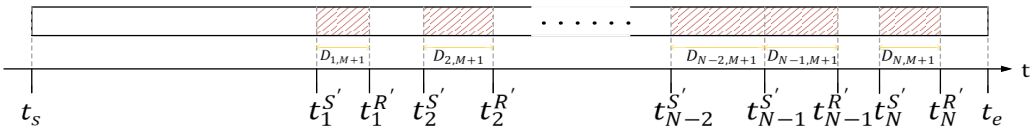

Fig. 4. Time division of the RLTS Π_0.

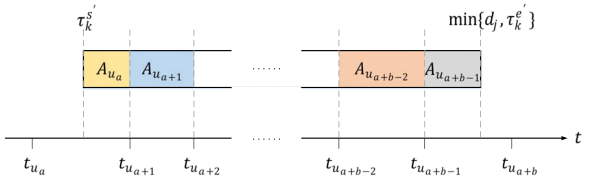

Fig. 5. Time interval $[\tau_k^{s'}, \min\{d_j, \tau_k^{e'}\}]$ is divided into b subtime intervals according to MD's walking path.

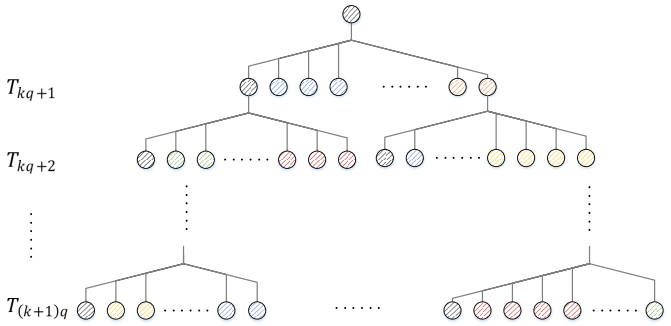

Fig. 6. Search tree for the kth group of tasks.

set of coverage server sets of these b sub-blocks, i.e., $CS_j = \mathbb{S}_{u_a} \cup \cdots \cup \mathbb{S}_{u_{a+b}}$.

Step 3: Compute the locally optimal task schedule for the kth group of tasks $\Pi_k = \{(T_j, \Delta_j, t_j^S, t_j^R) | kq+1 \le j \le (k+1)q\}$ in time interval $[\tau_k^{s'}, \tau_k^{e'}]$.

A feasible offloading process $\Gamma_j = (T_j, S_i, t_j^S, t_j^R)$ $(1 \le i \le M)$ should satisfy the following four conditions.

1) MEC server S_i is reachable for MD S_{M+1} in time interval $[t_j^S, t_j^S + D_{j,i}^U]$ and $[t_j^R, t_j^R + D_{j,i}^D]$, where $D_{j,i}^U$ and $D_{j,i}^D$ is the uplink and downlink latency. That is, the area A_{u_h} $(a \le h \le a+b)$ corresponding to time interval $[t_j^S, t_j^S + D_{j,i}^U]$ or $[t_j^R, t_j^R + D_{j,i}^D]$ is covered by MEC server S_i.
2) MD S_{M+1} does not transmit tasks to any other MEC servers or receive results of tasks from any other MEC servers during time interval $[t_j^S, t_j^S + D_{j,i}^U]$ and $[t_j^R, t_j^R + D_{j,i}^D]$.
3) After receiving task T_j, sever S_i can complete the computation of T_j before t_j^R, i.e., $t_j^S + D_{j,i}^U + D_{j,i}^P \le t_j^R$.
4) The computation time of task T_j does not conflict with the computation time of other tasks.

The computation process of locally optimal task schedule for the kth group of tasks $\{T_{kq+1}, \ldots, T_{(k+1)q}\}$ can be expressed as a search tree with $q+1$ levels. The search tree for the kth group of tasks $\{T_{kq+1}, \ldots, T_{(k+1)q}\}$ is shown in Fig. 6.

The nodes in search tree is divided into $q+1$ levels, where the root node is in level 0. Besides, the candidate schedules of task T_{kq+h} are presented in level l, where $1 \le h \le q$. For task T_{kq+h} $(1 \le h \le q)$, there are $|CS_j| + 1$ candidate schedules, i.e., processing T_{kq+h} locally at S_{M+1} and offloading T_{kq+h} to an MEC server $S_i \in \mathbb{S}_{u_c}$ at area A_{u_c} in time interval $[t_{u_c}, t_{u_{c+1}}]$, where $\mathbb{S}_{u_c} \subseteq CS_j$. The black node in level h denotes that task T_{kq+h} is processed locally at S_{M+1}. A node with other colors in level h denotes that task T_{kq+h} is offloaded to an MEC server at certain sub-block in a certain time interval.

As above, each sub-block has a representative color. Therefore, nodes with different colors denote that the task is offloaded to different MEC servers at different sub-blocks in different time intervals. For example, a blue node in level 1 of the search tree in Fig. 6 denotes that task T_{kq+1} is offloaded to MEC server $S_i \in \mathbb{S}_{u_{a+1}}$ when MD is at sub-block $A_{u_{a+1}}$ in time interval $[t_{u_{a+1}}, t_{u_{a+2}}]$ since *blue* is the representative color of sub-block $A_{u_{a+1}}$.

Since each node except root of the search tree presents a possible schedule for a task, then each path from root to a leaf node presents an task schedule for all tasks in the kth group.

An MD walks through at most p sub-blocks in area $\mathcal{A}$ in time interval $[\tau_k^s, \min\{d_j, \tau_k^e\}]$, and each sub-block is covered by at most M MEC servers. Therefore, a task has at most $pM+1$ candidate schedules. As a consequence, there are at most $(pM+1)^q$ paths of the search tree corresponding to $(pM+1)^q$ candidate task schedules for tasks in the kth group.

However, not all candidate schedules are feasible since they have to satisfy four conditions as mentioned above. That is the actual quantity of paths is much less than $(pM+1)^q$. Besides, the branch-and-bound method [29] is applied to explore branches of the search tree. Before enumerating the candidate schedules of a branch, the maximum energy consumption of all MEC servers in task schedule $\Pi_{k'}$, i.e., $\max_{1 \le i \le M} \lambda_i \sum_{j=kq+1}^{(k+1)q} E_{j,i}$, of the branch is computed, and is discarded if it is larger than the minimum maximum energy consumption of MD and all MEC servers that has found so far by the method.

A locally optimal task schedule Π_k with minimum maximum energy consumption of MD and MEC servers $\max_{1 \le i \le M+1} \lambda_i \sum_{j=kq+1}^{(k+1)q} E_{j,i}$ is computed by the branch-and-bound method. As a consequence, the start time and end time of Π_k is computed as (13) and (14).

3) Phase III [Task Schedule Construction (Line 18)]: Compute the approximate optimal task schedule for all tasks Π by these $\lceil (N/q) \rceil$ locally optimal task schedules, i.e., $\Pi = \bigcup_{k=1}^{\lceil (N/q) \rceil} \Pi_k$.

The detailed computation of approximate optimal task schedule Π for MEC-DMEC system with single MD is described in Algorithm 2.

C. Performance Analysis

1) Computation Complexity: In the first step of RLTS algorithm, the computation complexity is $O(N \log N)$ since

Algorithm 2: LoPRTC Algorithm

Input: MEC server set $\mathcal{S} = \{S_1, \ldots, S_M\}$, walking path $\mathcal{P} = \{(A_{u_1}, t_{u_1}), \ldots, (A_{u_p}, t_{u_p})\}$, coverage server set $\{\mathbb{S}_1, \ldots, \mathbb{S}_K\}$, task set $\mathcal{T} = \{T_1, \ldots, T_N\}$, integer q

Output: The approximation optimal task schedule Π

1 Π_0 =**RLTS**($\mathcal{T}$) ;
2 $\Pi = \emptyset$;
3 **for** $1 \le i \le K$ **do**
4 Select a distinct representative color for sub-block A_i;
5 Divide N tasks $T_1, T_2, \ldots, T_N$ into $\lceil \frac{N}{q} \rceil$ groups, i.e. $\{T_1, \ldots, T_q\}, \{T_{q+1}, \ldots, T_{2q}\}, \ldots, \{T_{\lfloor \frac{N}{q} \rfloor q+1}, \ldots, T_N\}$;
6 **for** $0 \le k \le \lfloor \frac{N}{q} \rfloor$ **do**
7 **if** $k = 0$ **then**
8 $\tau_1^{s'} = t_s$;
9 **else**
10 $\tau_k^{s'} = \tau_{k-1}^{e}$;
11 $\tau_k^{e} = t^{S'}_{(k+1)q+1}$;
12 **for** $kq + 1 \le j \le (k+1)q$ **do**
13 $CS_j = \emptyset$;
14 Divide time interval $[\tau_k^{s'}, \min\{d_j, \tau_k^{e'}\}]$ into b sub-time intervals according to the MD's walking path $\mathcal{P}$;
15 Compute the corresponding b sub-blocks $\{A_{u_a}, \ldots, A_{u_{a+b}}\}$ and their coverage server sets $\{\mathbb{S}_{u_a}, \ldots, \mathbb{S}_{u_{a+b}}\}$;
16 $CS_j = \mathbb{S}_{u_a} \cup \cdots \cup \mathbb{S}_{u_{a+b}}$;
17 Π_k=**Branch-and-bound method**($\{T_{kq+1}, \ldots, T_{(k+1)q}\}$, $\{CS_{kq+1}, \ldots, CS_{(k+1)q}\}$);
18 $\Pi = \Pi \cup \Pi_k$;
19 Return Π;

all tasks are scheduled in a nondescending order of their deadlines. In the second step, the computation complexity is $O(N)$. Thus, the computation complexity of RLTS algorithm is $O(N \log N)$.

From lines 2 to 5 of LoPRTC algorithm, dividing N tasks into $\lceil (N/q) \rceil$ groups needs $O(N)$ times. From lines 6 to 17, the tree searching process costs at most $O((pM+1)^q)$ times. In line 18, the above searching process will be iterated for $\lceil (N/q) \rceil$ times, and thus the computation complexity of PRESO algorithm is $O(\lceil (N/q) \rceil (pM+1)^q)$.

2) Approximate Ratio: Let ALG and OPT be the results of LoPRTC algorithm and the optimal solution, $E(\text{ALG})$ and $E(\text{OPT})$ be energy returned by both algorithms, and $\{T'_1, T'_2, \ldots, T'_N\}$ be the sequence of tasks be executed by OPT. Dividing such sequence into q groups and supposing

$$E_t^{\min} = \min\{p_1^t \times \frac{\alpha_j}{r_{i1}^U} + p_1^r \times \frac{\beta_j}{r_{i1}^D} | T_j \in \mathcal{T} \wedge S_i \in \mathcal{S}\}$$
$$E_c^{\max} = \max\{E_{j,M+1}^P | T_j \in \mathcal{T}\}$$

the approximate ratio of LoPRTC algorithm is analyzed by the following two theorems.

Theorem 3: If $\{T'_1, T'_2, \ldots, T'_q\} = \{T_1, T_2, \ldots, T_q\}$, then we have $[E(\text{ALG})/E(\text{OPT})] = 1 + ([(N-q)\varepsilon]/N)$, where $\varepsilon = (E_c^{\max}/E_t^{\min})$.

Proof: Obviously, since the energy consumed by computation is larger than the energy cost by data transmission, the most idealized schedule of the MD is to offload all tasks to severs. Therefore, we have

$$E(\text{OPT}) \ge NE_t^{\min}. \tag{15}$$

Suppose there is another schedule strategy ALG$'$, in which $\{T_1, \ldots, T_q\}$ have been scheduled by the LoPRTC algorithm, and the other tasks $\{T_{q+1}, \ldots, T_N\}$ have been processed by the MD. Let $E(\text{ALG}')$ be the energy returned by ALG$'$ and $E(\text{ALG}'(q))$ be the energy returned by the first part of ALG$'$, then we have

$$\begin{aligned} E(\text{ALG}) &\le E(\text{ALG}') \\ &\le E(\text{ALG}'(q)) + (N-q)E_c^{\max} \\ &\le E(\text{ALG}(q)) + (N-q)E_c^{\max} \end{aligned} \tag{16}$$

where $E(\text{ALG}(q))$ is the energy returned by LoPRTC algorithm after processing the first q tasks. Based on the property of LoPRTC algorithm, the schedule of the first q tasks is optimized since all reasonable schedules have been searched by the branch-and-bound method. Thus, we have

$$E(\text{ALG}(q)) \le E(\text{OPT}). \tag{17}$$

Combine (16) and (17), we have

$$E(\text{ALG}) \le E(\text{OPT}) + (N-q)E_c^{\max} \tag{18}$$

then

$$\begin{aligned} \frac{E(\text{ALG})}{E(\text{OPT})} &\le 1 + \frac{(N-q)E_c^{\max}}{E(\text{OPT})} \\ &\le 1 + \frac{(N-q)E_c^{\max}}{NE_t^{\min}} \\ &\le 1 + \frac{(N-q)\varepsilon E_t^{\min}}{NE_t^{\min}} \\ &\le 1 + \frac{(N-q)\varepsilon}{N}. \end{aligned} \tag{19}$$

The theorem is proved. ■

To be noticed that, the approximate ratio is related to the value of q. If $q = N$, the approximate ratio is equal to 1, which means the LoPRTC algorithm has constructed the optimal solution during the tree-searching process.

Theorem 4: If $\{T'_1, T'_2, \ldots, T'_q\} \ne \{T_1, T_2, \ldots, T_q\}$, then we have $[E(\text{ALG})/E(\text{OPT})] = \varepsilon$.

Proof: Suppose there is a new strategy ALG$''$, in which all tasks are computed by the MD, then the energy returned by ALG$''$ satisfies that

$$E(\text{ALG}'') \le NE_c^{\max}. \tag{20}$$

Combine (20) and (16), we have

$$\begin{aligned} E(\text{ALG}) &\le E(\text{ALG}'') \\ &\le NE_c^{\max} \\ &\le N\varepsilon E_t^{\min} \\ &\le E(\text{OPT})\varepsilon. \end{aligned} \tag{21}$$

Then we have $[E(\text{ALG})/E(\text{OPT})] = \varepsilon$. ■

V. Approximation Algorithm for MEC-DMEC System With Multi-MDs

In this section, the approximation algorithm for MEC-DMEC system with single MD is extended to multi-MDs. The approximation algorithm for MEC-DMEC system with multi-MDs is to minimize the maximum energy consumption of all MDs and all MEC servers.

The proposed algorithm, named as LoPRTC-MMD, is illustrated in Algorithm 3. It consists of the following two steps.

Step 1 (Lines 1 and 2): Compute the RLTS Π_0^l for tasks $\mathcal{T}_l = \{T_{l1}, \ldots, T_{lN_l}\}$ in each MD S_{M+l} by Algorithm 1 for $1 \le l \le L$. Therefore, each task T_j is assigned a latest local-based execution time interval $[t_j^{S'}, t_j^{R'}]$.

Step 2 (Lines 3–21): Combine all tasks into a task set $\mathcal{T}$ and compute a feasible task schedule Π for all tasks.

This step is similar to the algorithm in Section IV-B. First, all tasks are divided into $\lceil (N/q) \rceil$ groups according to their deadlines, i.e., $\{T_1, \ldots, T_q\}, \ldots, \{T_{\lfloor (N/q) \rfloor q+1}, \ldots, T_N\}$, where $1 \le q \le N$ is an integer. Second, for $0 \le k \le \lfloor (N/q) \rfloor$, compute the locally optimal task schedule $\Pi_k = \{(T_j, \Delta_j, t_j^S, t_j^R) | kq+1 \le j \le (k+1)q\}$ for the kth group of tasks $\{T_{kq+1}, \ldots, T_{(k+1)q}\}$.

However, there is a difference between this step and the method in Section IV-B. In step 3 of Section IV-B, the second condition is that the MD does not transmit tasks to any other MEC servers or receive results of tasks from any other MEC servers during time interval $[t_j^S, t_j^S + D_{j,i}^U]$ and $[t_j^R, t_j^R + D_{j,i}^D]$. In this section, if task T_j belongs to MD S_{M+l} ($1 \le l \le L$), i.e., $T_j \in \mathcal{T}_l$, only MD S_{M+l} does not transmit tasks to any other MEC servers or receive results of tasks from any other MEC servers during time interval $[t_j^S, t_j^S + D_{j,i}^U]$ and $[t_j^R, t_j^R + D_{j,i}^D]$. Other MDs can communicate with other MEC servers during time interval $[t_j^S, t_j^S + D_{j,i}^U]$ and $[t_j^R, t_j^R + D_{j,i}^D]$ without conflicts.

In step 1, tasks are scheduled in a nondescending order in each MD, and thus the computation complexity is $O(LN \log N)$. Based on the analysis in Section IV-C, the computation complexity of step 2 is $O(\lceil (N/q) \rceil (pM+1)^q)$. Therefore, the computation complexity of LoPRTC-MMD is $O(\lceil (N/q) \rceil (pM+1)^q)$.

To be mentioned that, the LoPRTC and LoPRTC-MMD algorithms are centralized algorithms. These two algorithms can be processed at the beginning when MDs entering an area where the MEC system being deployed. If the area is very large, we can partition the area into subareas to decrease the computation complexity. Therefore, although the MEC system is a distributed system, the LoPRTC and LoPRTC-MMD algorithms are suitable.

Algorithm 3: LoPRTC-MMD Algorithm

Input: MEC server set $\mathcal{S} = \{S_1, \ldots, S_M\}$, MD set $\mathcal{S}_0 = \{S_{M+1}, \ldots, S_{M+L}\}$, walking paths $\{\mathcal{P}_l = \{(A_{u_1}^l, t_{u_1}^l), \ldots, (A_{u_{p_l}}^l, t_{u_{p_l}}^l)\} | 1 \le l \le L\}$, coverage server set $\{\mathbb{S}_1, \ldots, \mathbb{S}_K\}$, task set $\mathcal{T} = \mathcal{T}_1 \cup \ldots \cup \mathcal{T}_L$, integer q

Output: The approximation optimal task schedule Π

1 **for** $1 \le l \le L$ **do**
2 $\quad \Pi_0^l$ =**RLTS** ($\mathcal{T}_l$);
3 $\Pi = \emptyset$;
4 **for** $1 \le i \le K$ **do**
5 $\quad$ Select a distinct representative color for sub-block A_i;
6 Sort all tasks in $\mathcal{T}_1 \cup \cdots \cup \mathcal{T}_L$ in the ascending order of their deadlines, i.e. $T_1, \ldots, T_N$;
7 Divide N tasks $T_1, T_2, \ldots, T_N$ into $\lceil \frac{N}{q} \rceil$ groups, i.e. $\{T_1, \ldots, T_q\}, \{T_{q+1}, \ldots, T_{2q}\}, \ldots, \{T_{\lfloor \frac{N}{q} \rfloor q+1}, \ldots, T_N\}$;
8 **for** $0 \le k \le \lfloor \frac{N}{q} \rfloor$ **do**
9 $\quad$ **if** $k = 0$ **then**
10 $\quad\quad \tau_1^{s'} = t_s$;
11 $\quad$ **else**
12 $\quad\quad \tau_k^{s'} = \tau_{k-1}^e$;
13 $\quad \tau_k^e = t_{(k+1)q+1}^{S'}$;
14 $\quad$ **for** $kq+1 \le j \le (k+1)q$ **do**
15 $\quad\quad CS_j = \emptyset$;
16 $\quad\quad$ Find the MD S_{M+l} that contains task T_j, i.e $T_j \in \mathcal{T}_l$;
17 $\quad\quad$ Divide time interval $[\tau_k^{s'}, \min\{d_j, \tau_k^{e'}\}]$ into b sub-time intervals according to the S_{M+l}'s walking path $\mathcal{P}_l$;
18 $\quad\quad$ Compute the corresponding b sub-blocks $\{A_{u_a}, \ldots, A_{u_{a+b}}\}$ and their coverage server sets $\{\mathbb{S}_{u_a}, \ldots, \mathbb{S}_{u_{a+b}}\}$;
19 $\quad\quad CS_j = \mathbb{S}_{u_a} \cup \cdots \cup \mathbb{S}_{u_{a+b}}$;
20 $\quad \Pi_k$=**Branch-and-bound method**($\{T_{kq+1}, \ldots, T_{(k+1)q}\}$, $\{CS_{kq+1}, \ldots, CS_{(k+1)q}\}$);
21 $\quad \Pi = \Pi \cup \Pi_k$;
22 Return Π;

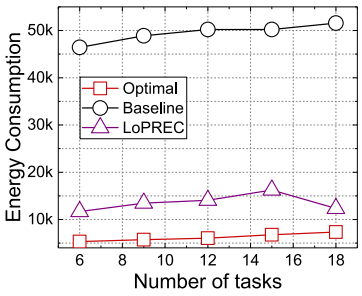

Fig. 7. Impact of the number of tasks.

VI. Experimental Results

A. Setups

A large number of simulations have been carried out to evaluate the performance of LoPRTC and LoPRTC-MMD. In the simulations, the MEC servers have been randomly deployed in a 500 m × 50 m area $\mathcal{A}$, and MDs will move from $(0, 25)$ to $(500, 25)$ straitly. The maximum radius of servers is randomly generated from $[25, 50]$, and the transmission and receiving powers of each server lays in $[0.1, 0.5$ mW$]$. The CPU frequency of MD is 3 GHz, and the CPU frequency of servers varies from 3 to 15 GHz. Different kinds of tasks are randomly generated in these simulations, the input and output size of each task varies from 1000 to 500 kB, and the CPU cycles of computation of each task lays in [3000, 10 000 Megacycles]. Furthermore, the scalar weight λ of each MD equals to 1, and for each server, the scalar weight is in $[0.1, 0.5]$.

We mainly examine the influence of the parameters of the MEC system and LoPRTC algorithm on the minimum maximum consumed energy. The parameters include the number of servers and tasks, the input and output size of each task, the ratio between CPU frequencies of servers and MD, and the

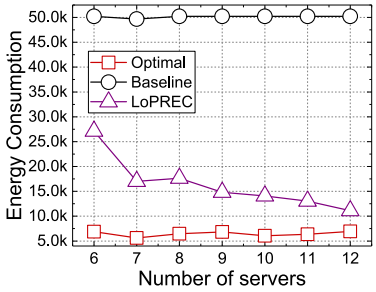

Fig. 8. Impact of the number of servers.

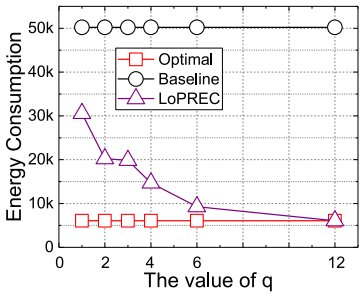

Fig. 9. Impact of the value of q.

algorithm parameter q. Furthermore, we have also investigated the performance of the LoPRTC-MMD algorithm. Although there are many existing works on scheduling MEC tasks, they do not consider the movement of the MDs. The MEC system model of this paper is different from that of others. Thus, it is not suitable enough to compare existing works with our algorithms. In the following simulations, we use an optimal solution which obtained by brutal searching process, and a baseline solution which obtained by processing all tasks locally, to compare with the results of LoPRTC and LoPRTC-MMD algorithms.

B. Impact of the Number of Tasks

We examine the impact of the number of tasks on the maximum minimum energy consumption by adjusting the number of tasks on the MD from 6 to 18. In this group of simulations, the number of servers is 10, and the baseline, optimal, and LoPRTC results are illustrated in Fig. 7. It can be seen that the energy consumption of the LoPRTC algorithm is much better than the baseline, and closes to the optimal result. The energy consumption returned by LoPRTC algorithm is 53.4% larger than the optimal result, which implies the approximate ratio is about 2 in these cases. Furthermore, as the number of tasks grows, the energy consumption of all methods increases.

C. Impact of the Number of Servers

In order to investigate the impact of the number of servers on the energy consumption, the number of tasks is fixed to 12, the number of servers varies from 6 to 12, and all tasks are divided into three groups in LoPRTC algorithm. The LoPRTC algorithm is processed based on above cases, and the returned energy consumption is shown in Fig. 8. It can be seen that as the number of servers increases, the output energy consumption of the LoPRTC algorithm decreases, and is closer to the optimal solution. Since the LoPRTC algorithm is mainly a tree searching process, the increasing of the number of servers will extend the solution space, and makes the results of LoPRTC algorithm better.

D. Impact of the Value of q

Since q is an important parameter of the LoPRTC algorithm, a series of simulations have been carried out to investigate the impact of the value of q on the energy returned by LoPRTC. In these simulations, 12 tasks are assumed to be carried by the MD, ten servers have been deployed in the MEC area, and the value of q is selected as the factors of 12, i.e., $\{1, 2, 3, 4, 6, 12\}$. The results are illustrated in Fig. 9. We can see that, as the value of q growths, the energy consumption returned by the LoPRTC algorithm is closer to the optimal solution. When $q = 1$, the LoPRTC algorithm is a typical greedy algorithm, and is 80% larger than the optimal results. However, as the increasing of q, when $q = 6$, the LoPRTC result is only 34% larger than the optimal energy consumption, and when $q = 12$, the LoPRTC algorithm is optimal. On the other side, the larger value of q may cause higher computation costs. Based on the simulation results, we recommend that selecting q from $[(N/4), (N/2)]$ will obtain an acceptable result with less computation costs, where N is the number of tasks.

E. Impact of the Maximum Size of Inputs and Outputs

In these series of simulations, we investigate the impact of the maximum size of inputs and outputs on the energy consumption. There are ten servers in the area, the MD carries 12 tasks, $q = 4$, and the maximum size of inputs and outputs varies from 500 to 1000 kB. Fig. 10(a) and (b) illustrates the results. It can be easily seen that, the energy consumption does not change obviously when the maximum sizes of input and output are increasing. In the MEC system, if the energy consumed by task transmission is larger than that of computation locally, the task will not be uploaded. Since task computation is the major energy consuming process, the input and output sizes of the task do not influence the final results, which explains the simulation results shown in Fig. 10(a) and (b).

F. Impact of the CPU Frequency Ratio

The CPU frequency ratio is the ratio between CPU frequencies of servers and the MD. In this simulation, there are 12 tasks and ten servers, $q = 4$, and the CPU frequency of the MD is set to be 3 GHz. We assume that all servers share the same CPU frequency, then the impact of the CPU frequency ratio on the energy consumption can be evaluated by varying such CPU frequency from 3 to 15 GHz (one to five times of the CPU frequency of the MD). The results are shown in Fig. 11. It is interesting to see that, as the ratio grows, the energy consumption increases rapidly. According to the LoPRTC algorithm, when the ratio grows from 1 to 5, the consumed energy increases 90%. The simulation results imply that the server with higher CPU frequency may cause

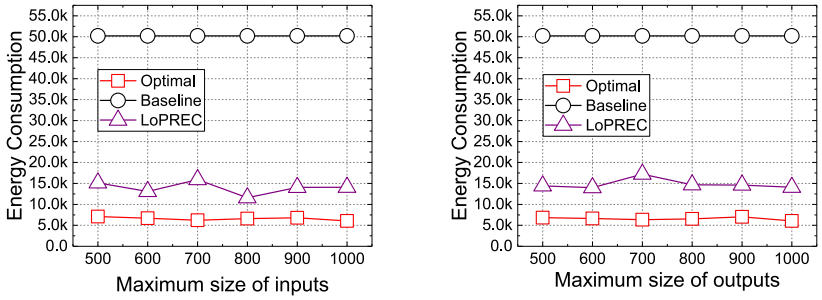

Fig. 10. Impact of the maximum size of inputs and outputs. (a) Uniform assignment. (b) Random assignment.

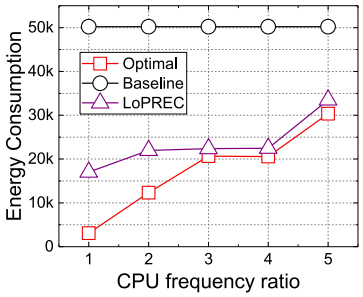

Fig. 11. Compared with centralized algorithms.

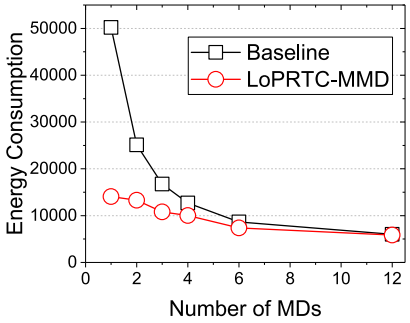

Fig. 12. Impact of the number of MDs.

higher energy consumption, and proper CPU frequencies of servers can lead to a lower energy consumption.

G. Performance of the LoPRTC-MMD Algorithm

In these simulations, we evaluate the performance of the LoPRTC-MMD algorithm by investigating the impact of the number of MDs on the final energy consumption. There are ten servers and 12 tasks in the whole MEC system, and each MD keeps the same number of tasks, then by adjusting the number of MDs from 1 to 12, the influence on the energy consumption is shown in Fig. 12. Due to the high computation complexity, we only use the baseline results to compare with LoPRTC-MMD. It is easily to be noticed that, the increasing of the number MDs can rapidly reduce the energy consumption. Comparing with single MD scenario, when there are 12 MDs in the MEC system, the energy consumption can be reduce for 58%. Furthermore, the LoPRTC-MMD algorithm is much better than the baseline algorithm when the number of MDs is smaller than 6. When the number of MDs increasing, the number of tasks in each MD is small and the difference between the LoPRTC-MMD algorithm and the baseline algorithm is not obvious.

VII. Conclusion

Task scheduling is an important problem in MEC systems. In this paper, we investigated how to scheduling tasks executions in a deadline-aware MEC system. Comparing with existing works, we first consider moving MDs, and a problem named as MEC-DMEC system is proposed to solve the challenges under such scenario. The problem is proved to be at least NP-hard. We have considered the scenarios when there is only one or multi-MDs in the MEC system, and two approximation algorithms are proposed accordingly. Theoretical and experimental results are given in this paper to examine the performance of these algorithms.

References

[1] L. Shi, Y. Wu, L. Liu, X. Sun, and L. Jiang, "Event detection and identification of influential spreaders in social media data streams," *Big Data Min. Anal.*, vol. 1, no. 1, pp. 34–46, Mar. 2018.

[2] T. Y.-H. Chen, L. R. Sivalingam, S. Deng, P. Bahl, and H. Balakrishnan, "Glimpse: Continuous, real-time object recognition on mobile devices," in *Proc. GetMobile*, vol. 20, 2016, pp. 26–29. [Online]. Available: http://doi.acm.org/10.1145/2972413.2972423

[3] J. Cohen, "Embedded speech recognition applications in mobile phones: Status, trends, and challenges," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Las Vegas, NV, USA, Mar./Apr. 2018, pp. 5352–5355. [Online]. Available: https://doi.org/10.1109/ICASSP.2008.4518869

[4] T. R. J. Kumari and H. S. Jayanna, "Speaker verification with the constraint of limited data," *J. Inf. Process. Syst.*, vol. 14, no. 4, pp. 807–823, 2018.

[5] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. B. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Cappadocia, Turkey, Jul. 2012, pp. 59–66. [Online]. Available: https://doi.org/10.1109/ISCC.2012.6249269

[6] K.-M. Koo and E.-Y. Cha, "Image recognition performance enhancements using image normalization," *Human Centric Comput. Inf. Sci.*, vol. 7, no. 1, p. 33, 2017.

[7] J. Liu *et al.*, "Applications of deep learning to MRI images: A survey," *Big Data Min. Anal.*, vol. 1, no. 1, pp. 1–18, Mar. 2018.

[8] Z. Duan, W. Li, and Z. Cai, "Distributed auctions for task assignment and scheduling in mobile crowdsensing systems," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Atlanta, GA, USA, 2017, pp. 635–644.

[9] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *IEEE Comput.*, vol. 43, no. 4, pp. 51–56, Apr. 2010. [Online]. Available: https://doi.org/10.1109/MC.2010.98

[10] R. Wang, J. Yan, D. Wu, H. Wang, and Q. Yang, "Knowledge-centric edge computing based on virtualized D2D communication systems," *IEEE Commun. Mag.*, vol. 56, no. 5, pp. 32–38, May 2018.

[11] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016. [Online]. Available: https://doi.org/10.1109/TCOMM.2016.2599530

[12] P. D. Lorenzo, S. Barbarossa, and S. Sardellitti, "Joint optimization of radio resources and code partitioning in mobile edge computing," *Comput. Sci.*, Jul. 2013.

[13] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017. [Online]. Available: https://doi.org/10.1109/TWC.2016.2633522

[14] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Power-delay tradeoff in multi-user mobile-edge computing systems," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, Washington, DC, USA, Dec. 2016, pp. 1–6. [Online]. Available: https://doi.org/10.1109/GLOCOM.2016.7842160

[15] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 1, no. 2, pp. 89–103, Jun. 2015. [Online]. Available: https://doi.org/10.1109/TSIPN.2015.2448520

[16] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016. [Online]. Available: https://doi.org/10.1109/TNET.2015.2487344

[17] M. Molina, O. Muñoz, A. Pascual-Iserte, and J. Vidal, "Joint scheduling of communication and computation resources in multiuser wireless application offloading," in *Proc. 25th IEEE Annu. Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC)*, Washington, DC, USA, Sep. 2014, pp. 1093–1098. [Online]. Available: https://doi.org/10.1109/PIMRC.2014.7136330

[18] Y. Yu, J. Zhang, and K. B. Letaief, "Joint subcarrier and CPU time allocation for mobile edge computing," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, Washington, DC, USA, Dec. 2016, pp. 1–6. [Online]. Available: https://doi.org/10.1109/GLOCOM.2016.7841937

[19] Z. Sheng, C. Mahapatra, V. C. M. Leung, M. Chen, and P. K. Sahu, "Energy efficient cooperative computing in mobile wireless sensor networks," *IEEE Trans. Cloud Comput.*, vol. 6, no. 1, pp. 114–126, Jan. 2018. [Online]. Available: https://doi.org/10.1109/TCC.2015.2458272

[20] X. Cao, F. Wang, J. Xu, R. Zhang, and S. Cui, "Joint computation and communication cooperation for mobile edge computing," in *Proc. 16th Int. Symp. Model. Optim. Mobile Ad Hoc Wireless Netw. (WiOpt)*, Shanghai, China, May 2018, pp. 1–6. [Online]. Available: https://doi.org/10.23919/WIOPT.2018.8362865

[21] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, Aug. 2017. [Online]. Available: https://doi.org/10.1109/TCOMM.2017.2699660

[22] M. S. ElBamby, M. Bennis, and W. Saad, "Proactive edge computing in latency-constrained fog networks," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, Oulu, Finland, Jun. 2017, pp. 1–6. [Online]. Available: https://doi.org/10.1109/EuCNC.2017.7980678

[23] S. Wang *et al.*, "Mobility-induced service migration in mobile micro-clouds," in *Proc. Mil. Commun. Conf. (MILCOM)*, Oct. 2014, pp. 835–840.

[24] R. Urgaonkar *et al.*, "Dynamic service migration and workload scheduling in edge-clouds," *Perform. Eval.*, vol. 91, pp. 205–228, Sep. 2015. [Online]. Available: https://doi.org/10.1016/j.peva.2015.06.013

[25] Q. Chen, H. Gao, Z. Cai, L. Cheng, and J. Li, "Energy-collision aware data aggregation scheduling for energy harvesting sensor networks," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2018, pp. 117–125.

[26] X. Zheng, Z. Cai, J. Li, and H. Gao, "A study on application-aware scheduling in wireless networks," *IEEE Trans. Mobile Comput.*, vol. 16, no. 7, pp. 1787–1801, Jul. 2017.

[27] X. Zheng, Z. Cai, J. Li, and H. Gao, "An application-aware scheduling policy for real-time traffic," in *Proc. IEEE 35th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2015, pp. 421–430.

[28] W. Zhang *et al.*, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013. [Online]. Available: https://doi.org/10.1109/TWC.2013.072513.121842

[29] A. H. Land and A. G. Doig, "An automatic method for solving discrete programming problems," in *50 Years of Integer Programming 1958–2008—From the Early Years to the State-of-the-Art*. Heidelberg, Germany: Springer, 2010, pp. 105–132. [Online]. Available: https://doi.org/10.1007/978-3-540-68279-0_5

Tongxin Zhu received the B.S. and M.S. degrees in computer science from the Harbin Institute of Technology, Harbin, China, where she is currently pursuing the Ph.D. degree with the School of Computer Science and Technology.

Her current research interests include Internet of Things and wireless sensor networks.

Tuo Shi received the B.S. and M.S. degrees in computer science from the Harbin Institute of Technology, Harbin, China, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Technology.

His current research interests include wireless sensor networks and battery-free sensor networks.

Jianzhong Li received the B.S. degree from Heilongjiang University, Harbin, China.

He is a Professor with the School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China. He was a Visiting Scholar with the University of California at Berkeley, Berkeley, CA, USA, a Staff Scientist with the Information Research Group, Lawrence Berkeley National Laboratory, Berkeley, and a Visiting Professor with the University of Minnesota, Minneapolis, MN, USA. He has published over 150 papers in refereed journals and conferences. His current research interests include data management systems, sensor networks, and data intensive computing.

Mr. Li has served on the editorial boards for distinguished journals, including the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING. He has been involved in the program committees of major computer science and technology conferences, including SIGMOD, VLDB, ICDE, INFOCOM, ICDCS, and WWW.

Zhipeng Cai (SM'15) received the B.S. degree from the Beijing Institute of Technology, Beijing, China, in 2001 and the M.S. and Ph.D. degrees from the Department of Computing Science, University of Alberta, Edmonton, AB, Canada, in 2004 and 2008, respectively.

He is currently an Assistant Professor with the Department of Computer Science, Georgia State University, Atlanta, GA, USA. His current research interests include networking, privacy, and big data.

Dr. Cai was a recipient of the National Science Foundation CAREER Award. He has served as the Program Chair for the International Conference on Wireless Algorithms, Systems, and Applications (WASA) 2014, COCOON 2014, IPCCC 2013, and ISBRA 2013, and the Vice General Chair for IPCCC 2014. He is currently a Steering Committee Co-Chair for WASA. He is an Editor/Guest Editor for *Algorithmica*, *Theoretical Computer Science*, the *Journal of Combinatorial Optimization*, the IEEE/ACM TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS, and the *International Journal of Sensor Networks*.

Xun Zhou received the bachelor's and master's degrees in computer science and technology from the Harbin Institute of Technology, Harbin, China, in 2007 and 2009, respectively, and the Ph.D. degree in computer science from the University of Minnesota, Minneapolis, MN, USA, in 2014, under the supervision of Prof. S. Shekhar.

He is currently a tenure-track Assistant Professor with the Department of Management Sciences, Tippie College of Business, University of Iowa, Iowa City, IA, USA. He is also part of the Iowa Informatics Initiative and the Interdisciplinary Graduate Program in Informatics.