Approximated Assignment Algorithms for Unordered and Ordered Tasks in Data Shared MEC Systems

Journal:	*Transactions on Mobile Computing*
Manuscript ID	TMC-2020-02-0127
Manuscript Type:	Regular
Keywords:	Mobile Edge Computing, Assignment Algorithm, Data Shared, Unordered Tasks, Ordered Tasks

Approximated Assignment Algorithms for Unordered and Ordered Tasks in Data Shared MEC Systems

Siyao Cheng, Jiayan Huang, Zhenyue Chen, Jie Liu *Fellow, IEEE* and Jianzhong, Li

Abstract—The appearance of Mobile Edge Computing (MEC) successfully solves bottlenecks of traditional Cloud based networks. Since mobile edges, *e.g.* base stations, and mobile devices have certain data processing capabilities, it is not necessary to offload all the computation tasks to the remote could for handling. Therefore, it is quite important to decide the optimal task assignment in a MEC system, and a series of algorithms have been proposed to solve it. However, the existing algorithms ignored the data distribution during task assignment, so that the applied ranges of these algorithms are quite limit. Considering the data sharing is quite important in a MEC system, this paper studies task assignment algorithm in Data Shared Mobile Edge Computing Systems in detail. Specifically, two algorithms are proposed to deal with the holistic and divisible unordered tasks respectively. Meanwhile, one heuristic algorithm is given to process the ordered tasks as well since they are also very common in MEC systems. The theoretical analysis on the hardness of the problem, the correctness, complexities and ratio bounds of the proposed algorithms are provided. Finally, the extensive experimental results were carried out. Both of theoretical analysis and experiment results show that all the proposed algorithms have high performance in terms of latency, satisfied rate and energy consumption.

Index Terms—Mobile Edge Computing, Assignment Algorithm, Data Shared, Unordered Tasks, Ordered Tasks.

1 Introduction

With the development of IoT techniques and the proliferation of smart mobile devices, wireless applications become increasingly complicated, such as intelligent manufacture, driverless vehicles and smart health care*etc.* Meanwhile, the QoS(Quality of Service) requirements by applications, including response delay, computation accuracy, congestion control, and energy conservation *etc.*, become much stricter than before. Such requirements cannot be satisfied well in traditional Cloud based networks, where all data are transmitted to the Cloud for processing. Furthermore, the data collected by current IoT devices manifest an explosive growth, so that it is also not practical to transmit all data to the Cloud.

Fortunately, the appearance of the Mobile Edge Computing (MEC) provides a feasible way to solve the above bottlenecks. In a MEC system, the computation capabilities of each mobile device and based station are utilized to decrease the response delay as well as the transmission burden of the network. Similar to previous studies [1] [2], the architecture of a MEC system considered in this paper have three levels as shown in Fig.1. The first level consists of a number of mobile devices, such as mobile phones, smart sensors *etc.*. These devices are mainly responsible for sampling data and interfacing with users. The second level of a MEC system contains a series of base stations, in which a small scale could can be deployed. The base station and the surrounding mobile devices are connected by radio access networks. The third level is usually the remote cloud which owns the richer resources and has higher data processing abilities comparing to base stations and mobile devices in the first two levels. However, the response latency is usually long and the energy cost is very high if all computation task are offloaded to the cloud.

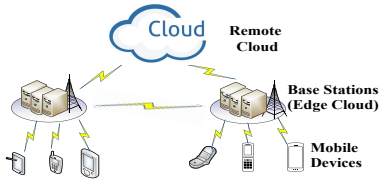


Fig. 1. The three levels of a MEC system.

Since base stations and mobile devices have certain data processing capabilities, it is also not necessary to offload all the tasks to the remote could for handling. Based on such observation, a series of task assignment algorithms have been proposed for MEC systems. The early ones investigate the problem of offloading computation tasks for a single user in MEC systems. To solve it, [3] and [4] proposed Binary offloading algorithms to optimize the latency and energy cost. These works utilized the processing abilities of the edges to improve the performance of the whole network. However, the situation discussed by them seems a little simple. Considering that a MEC system always serves multiple users, [5] formulated the computation offloading problem of multiple users via a single base station as a offloading game, and designed a distributed offloading mechanism that can achieve the Nash equilibrium to solve it. In [6], the authors explored a distributed task offloading algorithm for multiple users in multi-channel wireless networks. The

- *S. Cheng, J. Huang, Z. Cheng, J. Liu and J. Li are with the School of Computer Science and Technology, Harbin Institute of Technology, Harbin, 150001, China.*
  E-mail: csy@hit.edu.cn, jiayanfighting@qq.com, chenzyhit@126.com, jieliu@hit.edu.cn, ljzh@hit.edu.cn
- *S. Cheng is the corresponding author*

Manuscript received February 19, 2020;

above two methods are more practical comparing with the early ones, however, the cooperations among different base stations were not considered. Due to such reason, a new framework that allow a mobile device to offload the tasks to multiple base stations was studied in [7], and semidefinite relaxation-based algorithms were provided to determine the task assignment. However, the situation of serving multiple users has not been addressed sufficiently. More Recently, a series of new task offloading algorithms that serve for multiple users with multiple tasks in a MEC system containing three levels were studied in [8], [9] and [10]. Such algorithms make full use of the MEC systems, however, they rarely take data distribution and task deadline constraints into account during task assignments.

As necessary inputs, the data required by a computation task may be distributed among different mobile devices or even in different clusters. For example, in an intelligent traffic monitoring system, a user wants to know the average flow rate of vehicles in the whole city, while data sampled by his mobile device only show the vehicle flow rate in a small region. In an object tracking system, a mobile device is required to return the whole trajectory of the monitored object, while it only has partial trajectory information. Therefore, data sharing is quite important and common in MEC systems, thus, task assignment problem in such Data Shared Mobile Edge Computing Systems needs to be studied carefully. Furthermore, the limited capability of mobile devices and base stations and the deadline constraints of each task, which were ignored by the existing methods, must be taken into account during task assignment in MEC systems as well.

To solve the above problems, this paper studies task assignment algorithms in Data-Shared MEC systems. Considering the relationships among tasks, we differentiate the tasks into unordered tasks and ordered tasks.

For unordered tasks, they can be further divided into two categories, which are holistic tasks and divisible tasks. We notice that raw data transmission is inevitable if the tasks are holistic since they cannot be processed distributedly, thus, we try to minimize the total energy cost by transmission and computation for holistic tasks during task assignment. On the other hand, we try to avoid raw data transmission by migrating computation and rearranging tasks for divisible tasks since they can be processed in distributed manner. Because the computation operations and partial results are usually much smaller than the raw data, lots of energy can be saved.

For ordered tasks, the input of each task not only depends on its own data, but also relies on results from its predecessor tasks. Therefore, to reduce the energy cost and to schedule the tasks on time, both data distribution and order relationship among tasks should be taken into account. This paper uses a Directed Acyclic Graph to describe the order relationships among tasks, proves that the optimized task assignment problem is NP-hard for ordered tasks, and provides an energy efficient heuristic algorithm based on clustering to solve it approximately.

In summary, the main contributions are as follows.

- We are the first to consider the task assignment in Data-Shared MEC Systems with realistic assumption and constraints.
- For holistic unordered tasks, we prove that it is NP-complete for deciding the optimal task assignment in MEC systems, and give an approximation algorithm based on linear programming is proposed to solve it. The detailed analysis on the correctness, the complexity and the ratio bound, is also given.
- For divisible tasks, we provide two algorithms to meet different optimization goals, and analyze the performance of the algorithms theoretically.
- For ordered tasks, new computation and communication models are provided since the intermediate results should be taken into account. Furthermore, we formalize the Ordered Task Assignment problem into a mixed-integer nonlinear programming problem, which is NP-hard. Therefore, a heuristic algorithm based on clustering is proposed, which tries to decrease the energy consumption as much as possible under the condition that both deadline and resource constraints are met.
- We conduct the extensive experiments and show that all proposed algorithms were efficient in terms of latency, satisfied rate and energy consumption.

The rest of the paper is organized as follows. Section 2 provides system model and problem definitions. Section 3, 4 and Section 5 gives Approximate Assignment Algorithms for Holistic, Divisible and Ordered tasks, respectively. Section 6 shows the experiment results. Section 7 surveys related works. Section 8 concludes the paper.

2 Problem Definition

Without loss of generality, we assume that there exist k base stations,denoted by $B_1, B_2, ..., B_k$, and n users which are $U_1, U_2, ..., U_n$, in a MEC system. Each user U_i $(1 \leq i \leq n)$ has a corresponding mobile device, denoted by i. Each base station B_r $(1 \leq r \leq k)$ connects n_r mobile devices by radio access network, and these mobile devices form a cluster, where $\sum_{r=1}^{k} n_r = n$. Similar to existing work, such as [6], we also consider the quasi-static scenario during task assignments, *i.e.* each mobile device connects to the same base station in the period we considered.

According to the discussion in Section 1, each mobile device collects computation tasks from users. For task $\mathcal{T}_{ij}$ $(1 \leq i \leq n,\ 1 \leq j \leq m)$, which is the j-th computation task raised by user U_i, we use op_{ij} to denote its operations. Generally, a mobile device may not own all the data required on its computation tasks. Therefore, the input data of $\mathcal{T}_{ij}$ can be divided into two disjoint subsets, the local data LD_{ij} and the external data ED_{ij}. Let $\alpha_{ij} = |LD_{ij}|$, $\beta_{ij} = |ED_{ij}|$, and L_{ij} denote the possible cluster (or mobile devices) that may own ED_{ij}. Besides the input data, each computation task also requires some resources (*e.g.* memory) and has the deadline to meet, which are denoted by C_{ij} and T_{ij}, respectively. Thus, a computation task $\mathcal{T}_{ij}$ can be represented as a tuple, that is, $\mathcal{T}_{ij} = (op_{ij}, LD_{ij}, ED_{ij}, L_{ij}, C_{ij}, T_{ij})$. For the convenience of discussion, we assume that every user raises the same number of tasks in a MEC system. All proposed algorithms are also suitable for dealing with the more general situation by minor revisions.

When the tasks are holistic, then all the local and external data should be collected to a single subsystem for processing, that is a computation task $\mathcal{T}_{ij}$ could be carried out by one of three possible subsystems, *i.e.* the mobile device i, the base station connected to i or the remote cloud. We use the indicator variable x_{ijl} to show which subsystem is involved dealing with $\mathcal{T}_{ij}$, and x_{ijl} satisfies that

$$x_{ijl} = \begin{cases} 1, & \text{If } \mathcal{T}_{ij} \text{ is processed by subsystem } l \\ 0, & \text{Otherwise} \end{cases} \quad (1)$$

where Subsystem 1 (*i.e.* $l = 1$) means the mobile device i, Subsystem 2 (*i.e.* $l = 2$) is the base station connected to i and Subsystem 3 (*i.e.* $l = 3$) is regarded as the remote cloud.

Basically, the goal of assigning the computation tasks in a MEC system is to determine proper $\{x_{ijl}|1 \leq i \leq n, 1 \leq j \leq m, l \in \{1,2,3\}\}$ which can minimize the energy cost of the system on conditions that the deadlines of the tasks could be met. There are two main factors, *i.e.* computation and transmission, that affects the runtime and energy cost of each task. Therefore, we discuss the models of computation and transmission in the following two subsections.

2.1 Computation Model

Since the first situation we considered is that the computation tasks are indivisible and holistic, only one subsystem is chosen to carried out $\mathcal{T}_{ij}$, so $x_{ij1} + x_{ij2} + x_{ij3} = 1$. The situation that computation tasks are divisible will be considered in Section 4.

Let f_i represent the CPU frequency of mobile device i, $\lambda_{ij1}(y)$ be the function to define the number of CPU cycles to process $\mathcal{T}_{ij}$ on condition that the input data size is y, where $\lambda_{ij1}(y)$ can be determined by the computation complexity of $\mathcal{T}_{ij}$. Thus, if $x_{ij1} = 1$, *i.e.* task $\mathcal{T}_{ij}$ is processed locally, then the computation time and energy cost for processing $\mathcal{T}_{ij}$ can be estimated by the following formulas,respectively.

$$t_{ij1}^{(C)} = \frac{\lambda_{ij1}(\alpha_{ij} + \beta_{ij})}{f_i}, \; E_{ij1}^{(C)} = \kappa \lambda_{ij1}(\alpha_{ij} + \beta_{ij}) f_i^2 \quad (2)$$

according to [3] and [11], where κ is a constant related to the hardware architecture.

Similarly, if $x_{ij2} = 1$ or $x_{ij3} = 1$, that is, the base station which connects with mobile device i or the remote cloud is selected to deal with task $\mathcal{T}_{ij}$, the computation time $t_{ij2}^{(C)}$ and $t_{ij3}^{(C)}$ can be determined by the following formulas,respectively.

$$t_{ij2}^{(C)} = \frac{\lambda_{ij2}(\alpha_{ij} + \beta_{ij})}{f_s}, \; t_{ij3}^{(C)} = \frac{\lambda_{ij3}(\alpha_{ij} + \beta_{ij})}{f_c} \quad (3)$$

where $\lambda_{ij2}(\alpha_{ij} + \beta_{ij})$ and $\lambda_{ij3}(\alpha_{ij} + \beta_{ij})$ represents the number of CPU cycles of a base station and the cloud for dealing with the data with size $\alpha_{ij} + \beta_{ij}$ size, and f_s and f_c are the CPU frequencies of a base station and the cloud, respectively.

Since the energy cost of base station and the cloud during computation are not primary concern, they can be ignored.

2.2 Transmission Model

In order to obtain all inputs for a task, each mobile device should exchange information with the connected base station by the radio access network.

Let $r_i^{(U)}$ and $r_i^{(D)}$ be the upload and download rate of mobile device i $(1 \leq i \leq n)$. Both of them can be determined by [6] [7] using Shannon's Theory, that is

$$r_i^{(U)} = W_i^{(U)} \log_2(1 + \frac{g_i^{(U)} P_i^{(T)}}{\varpi_0}),$$
$$r_i^{(D)} = W_i^{(D)} \log_2(1 + \frac{g_i^{(D)} P^{(S)}}{\varpi_0})$$

where $g_i^{(U)}$ and $g_i^{(D)}$ are the uplink and downlink channel gains between mobile device i and the base station; $W_i^{(U)}$ and $W_i^{(D)}$ are the uplink and downlink channel bandwidths that the base station allocates for the mobile device i; $P_i^{(T)}$ and $P^{(S)}$ are the transmission powers of mobile device i and the base station; ϖ_0 is the power of the white noise.

Let $e_i^{(T)}(X)$, $e_i^{(R)}(X)$ and $e_{B,B}(X)$ denote energy cost of transmitting data with size X from mobile device i to the base station, from the base station to mobile device i, and between two base stations, respectively. All of them also can be determined by the transmission rate and power according to [6].

Thus, if $x_{ij1} = 1$, then the mobile device i should retrieve the external data with the size β_{ij} from other mobile devices through the MEC system. Based on the definition of the task, L_{ij} will indicate which mobile device may own such external data. Therefore, the data retrieving time $t_{ij1}^{(R)}$ can be determined by the following formula.

$$t_{ij1}^{(R)} = \begin{cases} \frac{\beta_{ij}}{r_{L_{ij}}^{(U)}} + \frac{\beta_{ij}}{r_i^{(D)}}, & \text{If } L_{ij}, i \text{ are in the same cluster} \\ \frac{\beta_{ij}}{r_{L_{ij}}^{(U)}} + \frac{\beta_{ij}}{r_i^{(D)}} + t_{B,B}(\beta_{ij}), & \text{Otherwise} \end{cases}$$

where $t_{B,B}(\beta_{ij})$ is the time spent for transmitting the data with size β_{ij} between two base stations.

The energy cost of retrieving the external data, $E_{ij1}^{(R)}$, satisfies that

$$E_{ij1}^{(R)} = \begin{cases} e_{L_i}^{(T)}(\beta_i) + e_i^{(R)}(\beta_i), & \text{If } L_{ij}, i \text{ are in same cluster} \\ e_{L_i}^{(T)}(\beta_i) + e_i^{(R)}(\beta_i) + e_{B,B}(\beta_i), & \text{Otherwise} \end{cases}$$

Secondly, if $x_{ij2} = 1$, then both of the local data and external data of $\mathcal{T}_{ij}$ should be transmitted to the base station for processing, and the result should be returned to mobile device i. Suppose that $\eta(y)$ represents the size of the computation result under the condition that the input data size equals to y, then the information transmission time, $t_{ij2}^{(R)}$ satisfies the following formula.

$$t_{ij2}^{(R)} = \begin{cases} \max\{\frac{\beta_{ij}}{r_{L_{ij}}^{(U)}}, \frac{\alpha_{ij}}{r_i^{(U)}}\} + \frac{\eta(\alpha_{ij}+\beta_{ij})}{r_i^D}, \\ \qquad \text{If } L_{ij} \text{ and } i \text{ are in the same cluster} \\ \max\{\frac{\beta_{ij}}{r_{L_{ij}}^{(U)}} + t_{B,B}(\beta_j), \frac{\alpha_{ij}}{r_i^{(U)}}\} + \frac{\eta(\alpha_{ij}+\beta_{ij})}{r_i^D}, \text{ Else} \end{cases}$$

The total energy cost during transmission, denoted by $E_{ij2}^{(R)}$, can be calculated by Formula (4).

$$E_{ij2}^{(R)} = \begin{cases} e_{L_i}^{(T)}(\beta_i) + e_i^{(T)}(\alpha_i) + e_i^{(R)}(\eta(\alpha_{ij} + \beta_{ij})), \\ \qquad \text{If } L_{ij}, i \text{ are in same cluster} \\ e_{L_i}^{(T)}(\beta_i) + e_i^{(T)}(\alpha_i) + e_i^{(R)}(\eta(\alpha_{ij} + \beta_{ij})) \\ +e_{B,B}(\beta_i), \qquad \text{Otherwise} \end{cases} \quad (4)$$

Finally, if $x_{ij3} = 1$, that is, the remote cloud is selected to deal with $\mathcal{T}_{ij}$. Therefore, both of local and external data are required to be delivered to the cloud for processing. Let $t_{B,C}(X)$ and $e_{B,C}(X)$ be the transmission delay and energy cost of transmitting data with size X from a base station to the remote cloud. Then, the time and energy spent during transmission, denoted by $t_{ij3}^{(R)}$ and $E_{ij3}^{(R)}$, can be determined by the following two formulas when $x_{ij3} = 1$.

$$t_{ij3}^{(R)} = \max\{\frac{\beta_{ij}}{r_{L_{ij}}^{(U)}}, \frac{\alpha_{ij}}{r_i^{(U)}}\} + \frac{\eta(\alpha_{ij}+\beta_{ij})}{r_i^D} + t_{B,C}(\alpha_{ij}+\beta_{ij}+\eta(\alpha_{ij}+\beta_{ij}))$$

$$E_{ij3}^{(R)} = e_{L_i}^{(T)}(\beta_i) + e_i^{(T)}(\alpha_i) + e_i^{(R)}(\eta(\alpha_{ij}+\beta_{ij})) + e_{B,C}(\alpha_{ij}+\beta_{ij}+\eta(\alpha_{ij}+\beta_{ij}))$$

Based on the above analysis, more data are required to transmit when $x_{ij3} = 1$, and the transmission time from a base station to the remote cloud is much larger than that between two base stations [12] [13], so that more energy will be consumed when $x_{ij3} = 1$, that is $E_{ij3}^{(R)} > E_{ij2}^{(R)}$.

2.3 Problem Definition

Based on the computation and transmission models, the total delay and energy cost for processing task $\mathcal{T}_{ij}$ when $l = 1, 2, 3$ are given as follows.

$$t_{ijl} = t_{ijl}^{(C)} + t_{ijl}^{(R)}, E_{ijl} = \begin{cases} E_{ijl}^{(R)} + E_{ijl}^{(C)} & \text{if } l = 1 \\ E_{ijl}^{(R)} & \text{if } l = 2, 3 \end{cases} \quad (5)$$

Meanwhile, the resources (*e.g.* memory, thread, virtual machine, processor) provided by mobile devices and base stations for computation are limited as discussed in Section 1, so that we use max_i and max_B to denote upper bound of corresponding resource. Thus, $\sum_{j=1}^{m} C_{ij}x_{ij1} \leq max_i$ and $\sum_{i=1}^{n}\sum_{j=1}^{m} C_{ij}x_{ij2} \leq max_B$, which means that the total resource occupied by the tasks assigned to mobile device i and a base station should be no more than max_i and max_B.

Thus, the problem of Holistic Task Assignment (HTA) in a MEC system is defined as follows.

Input:

1) A MEC system with n mobile devices,$\{1, 2, ...n\}$, and k base stations, $\{B_1, B_2, ..., B_k\}$;
2) A set of computation tasks $\{\mathcal{T}_{ij}|1 \leq i \leq n, 1 \leq j \leq m\}$, and the estimated CPU cycles and result size for each computation task.
3) Basic frequencies of mobile devices, base station and the cloud, $\{f_i|1 \leq i \leq n\}$, f_s, f_c;
4) The Network parameters, $r_i^{(U)}$, $r_i^{(D)}$, $t_{B,B}(X)$, $t_{B,C}(X)$ $e_i^{(T)}(X)$, $e_i^{(R)}(X)$, $e_{B,B}(X)$ and $e_{B,C}(X)$;
5) The resource limitations of each mobile device and base station $\{max_i|1 \leq i \leq n\}$, max_B.

Output: The task assignment strategy $\{x_{ijl}|1 \leq i \leq n, 1 \leq j \leq m, l = 1, 2, 3\}$, where the total energy cost of the system, *i.e.* $\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{l=1}^{3} E_{ijl}x_{ijl}$, is minimized, and the following conditions are satisfied.

$$\sum_{l=1}^{3} t_{ijl}x_{ijl} \leq T_{ij} \quad \forall i \in N, \forall j \in M, \quad \text{(C1)}$$

$$\sum_{j=1}^{m} C_{ij}x_{ij1} \leq max_i \quad \forall i \in N, \quad \text{(C2)}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m} C_{ij}x_{ij2} \leq max_S, \quad \text{(C3)}$$

$$\sum_{l=1}^{3} x_{ijl} = 1 \quad \forall i \in N, \forall j \in M, \quad \text{(C4)}$$

$$x_{ijl} \in \{0, 1\} \ \forall i \in N, \forall j \in M, \forall l \in \{1, 2, 3\} \quad \text{(C5)}$$

The hardness and solution of such problem will be analyzed in Section 3.

3 Holistic Task Assignment Algorithm

Before introducing the algorithm, we first analyze the hardness of HTA problem.

Theorem 1. The HTA problem is NP-complete.

Proof Consider a special case of the HTA problem, where $max_i = 0$, $T_{ij} = \infty$ for all $1 \leq i \leq n$ and $1 \leq j \leq m$. That is, the deadline of each task can be ignored and the mobile devices do not have any ability to process tasks.

Therefore, we have that $x_{ij1} = 0$ and $x_{ij3} = 1 - x_{ij2}$ and the HTA problem is equal to the following one

$$P1: \max \sum_{i=1}^{n}\sum_{j=1}^{m}(E_{ij3} - E_{ij2})x_{ij2}$$
$$s.t. \sum_{i=1}^{n}\sum_{j=1}^{m} C_{ij}x_{ij2} \leq max_B$$
$$x_{ij2} \in \{0, 1\} \ \forall i \in N, \forall j \in m$$

Since $E_{ij3} > E_{ij2}$, the problem $P1$ can be regarded as a Knapsack problem with $n \times m$ items, the value and weight of item (i, j) equal to $E_{ij3} - E_{ij2}$ and C_{ij}, and the capacity of Knapsack is max_B. Since Knapsack problem is NP-complete, *i.e.* the special case of the HTA problem is NP-complete, so the HTA problem is at least NP-complete. □

Since HTA problem is NP-complete, the following subsection will proposed an approximation algorithm for it.

3.1 Linear Programming based Algorithm

Based on Section 2, there only exist three possible subsystems to process task $\mathcal{T}_{ij}$, which are mobile device i, the connected base station B_r and the remote cloud. Therefore, each cluster can be considered separately. This section will discuss how to assign the task in a cluster.

Let $(\mathbf{y_1}, \mathbf{y_2}, ..., \mathbf{y_k})$ denote the combination of k vectors. For example, $(\mathbf{y}, \mathbf{z}) = (y_1, y_2, ..., y_r, z_1, z_2, ..., z_l)$ if $\mathbf{y} = (y_1, y_2, ..y_r)$ and $\mathbf{z} = (z_1, z_2., , , z_l)$. We use $\xi_{ij} = (x_{ij1}, x_{ij2}, x_{ij3})$ to represent the assignment of task $\mathcal{T}_{ij}$, $\xi_i = (\xi_{i1}, \xi_{i2}, ..., \xi_{im})$ to represent the assignment result of all the tasks given by user U_i, and vector $\xi = (\xi_1, \xi_2, ..., \xi_{n_r})$ denote the assignment result of all the tasks in the cluster. In the same way, $\mathbf{e}_{ij} = (E_{ij1}, E_{ij2}, E_{ij3})$ denotes the possible energy cost of task $\mathcal{T}_{ij}$, and we set $\mathbf{e}_i = (\mathbf{e}_{i1}, \mathbf{e}_{i2}, \ldots, \mathbf{e}_{im})$, and $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{n_r})$.

Obviously, both of ξ and $\mathbf{e}$ are vectors with length $3n_rm$. Based on ξ, $\mathbf{e}$ and the definition of the HTA problem, the relaxed linear programming problem is formalized as follows.

$$P2: \min_{\xi} \ \mathbf{e}\xi^T$$
$$s.t.\ \mathbf{A}_1\xi^T \le \mathbf{b}_1,\ \ \mathbf{A}_2\xi^T \le \mathbf{b}_2,$$
$$\mathbf{A}_3\xi^T \le \mathbf{b}_3,\ \ \mathbf{A}_4\xi^T = \mathbf{b}_4,$$
$$\xi[i] \in [0,1],\ \ \forall i = 1,2,\ldots,3\times n_r \times m$$

where $\xi[i]$ is the i-th element of vector ξ, $\mathbf{b}_1 = (T_{11}, T_{11}, T_{11}, \ldots, T_{ij}, T_{ij}, T_{ij}, \ldots, T_{n_rm}, T_{n_rm}, T_{n_rm})^T_{1\times 3n_rm}$, $\mathbf{b}_2 = (max_1, max_2 \ldots, max_i, \ldots, max_{n_r})^T_{1\times n_r}$, $\mathbf{b}_3 = [max_B]$, $\mathbf{b}_4 = (1,1,\ldots,1)^T_{n_rm\times 1}$, $\mathbf{A}_1 = diag(t_{111}, t_{112}, t_{113}, ..., t_{n_rm1}, t_{n_rm2}, t_{n_rm3})_{3n_rm\times 3n_rm}$, $\mathbf{A}_3 = [0, C_{i1}, 0, 0, C_{i2}, 0, \ldots, 0, C_{n_rm}, 0]_{1\times 3n_rm}$,

$$\mathbf{A}_2 = \begin{bmatrix} \mathbf{a}_1 & \mathbf{0}_{1\times 3m} & \cdots & \mathbf{0}_{1\times 3m} \\ \mathbf{0}_{1\times 3m} & \mathbf{a}_2 & \cdots & \mathbf{0}_{1\times 3m} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{1\times 3m} & \mathbf{0}_{1\times 3m} & \cdots & \mathbf{a}_{n_r} \end{bmatrix}_{n_r\times 3n_rm},$$

$$\mathbf{A}_4 = \begin{bmatrix} \mathbf{a}_{11} & \mathbf{0}_{1\times 3} & \cdots & \mathbf{0}_{1\times 3} \\ \mathbf{0}_{1\times 3} & \mathbf{a}_{12} & \cdots & \mathbf{0}_{1\times 3} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} & \cdots & \mathbf{a}_{n_rm} \end{bmatrix}_{n_rm\times 3n_rm},$$

and $\mathbf{a}_i = (C_{i1}, 0, 0, \ldots, C_{im}, 0, 0)_{1\times 3m}$,and $\mathbf{a}_{ij} = (1,1,1)$.

Based on above symbols, the Holistic Task Assignment Algorithm for (*LP-HTA* for short) has six steps.

Step 1. Solve the linear programming problem $P2$ to obtain ξ by using the interior points method algorithm [14].

Step 2. Construct an equivalent fractional matrix $\mathbf{X}$ according to ξ. That is, $\mathbf{X}[i,j,l] = \xi[3m\times(i-1)+3\times(j-1)+l]$ for all $1 \le i \le n_r$, $1 \le j \le m$ and $l \in \{1,2,3\}$.

Step 3. If all elements of $\mathbf{X}$ are binary, then set $x_{ijl} = X[i,j,l]$, and return $\{x_{ijl} | 1 \le i \le n_r, 1 \le j \le m, l \in \{1,2,3\}\}$ as the assignment result.

Otherwise, let $q = \arg\max_{l\in\{1,2,3\}} \mathbf{X}[i,j,l]$, *i.e.* $\mathbf{X}[i,j,q] = \max\{\mathbf{X}[i,j,1], \mathbf{X}[i,j,2], \mathbf{X}[i,j,3]\})$, then set

$$\widehat{x}_{ijl} = \begin{cases} 1, & \text{if } l = q \\ 0, & \text{Otherwise} \end{cases} \quad (6)$$

Step 4. Consider each $\widehat{x}_{ijq} = 1$. If $t_{ijq} \le T_{ij}$, set $x_{ijq} = 1$ and $x_{ijp} = 0$ for any $p \in \{1,2,3\}$ and $p \ne q$.

Otherwise, find l which satisfies $l = \max_{p\in\{p|t_{ijp}\le T_{ij}\}} \mathbf{X}[i,j,p]$, then set $x_{ijl} = 1$ and $x_{ijr} = 0$ for any $r \in \{1,2,3\}$ and $r \ne l$. If we cannot find such l, then we reject $\mathcal{T}_{ij}$ and inform users.

Step 5. For each mobile device i, reconsider Constraint $C2$ in HTA problem. Compute $\sum_{j=1}^{m} C_{ij}x_{ij1}$. If $\sum_{j=1}^{m} C_{ij}x_{ij1} > max_i$, the following steps are carried out.

1) Greedily select tasks from $S_1 = \{\mathcal{T}_{ij} | x_{ij1} = 1 \wedge t_{ij2} \le T_{ij}\}$ with the highest C_{ij};
2) Set $x_{ij1} = 0$ and $x_{ij2} = 1$ for any selected task $\mathcal{T}_{ij}$, that is, move the task $\mathcal{T}_{ij}$ to the base station for processing;
3) Repeat the above two steps until $\sum_{j=1}^{m} C_{ij}x_{ij1} \le max_i$ or $\{\mathcal{T}_{ij} | x_{ij1} = 1 \wedge t_{ij2} \le T_{ij}\} = \emptyset$;

If $\sum_{j=1}^{m} C_{ij}x_{ij1}$ is still larger than max_i, greedily reject tasks with high resource occupation and inform users.

Step 6. For the base station, we do the similar operations as Step 5. First, check whether $\sum_{i=1}^{n_r}\sum_{j=1}^{m} C_{ij}x_{ij2} \le max_B$ is satisfied. If it is, then return the assignment result $\{x_{ijl} | 1 \le i \le n_r, 1 \le j \le m, l \in \{1,2,3\}\}$. Otherwise,

1) Calculate $S_2 = \{\mathcal{T}_{ij} | x_{ij2} = 1 \wedge t_{ij3} \le T_{ij}\}$. If $S_2 \ne \emptyset$, greedily select $\mathcal{T}_{ij}$ from S_2 with the largest C_{ij}. Set $x_{ij2} = 0$ and $x_{ij3} = 1$, *i.e.* move $\mathcal{T}_{ij}$ to Cloud.
2) Repeat the above step until $\sum_{i=1}^{n_r}\sum_{j=1}^{m} C_{ij}x_{ij2} \le max_B$ or $\{\mathcal{T}_{ij} | x_{ij2} = 1 \wedge t_{ij3} \le T_{ij}\} = \emptyset$

If $\sum_{i=1}^{n_r}\sum_{j=1}^{m} C_{ij}x_{ij2} \le max_B$ still can not be satisfied, then reject some tasks with the highest resource occupation and inform the users.

3.2 Performance Analysis of the Algorithm

3.2.1 Correctness

Firstly, we need to verifies that *LP-HTA* algorithm provided a feasible solution for the HTA problem.

Suppose that $\{x_{ijl} | 1 \le i \le n_r, 1 \le j \le m, l \in \{1,2,3\}\}$ be the assignment result returned by *LP-HTA* algorithm. As shown in the algorithm, we only set one of x_{ij1}, x_{ij2} and x_{ij3} to be 1, and set the other two to be 0, so that Constraints $C4$ and $C5$ in HTA problem are certainly satisfied.

Meanwhile, **Step 4** of the algorithm considers the deadline constraint of each task. That is, x_{ijl} is set to be 1 only if $t_{ijl} \le T_{ij}$, where $1 \le i \le n_r$, $1 \le j \le m$ and $l \in \{1,2,3\}$. Furthermore, the deadline constraint is also considered in **Step 5** and **Step 6** when we remove the tasks to base station or to the cloud. Thus, Constraint $C1$ of HTA problem is satisfied.

Finally, **Step 5** of the algorithm guarantees that $\sum_{j=1}^{m} C_{ij}x_{ij1} \le max_i$ for each mobile i $(1 \le i \le n_r)$, so that Constraint $C2$ is satisfied. Similarly, Constrain $C3$ in HTA problem is also meet by applying **Step 6** of algorithms

In summary, $\{x_{ijl} | 1 \le i \le n_r, 1 \le j \le m, l \in \{1,2,3\}\}$ satisfies all the constraints, so it is a feasible solution for the HTA problem.

3.2.2 Complexity

Next, we will analyze the complexity of *LP-HTA* algorithm.

In **Step 1**, the interior points method is used for solve the linear programming problem and obtain ξ, which is a vector with $3n_rm$ size. The complexity is $O((n_rm)^{3.5})$ according to [14]. In **Step 2** and **Step 3**, we construct the equivalent fractional matrix $\mathbf{X}$ and $\{\widehat{x}_{ijl} | 1 \le i \le n_r, 1 \le j \le m, l \in \{1,2,3\}\}$ according to ξ. The complexity of such construction is $O(n_rm)$. In **Step 4**, **Step 5** and **Step 6**, we adjust $\{\widehat{x}_{ijl} | 1 \le i \le n_r, 1 \le j \le m, l \in \{1,2,3\}\}$ according the deadline and resource constraints, and get $\{x_{ijl} | 1 \le i \le n_r, 1 \le j \le m, l \in \{1,2,3\}\}$ finally. The complexity of such adjustment is still $O(n_rm)$. **In summary**, the total complexity of *LP-HTA* algorithm is $O((n_rm)^{3.5})$.

3.2.3 Ratio Bound

Before discussing the ratio bound of *LP-HTA* algorithm, the following lemma is proved.

Lemma 1. The intermediate result $\{\widehat{x}_{ijl} | 1 \le i \le n_r, 1 \le j \le m, l \in \{1,2,3\}$ calculated by Step 3 in the *LP-HTA* algorithm satisfies that $\sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3} E_{ijl}\widehat{x}_{ijl} \le 3\sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3} E_{ijl}x_{ijl}^{OPT}$, where $\{x_{ijl}^{OPT} | 1 \le i \le n_r, 1 \le j \le m, l \in \{1,2,3\}\}$ be the optimal assignment for HTA problem.

Proof. According to the definition of relax linear programming problem, *i.e.* Problem $P2$, we have $\mathbf{X}[i,j,1] + \mathbf{X}[i,j,2] + \mathbf{X}[i,j,3] = 1$. Thus, $\max_{l=1,2,3}\{\mathbf{X}[i,j,l]\} \ge \frac{1}{3}$.

Let r satisfies that $\mathbf{X}[i,j,r] = \max_{l=1,2,3}\{\mathbf{X}[i,j,l]\}$. Based on **Step 3**, we set $\widehat{x}_{ijr} = 1$, and $\widehat{x}_{ijl} = 0$ if $l \neq r$. As $\mathbf{X}[i,j,r] = \max_{l=1,2,3}\{\mathbf{X}[i,j,l]\} \geq \frac{1}{3} = \frac{1}{3}\widehat{x}_{ijr}$, we have

$$\begin{aligned}&\sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3} E_{ijl}\widehat{x}_{ijl} = \sum_{i=1}^{n_r}\sum_{j=1}^{m} E_{ijr}\widehat{x}_{ijr} \\ &\leq 3\sum_{i=1}^{n_r}\sum_{j=1}^{m} E_{ijr}\mathbf{X}[i,j,r] \leq 3\sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3} E_{ijl}\mathbf{X}[i,j,l]\end{aligned} \tag{7}$$

where $\sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3} E_{ijl}\mathbf{X}[i,j,l]$ is the optimal result of the linear programming problem.

Since $\{x_{ijl}^{OPT} | 1 \leq i \leq n_r, 1 \leq j \leq m, l \in \{1,2,3\}\}$ be the optimal assignment of HTA problem, so that it is just a feasible solution of the linear programming problem. Therefore, $\sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3} E_{ijl}\mathbf{X}[i,j,l] \leq \sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3} E_{ijl}x_{ijl}^{OPT}$. Combing with Formula (7), we have $\sum\limits_{i=1}^{n_r}\sum\limits_{j=1}^{m}\sum\limits_{l=1}^{3} E_{ijl}\widehat{x}_{ijl} \leq 3\sum\limits_{i=1}^{n_r}\sum\limits_{j=1}^{m}\sum\limits_{l=1}^{3} E_{ijl}x_{ijl}^{OPT}$. □

According to *LP-HTA* algorithm, **Step 4**, **Step 5** and **Step 6** are used to adjust $\{\widehat{x}_{ijl} | 1 \leq i \leq n_r, 1 \leq j \leq m, l \in \{1,2,3\}$ to satisfy Constraints $C1$, $C2$ and $C3$. In another words, some tasks are migrated among mobile devices, the base station and the cloud to meet the constraints. Let Δ be the growth of energy cost caused by the above migration, and $E_{LP}^{(OPT)} = \sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3} E_{ijl}\mathbf{X}[i,j,l]$ be the optimal result of the linear programming algorithm $P2$, then the ratio bound of *LP-HTA* algorithm meets the following theorem.

Theorem 2. The ratio bound of *LP-HTA* Algorithm, *i.e.* R, is no more than $3 + \frac{\Delta}{E_{LP}^{(OPT)}}$.

Proof. According to Lemma 1, we have

$$\sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3} E_{ijl}\widehat{x}_{ijl} \leq 3\sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3} E_{ijl}x_{ijl}^{OPT},$$

where $\{\widehat{x}_{ijl} | 1 \leq i \leq n_r, 1 \leq j \leq m, l \in \{1,2,3\}\}$ is intermediate result obtained by **Step 3** of *LP-HTA* algorithm.

Since Δ is the growth of energy cost caused by the task migration and $E_{LP}^{(OPT)} = \sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3} E_{ijl}\mathbf{X}[i,j,l] \leq \sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3} E_{ijl}x_{ijl}^{OPT}$ based on **Lemma 1**, we have

$$R = \frac{\sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3} E_{ijl}\widehat{x}_{ijl} + \Delta}{\sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3} E_{ijl}x_{ijl}^{OPT}} \leq 3 + \frac{\Delta}{E_{LP}^{(OPT)}}.\square$$

The parameter Δ and $E_{LP}^{(OPT)}$ in **Theorem 2** can be determined during the execution of *LP-HTA* algorithm.

Furthermore, the energy cost by transmission usually is much larger than that cost by computation, so that $E_{ij1} < E_{ij2} < E_{ij3}$ since more data are transmitted if we use base station or cloud to deal with task $\mathcal{T}_{ij}$. Therefore, we have the following corollary.

Corollary 1. The ratio bound of the *LP-HTA* algorithm, denoted by R, satisfied that $R \leq \min\{\frac{\max_{1\leq i\leq n_r, 1\leq j\leq m} E_{ij3}}{\min_{1\leq i\leq n_r, 1\leq j\leq m} E_{ij1}}, 3 + \frac{\Delta}{E_{LP}^{(OPT)}}\}$, on condition that $E_{ij1} < E_{ij2} < E_{ij3}$.

Proof. Since $E_{ij1} < E_{ij2} < E_{ij3}$, we have $\sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3} E_{ijl}x_{ijl} \leq \sum_{i=1}^{n_r}\sum_{j=1}^{m} E_{ij3}$ and $\sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3} E_{ijl}x_{ijl}^{OPT} \geq \sum_{i=1}^{n_r}\sum_{j=1}^{m} E_{ij1}$, where $\{x_{ijl} | 1 \leq i \leq n_r, 1 \leq j \leq m, l \in \{1,2,3\}\}$ is the assignment result return by *LP-HTA* algorithm.

Thus, $R \leq \frac{\max_{1\leq i\leq n_r, 1\leq j\leq m} E_{ij3}}{\min_{1\leq i\leq n_r, 1\leq j\leq m} E_{ij1}}$. That is, $R \leq \min\{\frac{\max_{1\leq i\leq n_r, 1\leq j\leq m} E_{ij3}}{\min_{1\leq i\leq n_r, 1\leq j\leq m} E_{ij1}}, 3 + \frac{\Delta}{E_{LP}^{(OPT)}}\}$ according to Theorem 2. □

Since $R \leq \min\{\frac{\max_{1\leq i\leq n_r, 1\leq j\leq m} E_{ij3}}{\min_{1\leq i\leq n_r, 1\leq j\leq m} E_{ij1}}, 3 + \frac{\Delta}{E_{LP}^{(OPT)}}\} \leq \frac{\max_{1\leq i\leq n_r, 1\leq j\leq m} E_{ij3}}{\min_{1\leq i\leq n_r, 1\leq j\leq m} E_{ij1}}$ according to Corollary 1, and $n_r m$ is quite large comparing with $\frac{\max_{1\leq i\leq n_r, 1\leq j\leq m} E_{ij3}}{\min_{1\leq i\leq n_r, 1\leq j\leq m} E_{ij1}}$, thus the ratio bound of *LP-HTA* algorithm can be regarded as a constant.

4 Divisible Task Assignment Algorithm

Besides the holistic computation tasks, there also exist many divisible tasks in a MEC system. We call a task to be divisible if and only if it can be implemented distributedly, *i.e.* the final result can be obtained by aggregating the partial results. For example, some statistics calculations, such as *Sum* or *Count*, can be regarded as divisible tasks. Considering that the partial results are always much smaller than the raw data, therefore, it will save much energy if we process such tasks in distributed manner. Furthermore, it is also useful to protect privacy if the tasks are processed distributedly in a MEC system [15].

According to the above discussion, $\{\mathcal{T}_{ij} | 1 \leq i \leq n, 1 \leq j \leq m\}$ stands for the task set in a MEC system, and thus, $D = \bigcup_{1\leq i\leq n, 1\leq j\leq m}(LD_{ij} \bigcup ED_{ij})$ denotes the total data set required to be processed. Let $D = \{d_1, d_2, ..., d_M\}$, where d_i $(1 \leq i \leq M)$ is regarded as a data block that can be determined by [16]. Meanwhile, we also use D_i to denote the local data owned by mobile device i, where $D \subseteq \bigcup_{i=1}^{n} D_i$ and $D_i \bigcap D_j$ may not be empty since the monitoring regions of two devices may overlap with each other.

In order to reduce energy cost and avoid the raw data transmission as much as possible, the divisible tasks should be rearranged so that each mobile device only needs to deal with the local data, and does not have to consider the external data. Therefore, the following two sub problems should be considered during the divisible task assignment.

(1). How to determine the responsible dataset for each mobile device $1 \leq i \leq n$?

(2). How to arrange the tasks based on the above result?

Obviously, the first problem is more critical during the task arrangement in a MEC system. The following two subsections will discuss it based on different optimization goals, moreover, the last subsections will provide the solution of the second problem.

4.1 Data Division for Minimizing the Parallel Processing Time

In order to determine the responsible dataset for each mobile device i $(1 \leq i \leq n)$, a division of D, denoted by $\mathcal{C}^{(\mathcal{T})} = \{C_1^{(T)}, C_2^{(T)}, ..., C_n^{(T)}\}$ should be firstly obtained, where $C_i^{(T)}$ is the subset assigned to mobile device i for processing, and $C_i^{(T)} \bigcap C_j^{(T)} = \emptyset$ for any $1 \leq i \neq j \leq n$ to avoid the duplicate computation.

Considering that each subset of D can be processed in parallel manner by different mobile devices, so that the first optimization aim of dividing D is to minimizing the parallel processing time.

Let y_{ri} be a indicator variable satisfies that $y_{ri} = 1$ if and only if d_r is assigned to mobile device i. Otherwise, $y_{ri} = 0$, and at_{ri} be the time cost by such assignment, where $1 \leq i \leq n$, $d_r \in D$. Specifically, at_{ri} includes the data block retrieving time and processing time, which can be determined by the transmission model and computation model in Section 2. In order to obtain the minimized parallel processing time on conditional that data set D is totally covered, the following problem should be solved.

$$
\begin{aligned}
&P3: \min\ T \\
&s.t.\ \sum_{i=1}^{n} y_{ri} = 1, && 1 \leq r \leq M, \\
&\quad \sum_{r=1}^{M} y_{ri} at_{ri} \leq T, && 1 \leq i \leq n, \\
&\quad y_{ri} \in \{0,1\}, && 1 \leq r \leq M, 1 \leq i \leq n.
\end{aligned}
$$

Suppose that $Y^{opt} = \{y_{ri}^{opt} | 1 \leq r \leq M, 1 \leq i \leq n\}$ be the optimal solution of Problem $P3$, and $T^{opt} = max_{i=1}^{n}\{\sum_{r=1}^{M} y_{ri}^{opt} at_{ri}\}$. We notice that $P3$ is equal to the problem of Scheduling on Unrelated Parallel Machine, which is NP-hard [17]. Therefore, we will provide an approximation algorithm, named as *DTA-Time*, to solve $P3$, which consists of the following five steps.

Step 1. The lower and upper bound of T^{opt} should be determined. We use the greedy algorithm to obtain a feasible solution of $P3$, denoted by $Y^g = \{y_{ri}^g | 1 \leq r \leq M, 1 \leq i \leq n\}$ and $T^g = max_{i=1}^{n}\{\sum_{r=1}^{M} y_{ri}^g at_{ri}\}$. Obviously, T^g is a upper bound of T^{opt}. Meanwhile, $\lfloor \frac{T^g}{n} \rfloor$ is the lower bound of T^{opt} because the worst case is that the greedy algorithm assigns all the data block to one mobile device and the optimal one assigns them uniformly.

Step 2. Set $C_i^{(T)} = \emptyset$ for all $1 \leq i \leq n$ and $D_{remain} = D$, $M_{assign} = \emptyset$ Relax $P3$ as a family of linear programs, denoted by $LP(P3)$, as follows.

$$
\begin{aligned}
&LP(P3): \min\ T \\
&s.t.\ \sum_{i=1}^{n} y_{ri} = 1, && 1 \leq r \leq M, \\
&\quad \sum_{r:(r,i)\in S_C} y_{ri} at_{ri} \leq T, && 1 \leq i \leq n, \\
&\quad 0 \leq y_{ri} \leq 1, && (r,i) \in S_C.
\end{aligned}
$$

where $S_C = \{(r,i) | at_{ri} \leq T\}$. Find the smallest T^* that makes $LP(P3)$ to have a feasible solution by binary searching in interval $[\lfloor \frac{T^g}{n} \rfloor, T^{(g)}]$. Using the interior points method algorithm [14] to obtain an extreme point solution of $LP(P3)$, denoted by $Y^{LP} = \{y_{ri}^{LP} | 1 \leq r \leq M, 1 \leq i \leq n\}$.

Step 3. For $\forall d_r \in D$, if $\exists i \in [1,n]$ satisfies that $y_{ri}^{LP} = 1$, then $C^(T)_i = C^(T)_i \cup \{d_r\}$, *i.e.* assign d_r to mobile device i, and set $D_{remain} = D_{remain} - \{d_r\}$, $M_{assign} = M_{assign} \cup \{i\}$.

Step 4. Let $H_i = \{d_r | y_{ri}^{LP} > 0 \wedge d_r \in D_{remain}\}$, and the following steps are carried out circularly until $D_{remain} = \emptyset$.

1) *Case 1.* If $\exists j \in [1,n]$ satisfying that $|H_j| = 1$ and $H_j = \{d_q\}$, then set $C_j^{(T)} = C_j^{(T)} \cup \{d_q\}$, $D_{remain} = D_{remain} - \{d_q\}$, $M_{assign} = M_{assign} \cup \{j\}$, and $H_l = H_l - \{d_q\}$ for all $d_q \in H_l$ and $l \notin M_{assign}$;
2) *Case 2.* Otherwise, random select an available mobile device k from $1, 2, ..., n - M_{assign}$, and select an an item from H_k, denoted by d_p, set $C_k^{(T)} = C_k^{(T)} \cup \{d_p\}$, $D_{remain} = D_{remain} - \{d_p\}$, $M_{assign} = M_{assign} \cup \{k\}$, and $H_l = H_l - \{d_p\}$ for all $d_p \in H_l$ and $l \notin M_{assign}$;

Step 5. Return $\mathcal{C}^{(\mathcal{T})} = \{C_1^{(T)}, C_2^{(T)}, ..., C_n^{(T)}\}$.

DTA-Time algorithm is polynomial, and the following two theorems verify its correctness and ratio bound.

Theorem 3. $\cup_{i=1}^{n}\{C_i^{(T)}\} = D$, where $\mathcal{C}^{(\mathcal{T})} = \{C_1^{(T)}, ..., C_n^{(T)}\}$ is the output of *DTA-Time* algorithm.

Proof. Based on the result of linear programming in Step 2, *i.e.* $Y^{LP} = \{y_{ri}^{LP} | 1 \leq r \leq M, 1 \leq i \leq n\}$, we construct a bipartite graph, G_B, with nodes $\{d_r | d_r \in D\} \cup \{1, 2, ..., n\}$, where d_r and i has an edge in G_B if and only if $y_{ri}^{LP} > 0$. Since Problem $LP(P3)$ requires that $\sum_{i=1}^{n} y_{ri}^{LP} = 1$, then all the data blocks are contained in G_B.

Firstly, for all the data blocks in set $\{d_r | \exists i \in [1,n] \wedge y_{ri}^{LP} = 1\}$ are completed assigned by Step 3 in algorithm.

Secondly, let G_{remain} be the induced subgraph of G_B determined by the nodes $\{d_r | d_r \in D_{remain}\} \cup \{1, 2, ..., n\}$. According to [14], both of G_B and G_{remain} are the pseudo-forests, such as shown in Fig.(2).

For any $d_q \in G_{remain}$, its degree is at least 2 since $d_q \in D_{remain}$ and every data blocks that can be integrally assigned have been delete from D_{remain} in Step 3. Thus, each data block node either has a leaf neighbor (*e.g.* d_{r_1}) or belongs to a circle (*e.g.* d_{r_2}) in G_{remain} based on the properties of pseudo-forest. Furthermore, all leaves in G_{remain} must be mobile devices.

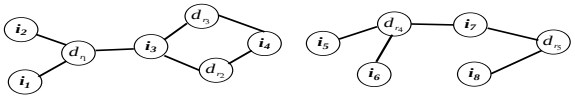


Fig. 2. An example of pseudo-forest

During the loop execution of Step 4, if d_q has a leaf neighbor, which must be a mobile device, then it will be assigned such mobile device based on *Case 1* of Step 4. Thus, all data blocks belong to such category are totally assigned.

On the other hand, if d_q belongs to a circle, the number of data blocks equals to the number of mobile devices on such circle as G_{remain} is a bipartite graph. Thus, based on *Case 2* in Step 4, we can obtain the prefect matching of such circle, and all data blocks in a circle are entirely assigned.

In summary, each $d_r \in D$ has been assigned, so that $\cup_{i=1}^{n}\{C_i^{(T)}\} = D$. □

Theorem 4. The ratio bound of *DTA-Time* algorithm is 2.

Proof. Let $\widehat{T}$ be the maximum processing time determined by *DTA-Time* algorithm. We notice that only Steps 3 and 4 of the algorithms will influence $\widehat{T}$ since the data block assignments are generated in these two steps. Let T_1 and T_2 be the maximum processing time caused by Step 3 and Step 4 respectively, then $\widehat{T} = \max_{i=1}^{n}\{\sum_{d_r \in C_i^{(T)}} at_{ri}\} \leq T_1 + T_2$.

According to Step 2 of *DTA-Time* algorithm, we relax Problem $P3$ as Problem $LP(P3)$, and T^* is the smallest value to make $LP(P3)$ to have a feasible solution, so that $T^* \leq T^{opt}$. Meanwhile, we also have $T_1 = \max_{1\leq i\leq n}\{\sum_{r:(r,i)\in S_C \wedge y_{ri}^{LP}=1} y_{ri}^{LP} at_{ri}\}$ according to Step 3, thus, $T^{opt} \geq T^* = \max_{1\leq i\leq n}\{\sum_{r:(r,i)\in S_C} y_{ri}^{LP} at_{ri}\} \geq \max_{1\leq i\leq n}\{\sum_{r:(r,i)\in S_C \wedge y_{ri}^{LP}=1} y_{ri}^{LP} at_{ri}\} = T_1$.

Furthermore, each mobile device at most assigned one extra data block as shown in Step 4, thus, $T_2 \leq max_{(p,j)\in S_C} at_{pj} \leq T^* \leq T^{opt}$. as $S_C = \{(r,i) | at_{ri} \leq T^*\}$.

In summary, $\widehat{T} \leq T_1 + T_2 \leq 2T^{opt}$, *i.e.* the ratio bound of *DTA-Time* algorithm is 2. □

4.2 Data Division for Minimizing Involved Mobile Devices

Secondly, in order to save energy for majority mobile devices and reduce the transmission cost as much as possible, some applications may require to minimize the number of devices for dealing with the tasks. Based on such motivation, the *Optimal Coverage of D with Smallest Set Number* is defined as follows.

Definition 1. $\mathcal{C}^{(\mathcal{N})} = \{C_{l_1}^{(N)}, C_{l_2}^{(N)}, ..., C_{l_r}^{(N)}\}$ is called as an *Optimal Coverage of D with Smallest Set Number* if and only if $\mathcal{C}$ satisfies: **(1)** $C_{l_i}^{(N)} \neq \emptyset$, $C_{l_i}^{(N)} \subset D \bigcap D_{l_i}$; **(2)** $C_{l_i}^{(N)} \bigcap C_{l_j}^{(N)} = \emptyset$ for any $1 \leq i \neq j \leq n$, and $\bigcup_{i=1}^{l} C_{l_i}^{(N)} = D$; **(3)** For all $\mathcal{C}'$ which satisfies the above two conditions, we have $|\mathcal{C}'| \geq |\mathcal{C}^{(\mathcal{N})}|(= r)$.

Considering D is a universe, and $\mathcal{S}^{(\mathcal{N})} = \{UD_1, UD_2, ..., UD_n\}$ is a family of subsets of D, where $UD_i = D \bigcap D_i$ as given in Section 4.1. Therefore, the problem of calculating *Optimal Coverage of D with Smallest Set Number*, *i.e.* $\mathcal{C}^{(\mathcal{N})} = \{C_{l_1}^{(N)}, C_{l_2}^{(N)}, ..., C_{l_r}^{(N)}\}$, can be regarded as the Set Cover Problem whose input are D and $\mathcal{S}^{(\mathcal{N})}$. Since the Set Cover Problem is proved to be NP-complete in [18], therefore, it is hard to obtain $\mathcal{C}^{(\mathcal{N})} = \{C_{l_1}^{(N)}, C_{l_2}^{(N)}, ..., C_{l_r}^{(N)}\}$ in polynomial time. The following part will provide a greedy algorithm, named as *DTA-Number*, to get approximation result The algorithm consists of three steps.

First, let $UD_i = D \bigcap D_i$ for all $1 \leq i \leq n$, and $\mathcal{C}^{(\mathcal{N})} = \emptyset$.

Second, determine r by $r = \max_{1 \leq i \leq n} |UD_i \cap D|$, let $\mathcal{C}^{(\mathcal{N})} = \mathcal{C}^{(\mathcal{N})} \bigcup \{UD_r \cap D\}$ and $D = D - (UD_r \cap D)$;

Third, repeat Step 2 until $D = \emptyset$. Return $\mathcal{C}^{(\mathcal{N})}$.

Obversely, the complexity of the above algorithm is $O(n)$. Its ration bound is $O(H(n)) = O(\ln n)$ according to [18], where $H_n = 1 + \frac{1}{2} + \cdots + \frac{1}{n}$.

4.3 Task Rearrangement Method

The tasks in set $\{\mathcal{T}_{rl} | C_i^{(S)} \bigcap (LD_{rl} \bigcup ED_{rl}) \neq \emptyset\}$ (or $\{\mathcal{T}_{rl} | C_i^{(N)} \bigcap (LD_{rl} \bigcup ED_{rl}) \neq \emptyset\}$) will be rearranged to mobile device i after obtaining $\mathcal{C}^{(\mathcal{T})} = \{C_1^{(T)}, C_2^{(T)}, ..., C_l^{(T)}\}$ (or $\mathcal{C}^{(\mathcal{N})} = \{C_1^{(N)}, C_2^{(N)}, ..., C_l^{(N)}\}$) Then, the *LP-HTA* algorithm in Section 3 is applied to schedule these new tasks and obtain the partial results. Finally, the partial results are aggregated according to users' requirements.

As only partial results are required to transmit in MEC system, much energy will be saved.

5 Ordered Task Assignment Algorithm

Although the algorithms proposed in above two sections can solve the task assignment problem efficiently in MEC systems, they ignore the order relationship among tasks. In practice , many tasks in a MEC system are in order. For example, in an application of automatic driving, there always exist three important tasks: environment representation, motion planning and vehicle controlling [19]. To represent the environment, a vehicle needs to obtain the surrounding environmental information through various sensors. Then, the vehicle would analyze the driving track of surrounding objects, including vehicles and pedestrians, based on the environmental representation result, then determine a proper motion plan in the further. Finally, a specific controlling algorithm is utilized to control the vehicle according to motion plan. Such three tasks are ordered. Similarly, in the application of Augmented Reality [20], there also exist many tasks that are partially ordered. In above situation, the algorithms given in above two sections are not applicable, so that new method is required.

5.1 New System Model for Ordered Tasks

In order to describe the order relationship among tasks, the system model in Section 2 should be updated.

In this section, we still use the tuple $\mathcal{T}_{ij} = (op_{ij}, LD_{ij}, ED_{ij}, L_{ij}, C_{ij}, T_{ij})$ to stand for j-th task of the user U_i, where $1 \leq i \leq n$, $1 \leq j \leq N_i$, and N_i is the number of tasks proposed by user U_i. Meanwhile, a directed acyclic graph, *i.e.* $\mathcal{G} = (V, E, \Gamma)$, is utilized to denote the order relationship among tasks, where $V = \{\mathcal{T}_{ij} | 1 \leq i \leq n, 1 \leq j \leq N_i\}$, and E describes the order relationship between two tasks. That is, $(\mathcal{T}_{ij}, \mathcal{T}_{rs}) \in E$ if and only if (*iff* for short) $\mathcal{T}_{ij}$ needs transmit the intermediate result, whose size is $\gamma_{ij,rs}$, to $\mathcal{T}_{rs}$, and For the convenience of discussion, we also define the immediate predecessor and successor sets for each $\mathcal{T}_{ij}$, *i.e.* $pred(\mathcal{T}_{ij})$ and $suc(\mathcal{T}_{ij})$, where $pred(\mathcal{T}_{ij}) = \{\mathcal{T}_{uv} | (\mathcal{T}_{uv}, \mathcal{T}_{ij}) \in E\}$ and $suc(\mathcal{T}_{ij}) = \{\mathcal{T}_{rs} | (\mathcal{T}_{ij}, \mathcal{T}_{rs}) \in E\}$.

In the following section, we assume that each task in G is holistic. If there exist some divisible tasks, we can use the method in Section 4 to decompose them into holistic ones.

Similar to Section 2, a boolean variable, *i.e.* x_{ijl}, is utilized to denote the processor responding for $\mathcal{T}_{ij}$. That is, $x_{ijl} = 1$ if and only if $\mathcal{T}_{ij}$ is processed by mobile device l, where $l = 0, 1, ..., k + n$ since the transmission of the intermediate results should also be considered. Specifically, $\mathcal{T}_{ij}$ is processed by the cloud *iff* $l = 0$, $\mathcal{T}_{ij}$ is processed by the base station B_q $(1 \leq q \leq k)$ *iff* $l = q$, $\mathcal{T}_{ij}$ is processed by the mobile device MD_p $(1 \leq p \leq n)$ *iff* $l = p + k$. $\mathcal{T}_{ij}$ is called as the **Start Task** *iff* $pred(\mathcal{T}_{ij}) = \emptyset$, similarly, $\mathcal{T}_{ij}$ is called as the **End Task** *iff* $suc(\mathcal{T}_{ij}) = \emptyset$.

5.1.1 New Computation Model for Ordered Tasks

The computation model of a task does not only depend on the amount of raw data it requires, but also depends on the size of intermediate results it receives. For the convenience of discussion, we use η_{ij} to denote such size, then $\eta_{ij} = \sum_{\mathcal{T}_{rs} \in pred(\mathcal{T}_{ij})} \gamma_{rs,ij}$. Thus, new computation time and energy cost for processing $\mathcal{T}_{ij}$ are updated as follows.

$$t_{ijl}^{(C)} = \begin{cases} \frac{\lambda_{ij0}(\alpha_{ij} + \beta_{ij} + \eta_{ij})}{f_c}, & \text{If } l = 0 \\ \frac{\lambda_{ijl}(\alpha_{ij} + \beta_{ij} + \eta_{ij})}{f_s}, & \text{If } 1 \leq l \leq k \\ \frac{\lambda_{ijl}(\alpha_{ij} + \beta_{ij} + \eta_{ij})}{f_i}, & \text{If } k+1 \leq l \leq k+n \end{cases} \tag{8}$$

$$e_{ijl}^{(C)} = \begin{cases} 0, & \text{If } 0 \leq l \leq k \\ \kappa \lambda_{ij}^{(m)} (\alpha_{ij} + \beta_{ij} + \eta_{ij}) f_i^2, & \text{If } k+1 \leq l \leq k+n \end{cases} \tag{9}$$

where $l = 0$ means that $\mathcal{T}_{ij}$ are carried out in cloud, $1 \leq l \leq k$ means that $\mathcal{T}_{ij}$ is processed by the l-th base station, and $k + 1 \leq l \leq k + n$ means that $\mathcal{T}_{ij}$ is handled by the $l - k$-th mobile device.

5.1.2 New Transmission Model for Ordered Tasks

Let $\mathcal{T}_{ij}$ denote a task in G. Different from the previous discussion, the data required by $\mathcal{T}_{ij}$ not only include in local and external data, but also consist of the intermediate results from the immediate predecessors of $\mathcal{T}_{ij}$. Therefore, the data required by a given task, $\mathcal{T}_{ij}$, could be distributed in cloud, base stations or mobile devices. Therefore, more complicated situations need to be considered.

Suppose that l_1 is the owner of a dataset required by $\mathcal{T}_{ij}$, l_2 and X denoted the destination and the size of such dataset. Actually, l_2 is also the processor that finally deal with $\mathcal{T}_{ij}$ since $\mathcal{T}_{ij}$ is holistic. We use $T^{(R)}_{l_1,l_2}(X)$ to denote the transmission time of such dataset, and use $E^{(R)}_{l_1,l_2}(X)$ to denote the energy cost for transmitting such dataset from l_1 to l_2. According to the above discussion, l_1 could be a mobile device, a base station or the cloud. Considering the different value of l_1, the following cases need to be considered.

Case 1. If $l_1 = l_2$, we have

$$T^{(R)}_{l_1,l_2}(X) = 0, \;\; E^{(R)}_{l_1,l_2}(X) = 0$$

since the required data set is stored locally and it does not need any data transmission.

Case 2. If $l_1 \neq l_2$ and $l_1 = 0$, *i.e.* the required data set is stored in the cloud, then we need to take two situations into account. First, if $1 \leq l_2 \leq k$, then the destination of such data set is a base station, so that the transmission time and energy cost satisfy that $T^{(R)}_{l_1,l_2}(X) = t_{B,C}(X)$ and $E^{(R)}_{l_1,l_2}(X) = e_{B,C}(X)$. Otherwise, if $k+1 \leq l_2 \leq k+n$, then the destination of the data set is a mobile device, so that the transmission time and energy cost satisfy that $T^{(R)}_{l_1,l_2}(X) = t_{B,C}(X) + \frac{X}{r^{(D)}_{l_2}}$ and $E^{(R)}_{l_1,l_2}(X) = e_{B,C}(X) + e^{(R)}_{l_2}(X)$ based on the analysis in Section 2. In summary, we have,

$$T^{(R)}_{l_1,l_2}(X) = \begin{cases} t_{B,C}(X), & \text{If } l_1 = 0, 1 \leq l_2 \leq k \\ t_{B,C}(X) + \frac{X}{r^{(D)}_{l_2}}, & \\ \qquad \text{If } l_1 = 0, k+1 \leq l_2 \leq k+n \end{cases}$$

$$E^{(R)}_{l_1,l_2}(X) = \begin{cases} e_{B,C}(X), & \text{If } l_1 = 0, 1 \leq l_2 \leq k \\ e_{B,C}(X) + e^{(R)}_{l_2}(X), & \\ \qquad \text{If } l_1 = 0, k+1 \leq l_2 \leq k+n \end{cases}$$

Case 3. If $l_1 \neq l_2$ and $1 \leq l_1 \leq k$, then the required dataset is stored in a base station. Thus, there exist four situations for consideration according to the value of l_2. First, if $l_2 = 0$, then the destination of such dataset is the cloud. Second, if $1 \leq l_2 \leq k$, then the destination of the dataset is other base station rather than l_1. Third, if $k+1 \leq l_2 \leq k+n$, then the destination of the dataset is a mobile device, which could belong to the same cluster with l_1 or not. Therefore, $T^{(R)}_{l_1,l_2}(X)$ and $E^{(R)}_{l_1,l_2}(X)$ satisfies the following formulas.

$$T^{(R)}_{l_1,l_2}(X) = \begin{cases} t_{B,C}(X), \text{ If } 1 \leq l_1 \leq k, l_2 = 0 \\ t_{B,B}(X), \text{ If } 1 \leq l_1 \leq k, 1 \leq l_2 \leq k, l_1 \neq l_2 \\ \frac{X}{r^{(D)}_{l_2}}, \quad \text{If } 1 \leq l_1 \leq k, k+1 \leq l_2 \leq k+n \\ \qquad \wedge\, l_1 \text{ and } l_2 \text{ are in the same cluster} \\ \frac{X}{r^{(D)}_{l_2}} + t_{B,B}(X), \\ \qquad \text{If } 1 \leq l_1 \leq k, k+1 \leq l_2 \leq k+n \\ \quad \wedge\, l_1 \text{ and } l_2 \text{ are not in the same cluster} \end{cases}$$

$$E^{(R)}_{l_1,l_2}(X) = \begin{cases} e_{B,C}(X), \text{ If } 1 \leq l_1 \leq k, l_2 = 0 \\ e_{B,B}(X), \text{ If } 1 \leq l_1 \leq k, 1 \leq l_2 \leq k, l_1 \neq l_2 \\ e^{(R)}_{l_2}(X), \text{ If } 1 \leq l_1 \leq k, k+1 \leq l_2 \leq k+n \\ \qquad \wedge\, l_1 \text{ and } l_2 \text{ are in the same cluster} \\ e^{(R)}_{l_2}(X) + e_{B,B}(X), \\ \qquad \text{If } 1 \leq l_1 \leq k, k+1 \leq l_2 \leq k+n \\ \quad \wedge\, l_1 \text{ and } l_2 \text{ are not in the same cluster} \end{cases}$$

Case 4. If $l_1 \neq l_2$ and $k+1 \leq l_1 \leq k+n$, that means l_1 is a mobile device. Similar to the analysis in **Case 3**, the destination of the dataset, l_2, could be the cloud, a base station in the same or different cluster, and a mobile device in the same or different cluster. Therefore, there totally exist five situations needed to be considered, and $T^{(R)}_{l_1,l_2}(X)$ and $E^{(R)}_{l_1,l_2}(X)$ satisfies Formulas (10) and (11).

$$T^{(R)}_{l_1,l_2}(X) = \begin{cases} \frac{X}{r^{(U)}_{l_1}} + t_{B,C}(X), \\ \qquad \text{If } k+1 \leq l_1 \leq k+n, l_2 = 0 \\ \frac{X}{r^{(U)}_{l_1}}, \quad \text{If } k+1 \leq l_1 \leq k+n, 1 \leq l_2 \leq k \\ \qquad \wedge\, l_1 \text{ and } l_2 \text{ are in the same cluster} \\ \frac{X}{r^{(U)}_{l_1}} + t_{B,B}(X), \\ \qquad \text{If } k+1 \leq l_1 \leq k+n, 1 \leq l_2 \leq k \\ \quad \wedge\, l_1 \text{ and } l_2 \text{ are not in the same cluster} \\ \frac{X}{r^{(U)}_{l_1}} + \frac{X}{r^{(D)}_{l_2}}, \\ \quad \text{If } k+1 \leq l_1 \leq k+n, k+1 \leq l_2 \leq k+n, \\ \quad \wedge\, l_1 \text{ and } l_2 \text{ are in the same cluster} \\ \frac{X}{r^{(U)}_{l_1}} + \frac{X}{r^{(D)}_{l_2}} + t_{B,B}(X), \\ \quad \text{If } k+1 \leq l_1 \leq k+n, k+1 \leq l_2 \leq k+n \\ \quad \wedge\, l_1 \text{ and } l_2 \text{ are not in the same cluster} \end{cases} \tag{10}$$

$$E^{(R)}_{l_1,l_2}(X) = \begin{cases} e^{(T)}_{l_1}(X) + e_{B,C}(X), \\ \qquad \text{If } k+1 \leq l_1 \leq k+n \wedge l_2 = 0 \\ e^{(T)}_{l_1}(X), \text{ If } k+1 \leq l_1 \leq k+n, 1 \leq l_2 \leq k \\ \qquad \wedge\, l_1 \text{ and } l_2 \text{ are in the same cluster} \\ e^{(T)}_{l_1}(X) + e_{B,B}(X), \\ \qquad \text{If } k+1 \leq l_1 \leq k+n, 1 \leq l_2 \leq k \\ \quad \wedge\, l_1 \text{ and } l_2 \text{ are not in the same cluster} \\ e^{(T)}_{l_1}(X) + e^{(R)}_{l_2}(X), \\ \quad \text{If } k+1 \leq l_1 \leq k+n, k+1 \leq l_2 \leq k+n, \\ \quad \wedge\, l_1 \text{ and } l_2 \text{ are in the same cluster} \\ e^{(T)}_{l_1}(X) + e^{(R)}_{l_2}(X) + e_{B,B}(X), \\ \quad \text{If } k+1 \leq l_1 \leq k+n, k+1 \leq l_2 \leq k+n \\ \quad \wedge\, l_1 \text{ and } l_2 \text{ are not in the same cluster} \end{cases} \tag{11}$$

As shown in Fig.(3), if the source of the required dataset is MD_2, *i.e.* $l_1 = MD_2$, then possible destination of such dataset could be C_0, B_1, B_2, MD_1, $MD_3 \ldots$. If $l_2 = MD_3$, the transmission time and energy consumption equal to $\frac{X}{r^{(U)}_{l_1}} + \frac{X}{r^{(D)}_{l_2}} + t_{B,B}(X)$ and $e^{(T)}_{l_1}(X) + e^{(R)}_{l_2}(X) + e_{B,B}(X)$, respectively.

5.1.3 Problem Definition

Based on the new computation and transmission models, the problem of Ordered Task Assignment (OTA) in a MEC system is presented as follows, which is similar to HTA problem with slightly adjusting its input and output.

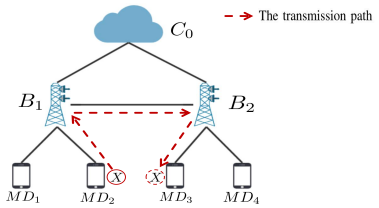


Fig. 3. An example for determining transmission time and cost.

Input:

1) A MEC system with n mobile devices,$\{1, 2, ...n\}$, and k base stations, $\{B_1, B_2, ..., B_k\}$;
2) A DAG, $\mathcal{G}$, which represents the order relationship between tasks;
3) Basic frequencies of mobile devices, base stations and the cloud, $\{f_i|1 \le i \le n\}$, f_s, f_c;
4) The Network parameters, $r_i^{(U)}$, $r_i^{(D)}$, $t_{B,B}(X)$, $t_{B,C}(X)$ $e_i^{(T)}(X)$, $e_i^{(R)}(X)$, $e_{B,B}(X)$ and $e_{B,C}(X)$;
5) The limitations on computation resources of each mobile device and base station $\{max_l|1 \le l \le k+n\}$.

Output: The task assignment plan, $S = \{(\mathcal{T}_{ij}, X_{ij}, t_{ij})|1 \le i \le n, 1 \le j \le N_i\}$, with the smallest energy cost, where $X_{ij} = [x_{ij0}, x_{ij1}..., x_{ij(k+n)}]$, and S satisfies the following conditions.

$$\sum_{l=0}^{k+n} x_{ijl} = 1 \wedge x_{ijl} \in \{0,1\} \quad \forall 1 \le i \le n, 1 \le j \le N_i \quad \text{(C1)}$$

$$t_{ij} \ge \max\{\sum_{l=0}^{k+n} T^{(R)}_{(k+i),l}(\alpha_{ij})x_{ijl}, \sum_{l=0}^{k+n} T^{(R)}_{L_{ij},l}(\beta_{ij})x_{ijl}, \sum_{l_1=0}^{k+n}\sum_{l_2=0}^{k+n} T^{(R)}_{l_1,l_2}(\gamma_{rs,ij})x_{rsl_1}x_{ijl_2} + t_{rs} + \sum_{l=0}^{k+n} t^{(C)}_{rsl}x_{rsl}\}$$
$$\forall 1 \le i \le n, 1 \le j \le N_i, \mathcal{T}_{rs} \in pred(\mathcal{T}_{ij}) \quad \text{(C2)}$$

$$\sum_{l=0}^{k+n} t^{(C)}_{ijl}x_{ijl} + t_{ij} \le T_{ij} \quad \forall 1 \le i \le n, 1 \le j \le N_i \quad \text{(C3)}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{N_i} C_{ij}x_{ijq} \le max_q \quad \forall 1 \le q \le k+n \quad \text{(C4)}$$

where $C1$ indicates that tasks can only be executed on one processor. $C2$ shows the constraint on the start time of each task, which means that each task should wait for all the data, including the local data, external data and intermediate results, to be ready before starting. $C3$ and $C4$ present the deadline and resource constraints of each task, which is same as the HTA problem.

Obviously, the Ordered Task Assignment (OTA) problem can be formulated as a mixed-integer nonlinear programming problem, which is NP-hard [21] [22]. Therefore, we propose heuristic algorithm based on greedy strategy in the next section as it is good choice for scheduling tasks in a distributed system [23].

5.2 Heuristics Ordered Task Assignment Algorithm

To solve the OTA problem and obtain the near optimal assignment result, the following sub problems need to be considered.

Firstly, according to the definition in above section, the weight in $\mathcal{G}$ only provides the size of intermediate results transmitting between two tasks, but not include the time information, which is very important for the ordered task assignment. Therefore, the weights that implicate the time information of a DAG should be initiated before task assignment.

Secondly, we wish that the neighbor tasks are clustered together in order to reduce the transmission cost as much as possible. However, the size of each cluster should not be too large since it will exceed the processing ability of the mobile devices or the base stations. Thus, the problem of how to cluster the tasks is also very significant.

Thirdly, we need to assign the proper processors and start time for each cluster and task for returning the final assignment result.

Therefore, we use three phases to solve the above three problems, and finally get the Heuristics Ordered Tasks Assignment Algorithm based on Clustering (*HOTA-C* for short). The detailed solutions of these three phases ar presented in the following three subsections.

Algorithm 1: Heuristics Ordered Task Assignment Algorithm based on Clustering (*HOTA-C*)

Input: k, n, $\mathcal{G}$, max_l, λ, f_i, f_s, f_c, $r_i^{(U)}$, $r_i^{(D)}$, $t_{B,B}(X)$, $t_{B,C}(X)$, $e_i^{(T)}(X)$, $e_i^{(R)}(X)$, $e_{B,B}(X)$, $e_{B,C}(X)$, $\forall 1 \le i \le n, 1 \le l \le k+n$

Output: S

1 $\mathcal{G}'^{(0)}$ = Initiating the weights of a DAG ($\mathcal{G}$,max_l); //Discussed in ***Phase I***

2 $[CLUSTER, \mathcal{G}'^{(g)}]$= *Task Clustering Algorithm* ($\mathcal{G}'^{(0)}$,max_l); //Discussed in ***Phase II***

3 S = *Task Assignment Algorithm* ($CLUSTER$, $\mathcal{G}'^{(g)}$); //Discussed in ***Phase III***

4 Return S;

5.2.1 ***Phase I:*** *Initiating the weights of a DAG*

We notice that the input DAG, $\mathcal{G}$, does not carry on the time information in it. Therefore, the first step is to construct $\mathcal{G}'$ according to $\mathcal{G}$, where the weighted nodes and edges in $\mathcal{G}'$ indicate the time spent in computation and transmission for each task.

The construction of $\mathcal{G}'$ is as follows.

First, build a corresponding node for each $\mathcal{T}_{ij} \in V$, and its weight, denoted by $t^{(C)}_{ij}$, is calculated by $t^{(C)}_{ij} = \sum_{l=0}^{k+n} t^{(C)}_{ijl}x_{ijl}$.

Second, build a corresponding edge for each edge $(\mathcal{T}_{rs}, \mathcal{T}_{ij}) \in E$, and its weight, denoted by $t^{(R)}_{rs,ij}$, is calculated by $t^{(R)}_{rs,ij} = \sum_{l_1=0}^{k+n}\sum_{l_2=0}^{k+n} T^{(R)}_{l_1,l_2}(\gamma_{rs,ij})x_{rsl_1}x_{ijl_2}$.

According to the definition of OTA problem, $t^{(C)}_{ij}$ denotes the computation time of $\mathcal{T}_{ij}$, and $t^{(R)}_{rs,ij}$ denotes the time cost by transmitting the intermediate results from $\mathcal{T}_{rs}$ to $\mathcal{T}_{ij}$. Therefore, the assignment result is easily get if $\mathcal{G}'$ are determined. However, since all the $\{x_{ijl}|1 \le i \le n, 1 \le j \le N_i, 0 \le l \le k+n\}$ are unknown at the beginning, $\mathcal{G}'$ cannot be obtained directly. In our method, we adopt

the iterative approach for achieving $\mathcal{G}'$. Thus, $\mathcal{G}'$ need to be initiated firstly, and we call such initial DAG as $\mathcal{G}'^{(0)}$.

According to [24] [25] [26], it is usually to adopt the average value for initializing such computation and transmission time. There, we use $\bar{t_{ij}}^{(C)}$ and $\bar{t_{rs,ij}}^{(R)}$ to initial the weights of node $\mathcal{T}_{ij}$ and edge $(\mathcal{T}_{rs}, \mathcal{T}_{ij})$ in $\mathcal{G}'^{(0)}$ respectively, that is

$$\bar{t_{ij}}^{(C)} = \frac{1}{k+n+1} \sum_{l=0}^{k+n} t_{ijl}^{(C)}$$

$$\bar{t_{rs,ij}}^{(R)} = \frac{1}{(k+n+1)^2} \sum_{l_1=0}^{k+n} \sum_{l_2=0}^{k+n} T_{l_1,l_2}^{(R)}(\gamma_{rs,ij})$$

Based on $\mathcal{G}'^{(0)}$, the tasks are clustered and the weighted DAG is updated, which will be introduced in the following phases.

TABLE 1
Parameters of Tasks

Task	α_{ij}	β_{ij}	L_{ij}	C_{ij}	T_{ij}
$\mathcal{T}_{11}$	20	10	MD_2	30	20
$\mathcal{T}_{12}$	20	6	MD_3	30	26
$\mathcal{T}_{13}$	12	4	MD_4	36	32
$\mathcal{T}_{14}$	15	5	MD_2	40	50
$\mathcal{T}_{21}$	24	8	MD_1	32	28
$\mathcal{T}_{22}$	6	2	MD_3	40	46
$\mathcal{T}_{23}$	10	4	MD_4	30	60

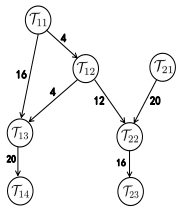


Fig. 4. The Order Relationship of Tasks.

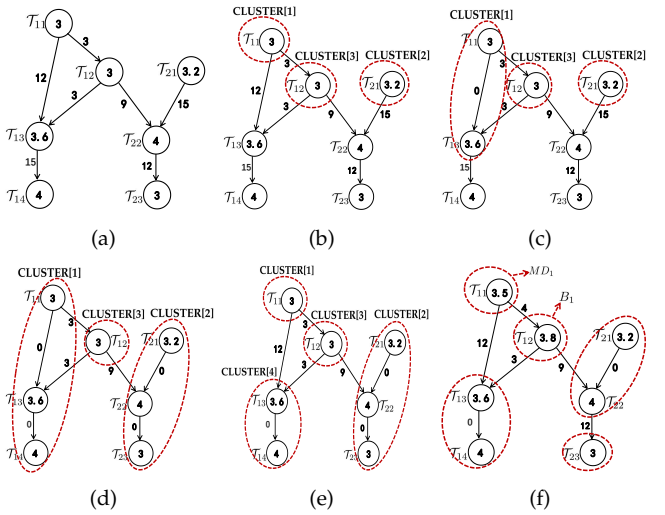


Fig. 5. An example of *HOTA-C*. (a) Phase I: $\mathcal{G}'^{(0)}$. (b) Phase II: The beginning of an iteration. (c) Phase II: The end of an iteration. (d) Phase III: Clustering result. (e) Phase III: The Division of a Cluster. (f) Phase III: Partial Assignment Result.

In order to make it more clearly, an example is present in Fig.(5), where the parameters of each task and their order relationship are given in Table I and Fig.(4), respectively. After calculating the average computation and transmission time for each task, we have $\mathcal{G}'^{(0)}$ shows in Fig.5(a).

5.2.2 ***Phase II:*** *Clustering the Ordered Tasks*

Next, we will discuss how to cluster the tasks based on $\mathcal{G}'^{(0)}$. According to [27], the definition of *tlevel* and *blevel* is firstly presented.

***Definition* 2.** For each task $\mathcal{T}_{ij}$ in an ordered task set, $tlevel(\mathcal{T}_{ij})$ and $blevel(\mathcal{T}_{ij})$ are called as its *tlevel* and *blevel* *iff* they satisfy:

$$tlevel(\mathcal{T}_{ij}) = \max_{\mathcal{T}_{rs} \in pred(\mathcal{T}_{ij})} (tlevel(\mathcal{T}_{rs}) + t_{rs}^{(C)} + t_{rs,ij}^{(R)})$$

$$blevel(\mathcal{T}_{ij}) = \max_{\mathcal{T}_{uv} \in suc(\mathcal{T}_{ij})} (blevel(\mathcal{T}_{uv}) + t_{ij}^{(C)} + t_{ij,uv}^{(R)})$$

$tlevel(\mathcal{T}_{ij}) = 0$ if $\mathcal{T}_{ij}$ is the **Start Task**, and $blevel(\mathcal{T}_{ij}) = t_{ij}^{(C)}$ if $\mathcal{T}_{ij}$ is the **End Task**.

Obviously, $tlevel(\mathcal{T}_{ij})$ is the weight of the longest path from the **Start Task** to $\mathcal{T}_{ij}$, and $blevel(\mathcal{T}_{ij})$ denotes the weight of the longest path from $\mathcal{T}_{ij}$ to the **End Task**.

For example, in Fig.5(a), $tlevel(\mathcal{T}_{13}) = \max(3+12, 3+3+3+3) = 15$ and $blevel(\mathcal{T}_{13}) = 3.6+15+4 = 22.6$.

Based on the above Definition 3 the ordered task clustering algorithm consists of four steps.

Step 1. Set $UC = \{\mathcal{T}_{ij} | 1 \le i \le n, 1 \le j \le N_i\}$, where UC contains the tasks that are not clustered. Calculate C_{max} by $C_{max} = \max_{1 \le l \le k+n}(max_l)$. Let $g = 0$ which stands for the times of iteration. Compute $blevel(\mathcal{T}_{ij})$ for each $\mathcal{T}_{ij} \in V$.

Step 2. Investigate each task $\mathcal{T}_{ij} \in UC$ satisfying that $pred(\mathcal{T}_{ij}) \cap UC = \emptyset$, which means that all the immediate predecessors have been clustered, compute its priority by $PRIO(\mathcal{T}_{ij}) = tlevel(\mathcal{T}_{ij}) + blevel(\mathcal{T}_{ij})$ according to [27] based on $\mathcal{G}'^{(g)}$, and select one with the highest priority to be the candidate for clustering in next step.

Step 3. Suppose that $\mathcal{T}_{ij}$ is the task selected in **Step 2**, then $\mathcal{T}_{ij}$ joins to $CLUSTER[q]$ if it satisfies the following conditions.

1) $CLUSTER[q] \cap pred(\mathcal{T}_{ij}) \neq \emptyset$, that is, there at least exists one of the immediate predecessors of $\mathcal{T}_{ij}$ belongs to $CLUSTER[q]$ so that the transmission time and cost is reduced.
2) $\sum_{\mathcal{T}_{rs} \in CLUSTER[q] \cap pred(\mathcal{T}_{ij})} \gamma_{rs,ij} \ge \alpha_{ij}$, *i.e.* the size of the intermediate results exceeds the size of the local data due to the same reason.
3) $\sum_{\mathcal{T}_{rs} \in CLUSTER[q]} C_{rs} + C_{ij} \le C_{max}$, which means that the total resources occupied by the tasks in $CLUSTER[q]$ should be bounded. We notice that the cluster will grow largely and exceed all the processing abilities of all the edge devices if its size is not constrained. Therefore, all the tasks in the cluster are transmitted to the cloud for processing, and it will dramatically increase the processing delay and transmission cost for each task. Thus, it is quite necessary for using the above constraint to limit the size of each cluster.
4) If there exist multiple clusters to satisfy the above three conditions, then calculate a new $tlevel^{(q)}(\mathcal{T}_{ij})$ for each candidate cluster $CLUSTER[q]$. That is, $tlevel^{(q)}(\mathcal{T}_{ij}) = \max\{\max_{\mathcal{T}_{rs} \in pred(\mathcal{T}_{ij}) \cap CLUSTER[q]}(tlevel(\mathcal{T}_{rs}) + t_{rs}^{(C)}), \max_{\mathcal{T}_{rs} \in \{pred(\mathcal{T}_{ij}) - CLUSTER[q]\}}(tlevel(\mathcal{T}_{rs}) + t_{rs}^{(C)} + t_{rs,ij}^{(R)})\}$ since the transmission time will reduce to 0 for all the task in the same cluster. Then, $\mathcal{T}_{ij}$ will select the one with smallest $tlevel^{(q)}(\mathcal{T}_{ij})$ to join, and update $tlevel(\mathcal{T}_{ij})$ with $tlevel^{(q)}(\mathcal{T}_{ij})$ to get $\mathcal{G}'^{(g+1)}$, then $g = g+1$.

If there does not exist any cluster satisfies the above conditions, then initiate a new cluster, $CLUSTER[p]$, where $CLUSTER[p] = \{\mathcal{T}_{ij}\}$. Finally, $UC = UC - \{\mathcal{T}_{ij}\}$.

Step 4. Repeat **Step 2** and **Step 3** until $UC = \emptyset$.

An example to illustrate the procedure of Task Clustering Algorithm is given in Fig.5(b) and Fig.5(c). Fig.5(b)

presents the beginning of an iteration and there exist three clusters, which are $CLUSTER[1] = \{\mathcal{T}_{11}\}$, $CLUSTER[2] = \{\mathcal{T}_{12}\}$ and $CLUSTER[3] = \{\mathcal{T}_{21}\}$. Next, the priorities of $\mathcal{T}_{13}$ and $\mathcal{T}_{22}$ are compared since their immediate predecessors have been already clustered. According to the definition, $PRIO(\mathcal{T}_{13}) = tlevel(\mathcal{T}_{13}) + blevel(\mathcal{T}_{13}) = 15 + 22.6 = 37.6$, similarly, $PRIO(\mathcal{T}_{22}) = 18.2 + 19 = 37.2$. Thus, $\mathcal{T}_{13}$ is firstly selected for clustering due to $PRIO(\mathcal{T}_{13}) > PRIO(\mathcal{T}_{22})$. Based on Fig.5(b), there exist two candidate clusters, which are $CLUSTER[1]$ and $CLUSTER[3]$, that $\mathcal{T}_{13}$ can join. Then, $tlevel^{(1)}(\mathcal{T}_{13}) = 12$ if $\mathcal{T}_{13}$ selected $CLUSTER[1]$ to join, similarly, $tlevel^{(3)}(\mathcal{T}_{13}) = 15$ if $CLUSTER[3]$ is chosen for containing $\mathcal{T}_{13}$. Therefore $\mathcal{T}_{13}$ will join to $CLUSTER[1]$ as shown in Fig.5(c) since $tlevel^{(1)}(\mathcal{T}_{13})$ is smaller.

5.2.3 ***Phase III:*** *Ordered Task Assigning*

Since each task have to wait for all the intermediate results before starting and it also has the deadline, and thus it implicates two important time points for each task, which are the **Earliest Start Time** (*EST* for short) and the **Latest Finish Time** (*LFT* for short). Such two time points are quite important for task assignment, and their definitions are provided as follows.

Definition 3. For each task $\mathcal{T}_{ij}$ in an ordered task set, *EST*($\mathcal{T}_{ij}$) and *LFT*($\mathcal{T}_{ij}$) are called as its *EST* and *LFT iff* they satisfy:

$$EST(\mathcal{T}_{ij}) = \begin{cases} \max(T^{(R)}_{(k+i),p_{ij}}(\alpha_{ij}), T^{(R)}_{L_{ij},p_{ij}}(\beta_{ij})), \\ \qquad \text{if } \mathcal{T}_{ij} \text{ is the } \textbf{Start Task} \\ \max(\max\limits_{\mathcal{T}_{rs}\in pred(\mathcal{T}_{ij})}(EST(\mathcal{T}_{rs}) + t^{(C)}_{rs} + \\ t^{(R)}_{rs,ij}), \quad T^{(R)}_{(k+i),p_{ij}}(\alpha_{ij}), T^{(R)}_{L_{ij},p_{ij}}(\beta_{ij})), \\ \qquad \text{Otherwise} \end{cases}$$

$$LFT(\mathcal{T}_{ij}) = \begin{cases} T_{ij}, \qquad \text{if } \mathcal{T}_{ij} \text{ is the } \textbf{End Task} \\ \min(\min\limits_{\mathcal{T}_{uv}\in suc(\mathcal{T}_{ij})}(LFT(\mathcal{T}_{uv}) - t^{(R)}_{ij,uv} - \\ t^{(C)}_{uv}), T_{ij}), \qquad \text{Otherwise} \end{cases}$$

where $p_{ij} = l$ on condition that $x_{ijl} = 1$, *i.e.* p_{ij} denotes the processor that handles $\mathcal{T}_{ij}$.

Similarly, given a $CLUSTER[q]$, it also have the Latest Finish Time, denoted by $LFTC[q]$, where $LFTC[q] = \min_{\mathcal{T}_{ij}\in CLUSTER[q]}(LFT(\mathcal{T}_{ij}))$. Obviously, $LFTC[q]$ actually denotes the priority of $CLUSTER[q]$ during assignment.

Based on the above definitions and the clustering result returned in Phase II, the task assignment algorithm contains five steps, where it tries to reduce the energy cost as much as possible on condition that the constraints on resource and deadline are sufficiently considered.

Firstly, let CQ be a queue that contains all the unassigned clusters that are ordered according to their *LFT* since such parameter represents the assignment priority of each cluster. Initially, CQ consists of all the clusters returned in Phase II.

Secondly, select the first cluster in CQ, denoted by $CLUSTER[q]$, we will check each task in it by the following steps.

1) For each $\mathcal{T}_{ij} \in CLUSTER[q]$, if $\exists \mathcal{T}_{rs} \in pred(\mathcal{T}_{ij}) - CLUSTER[q]$, and $\mathcal{T}_{rs}$ has not been assigned to any processors, then remove $\mathcal{T}_{ij}$ from $CLUSTER[q]$, *i.e.* $CLUSTER[q] = CLUSTER[q] - \{\mathcal{T}_{ij}\}$.
2) Repeat the above step until there does not exist any task need to be removed from $CLUSTER[q]$.
3) All the removed tasks become a new cluster, calculate the *LFT* of the new cluster, and insert it in CQ according to its *LFT*.

Thirdly, select the processors, $p_1, p_2, ..., p_r$,for $CLUSTER[q]$ on condition that the following constraints are met.

1) For $\forall \mathcal{T}_{ij} \in CLUSTER[q]$, the deadline constraint of $\mathcal{T}_{ij}$ is met if $CLUSTER[q]$ is assigned to processor p_l $(1 \le l \le r)$
2) $\sum_{\mathcal{T}_{ij}\in CLUSTER[q]} C_{ij} \le max_{p_l}$ for all $1 \le l \le r$.

Based on $\{p_1, p_2, ..., p_r\}$, the following cases need further considered.

1) If $r \ge 1$, *i.e.* there at least exists a processor satisfying the above constraint, then select the one whose energy consumption is minimized for processing $CLUSTER[q]$, denoted by p_q, and assign $CLUSTER[q]$ to it.
2) If $r = 0$, *i.e.* there does not exit any processor satisfying the above constraints, then greedily remove the task from the set $\{\mathcal{T}_{rs} | \mathcal{T}_{rs} \in CLUSTER[q] \wedge suc(\mathcal{T}_{rs}) \cap CLUSTER[q] = \emptyset\}$ based on the resource it occupied.

Repeat the whole step until we find a proper processor for $CLUSTER[q]$ or $CLUSTER[q] = \emptyset$. **If** $CLUSTER[q] \neq \emptyset$, then assign $CLUSTER[q]$ to p_q, and all the removed task become a new cluster and insert to the CQ based on its *LFT*.

Fourthly, remove $CLUSTER[q]$ from the cluster queue CQ, $max_{p_q} = max_{p_q} - \sum_{\mathcal{T}_{ij}\in CLUSTER[q]} C_{ij}$. Meanwhile, let $p_{ij} = p_q$ for $\forall \mathcal{T}_{ij} \in CLUSTER[q]$, that is, x_{ijl} *iff* $l = p_q$ and others equals zero. and $t_{ij} = EST(\mathcal{T}_{ij})$. Update the DAG $\mathcal{G}'$ with new computation time as well as new transmission time between a task and its immediate predecessor for each task in $CLUSTER[q]$.

Finally, repeat the above three steps until $CQ = \emptyset$.

TABLE 2
Energy Cost for Processing Clusters

Processor	$CLUSTER[1]$	$CLUSTER[3]$	$CLUSTER[2]$
MD_1	60	Out of Resource	Out of Resource
MD_2	120	183	Out of Resource
MD_3	195	156	Over Time
MD_4	196	192	Out of Resource
B_1	150	153	Out of Resource
B_2	165	162	Over Time
Cloud	210	210	Over Time

Another example is presented in Fig.5(d) and Table II to explain the procedure of above Task Assignment Algorithm, where Table II is provided the energy cost for each processor to deal with the given cluster. If the **Deadline Constraint** or **Resource Constraint** of given cluster cannot be met by a specific processor, the corresponding grid is marked as "Over Time" or "Out of Resource", respectively. As shown In Fig. 4(d), the head of CQ is $CLUSTER[1]$ since

$LFTC[1] < LFTC[3] < LFTC[2]$, and it is firstly selected and processed. Because an immediate predecessor of $\mathcal{T}_{13}$ has not been assigned, so it is removed from $CLUSTER[1]$ and form a new cluster, denoted by $CLUSTER[4]$. Based on Table II, the rest tasks will be assigned to MD_1 due to the minimal energy cost. Another example, when $CLUSTER[2]$ becomes the head of CQ, we find that there does not exist any processor that can meet both of its **Deadline Constraint** and **Resource Constraint**, therefore, $\mathcal{T}_{23}$ is removed from $CLUSTER[2]$ and become a new cluster since it occupy the largest resource in $CLUSTER[2]$, as shown in Fig.5(f).

5.3 Performance Analysis of *HOTA-C*

According to the above analysis, *HOTA-C* contains three phases, which are initial phase, task clustering phase and task assignment phase. In the second phase, each task only belongs to one cluster based on Step 3 of Task Clustering Algorithm. Furthermore, each cluster is assigned to a single processor by Step 3 of Task Assignment Algorithm in the third phase, therefore, Constraint $C1$ in OTA problem is satisfied. Secondly, the partial order relationships among tasks are fully considered by using the *EST* to determine the star time for each task in the third phase, where *EST* implicates the order information of each task according to its definition, and thus Constraint $C2$ in OTA problem is met. Finally, both of Deadline and Resource Constraints are taken into account by Step 3 of Task Assignment Algorithm in the third phase, so that Constraints $C3$ and $C4$ are also maintained. In summary, the result returned by the above *HOTA-C* Algorithm is a feasible solution that meet all the constraints of OTA problem.

Next, we will analysis the complexity of *HOTA-C* Algorithm. For the convenience of discussion, We use n_T to denote the total number of tasks, that is $n_T = \sum_{i=1}^{n} N_i$, and use d_{max} to denote the maximum in-degree of $\mathcal{G}$. Let δ be the number of generated clusters during the task assignment.

As mentioned above, *HOTA-C* Algorithm contains three phases.

In Phase I, the complexity of computing $\bar{t_{ij}}^{(C)}$ $(\mathcal{T}_{ij} \in V)$ is $O((k+n)n_T)$, and the complexity of computing $\bar{t_{rs,ij}}^{(R)}$ $((\mathcal{T}_{ij}, \mathcal{T}_{rs}) \in E)$ is $O((k+n)^2 n_T d_{max})$.

In Phase II, the complexities of Step 1 and Step 2 are $O(k+n)$ and $O(n_T)$ respectively for determining C_{max}, the *tlevel* and *blevel* for each task. In the Step 3 of Phase II, the candidate clusters that the selected task considers to join is at most d_{max}, so that the complexity is $O(d_{max})$. Finally, Step 2 and Step 3 are repeated n_T times based on Step 4 of Task Clustering Algorithm. Thus, the total complexity of Phase II is $O(\max\{d_{max} n_T, k+n\})$.

In Phase III, the complexity for calculating the *LFTC* of all the clusters is $O(n_T)$ since the *LFT* of each task should be considered, and the complexity for sorting the clusters in the cluster queue, CQ, is $O(\delta_{max} \log(\delta_{max})) \leq O(n_T \log n_T)$ in the first step. Secondly, let $CLUSTER[q]$ be a cluster in CQ, and $|CLUSTER[q]|$ be the size of $CLUSTER[q]$. Then, some tasks from $CLUSTER[q]$ is removed firstly in order to meet all the given constraints, the maximum size of removing tasks is $|CLUSTER[q]| - 1$, so that the complexity is $O(|CLUSTER[q]|) \leq O(n_T)$ since $|CLUSTER[q]| \leq n_T$. Then, we need the transmission time between a specific task and its immediate predecessors to judge whether its deadline is satisfied when the processor is appointed. Therefore, the complexity is $O((k+n)d_{max})$ for each task, and $O((k+n)d_{max}|CLUSTER[q]|) \leq O((k+n)d_{max}n_T)$ for $CLUSTER[q]$. Fourthly, the complexity is $O(\delta_{max}|CLUSTER[q]|) \leq O(d_{max}n_T)$ since updating $\mathcal{G}'$ with new computation time and new transmission time between a task and its immediate predecessor for each task. Finally, the above three step is repeated for each cluster in CQ, therefore, the total complexity of Phase III is $O(\delta_{max}(k+n)d_{max}n_T) \leq O(n_T^2(k+n)d_{max})$.

In summary, the complexity of our *HOTA-C* Algorithm is $O(d_{max}n_T(k+n)\max\{k+n, n_T\})$. Since d_{max} can be regards as the constant comparing with n_T, and thus the complexity of *HOTA-C* Algorithm is $O(n_T(k+n)\max\{k+n, n_T\})$, which equals to $O(n_T^2)$ if the number of tasks are dramatically larger than the number of base stations and mobile devices. Thus, our *HOTA-C* Algorithm is a polynomial algorithm.

6 Performance Evaluation

6.1 Experiment Settings

Similar as [28] [29], we also assume that the energy cost, required CPU cycles and result size is linear to the size of input data in the simulated MEC system. That is, the required CPU cycles and the energy cost for processing such data are λX and $\kappa\lambda X$, the result size can be estimated by ηX when the size of input data is X, where $\kappa = 10^{-27}$ (J/cycle), $\lambda = 330$ (cycle/byte), $\eta = 0.2$ based on [28], [29]. The CPU frequencies of mobile devices varies from 1GHz to 2GHz.

For a based station, its CPU frequency is set to be 4GHz, and the transmission delay between two base stations are 15ms according to [12]. For the cloud, we take Amazon T2.nano as example and set its CPU frequency to be 2.4GHz, the delay between base station and the cloud to be 250ms based on [13].

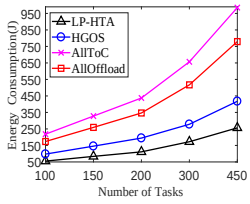
Finally, in the following simulated MEC system, each mobile device connects with the base station by 4G or WiFi randomly, where the download, upload, transmitting and receiving powers of simulated wireless network are summarized in Table III based on [30] and [31].

TABLE 3
Parameters of Wireless Networks

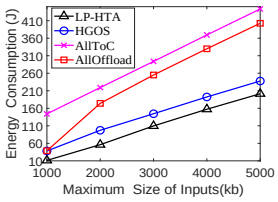
NetWork	Download speed	Upload speed	P^T	P^R
4G	13.76 Mbps	5.85 Mbps	7.32 W	1.6W
Wi-Fi	54.97 Mbps	12.88 Mbps	15.7 W	2.7 W

6.2 Performance of *LP-HTA* Algorithm

In this section, we will investigate the performance of our *LP-HTA* algorithm by comparing with *HGOS* and two baseline algorithms, *i.e. AllToC* and *AllOffload*, where *HGOS* denotes the Heuristic Greedy Offloading Scheme proposed in [9], *AllToC* and *AllOffload* means that all the tasks are offloaded to the cloud and to the base stations and the cloud, respectively. The reasons of choosing the above three methods are as follows. Firstly, we compare *LP-HTA* with *HGOS* [9] since it is the latest and efficient task assignment



(a) Affected by Task Number



(b) Affected by Input Data Size

Fig. 6. Energy Cost of *LP-HTA*.

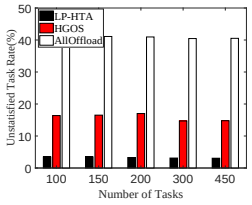
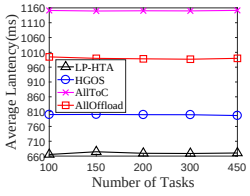
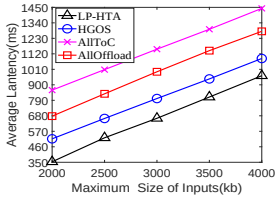


Fig. 7. The Rate of Unsatisfied Task.

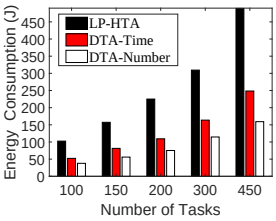


(a) Affected by Task Number

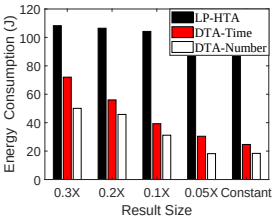


(b) Affected by Input Data Size

Fig. 8. Latency of *LP-HTA*.

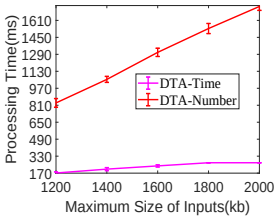


(a) Affected by Task Number

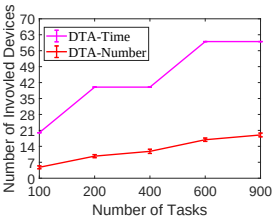


(b) Affected by Size of Task Result

Fig. 9. Energy Cost of *LP-HTA*, *DTA-Workload*, and *DTA-Number*.



(a) Processing Time



(b) Invovled Devices

Fig. 10. The comparison of *DTA-Workload* and *DTA-Number*.

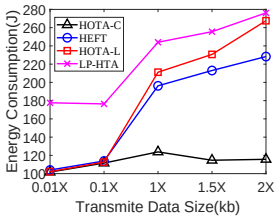


Fig. 11. Affected by Transmission Data Size.

algorithms in a MEC system. Secondly, *AllToC* and *AllOffload* are selection for comparison because they are most classical and representative methods when the computation abilities of the mobile devices are not considered. The size of the external data is set to 0 to 0.5 times the local data.

The first group of experiments investigate the impact of the number of tasks on the energy cost of *LP-HTA*. In the experiments, the energy consumed by *LP-HTA*, *HGOS*, *AllToC* and *AllOffload* were calculated while the number of the tasks in a MEC system increased from 100 to 450, and the maximum input data size for each task is 3000kb. According to the results illustrated in Fig.6(a), the energy consumption of *LP-HTA* is much smaller than that of *AllToC* and *AllOffload* as the computation abilities of the mobile edges, including mobile devices and the base stations are sufficiently used. Furthermore, the energy consumed by our *LP-HTA* is also lower than that consumed by *HGOS*. Since the constraints of each mobile edge is fully considered during task offloading, the task assignment results of our *LP-HTA* is more reasonable and efficient comparing with *HGOS*, and thus much energy will be saved. Finally, we find that the energy consumed by *LP-HTA* increases slowly with the increment of the tasks, which means that our *LP-HTA* algorithm is very suitable to process the computation intensive tasks.

The second group of experiments observe the energy consumption of *LP-HTA* affected by the input data size. In the experiments, the energy cost of *LP-HTA*, *HGOS*,*AllToC* and *AllOffload* were calculated while the maximum input data size of each task increased form 1000 kb to 5000 kb, and the number of total tasks is 100. The results presented in Fig.6(b) indicate that the energy consumption of *LP-HTA* is also the smallest even when the amount of data required to be processed increases, which means that our *LP-HTA* is also suitable to deal with the data-intensive tasks.

The third group of experiments is to investigate the unsatisfied task rate of all the task assignment algorithms. The unsatisfied task rate is denoted by the proportion of the tasks whose delay constraints can not be met among all the tasks. Since the unsatisfied task rate of *AllToC* is quite high due to the large latencies of transmitting all the task to the remote cloud, we only compare *HGOS*, *AllOffload* and our *LP-HTA* in the experiments. As shows in Fig.7, the unsatisfied task rate of the three algorithms were calculated when the number of tasks increased form 100 to 450. The results shows that the unsatisfied task rate of our *LP-HTA* is quite small comparing with *HGOS* and *AllOffload*. Since we take the delay constraint of each task into account during task assignment, only a small portion of task are unsatisfied. Therefore, our *LP-HTA* algorithms are more efficient and effective to process the tasks that has strict delay constraints. Furthermore, although the energy cost of *HGOS* are close to *LP-HTA* based on Fig.6, it has quite large unsatisfied task rate, so that the performance of *HGOS* is not good enough comparing with *LP-HTA* as large amount of deadline constraints cannot be met.

The fourth group of experiments is to investigate the impact of the amount of tasks on the latency of *LP-HTA*. In the experiments, the average latency of *LP-HTA*, *HGOS*, *AllToC* and *AllOffload* were calculated while the number of tasks grows from 100 to 450, and the maximum size of input data is 3000kb. From results in Fig.8(a), we can see that the average latency of *LP-HTA* is extremely small comparing with *AllToC* and *AllOffload* since the mobile devices are fully used and some of task are processed immediately by *LP-HTA*. Moreover, the average latency of *LP-HTA* is also much lower than that of *HGOS* as *LP-HTA* considers both of task deadline constraints and the limited computing resources of each mobile edge sufficiently during task assignment. Thus,LP-HTA is efficient to process the latency-intensive tasks in MEC systems.

The fifth group of experiments is used to analyze the relationship between the latency of *LP-HTA* and the size of input data. The average latency of *LP-HTA*, *HGOS*, *AllToC*

and *AllOffload* were calculated while the maximum size of the input data for each task varies from 1000kb to 5000kb, and the number of total tasks is 100. Based on the results in Fig.8(b), the average latency of *LP-HTA* is still the smallest comparing with *HGOS*, *AllToC* and *AllOffload*. However, the advantage of *LP-HTA* on latency is not so much obvious comparing with *HGOS*, which is because more tasks will exceed the processing abilities of mobile devices with the increment of input data size for both *LP-HTA* and *HGOS*, so that they are offloaded to base station or to the cloud, and the latency will enlarge accordingly. In spite of that, the *LP-HTA* still utilize the processing abilities of the mobile edges, including mobile devices and base stations, as much as possible. Meanwhile, the *LP-HTA* also guarantees that much more deadline constraints of the tasks are satisfied even when the average latency is enlarged according to the results in Fig.7. Such results also verified that our *LP-HTA* is efficient and effective to deal with the data-intensive tasks.

6.3 Performance of *DTA*

The section will evaluate of the performance of Divisible Task Assignment algorithms, and we use *DTA-Time* and *DTA-Number* to stand for the algorithms in Sections 4(a) and 4(b), respectively. In the following experiments, the comparisons among *DTA-Time*, *DTA-Number* and *LP-HTA* are considered since *LP-HTA* has the best performance according to the analysis in the above section

The first group of experiments compares the energy cost by *DTA-Time*,*DTA-Number* and *LP-HTA* for processing the tasks with different amount. The energy cost of three methods were computed while the number of tasks increased form 100 to 450, the maximum size of input data was 3000kb for each task, and the ratio of the result size to the amount of input data was 0.2. Fig.9(a) shows that the energy cost of *DTA-Time* and *DTA-Number* are quite small especially when the amount of tasks increases. Since more raw data are avoided to transmit by *DTA-Time* and *DTA-Number* when the amount of tasks increases, lots of energy will be saved.

The second group of experiments observes the energy cost of the above three algorithms while the result size is varying. In the experiments, the result size is set to be $0.3X$, $0.2X$, $0.1X$, $0.05X$ and constant, where X is the size of input data. The number of total tasks in a MEC system was 100, and the maximum size of input data was 3000kb for each task. Fig.9(a) presents the energy cost of three algorithm for different result size. It shows that *DTA-Time* and *DTA-Number* consume extremely small energy when the result size decreases. Considering that the sizes of final result and partial results are alway much smaller comparing with the raw data, *DTA-Time* and *DTA-Number* will achieve high performance for processing many kinds of tasks.

The last two groups of experiments are going to compare *DTA-Time* and *DTA-Number* separately. In Fig.10(a), the processing time of *DTA-Time* and *DTA-Number* were calculated while the maximum size of input data increased from 1200kb to 2000kb, and the number of total tasks was 200. In Fig.10(b), the numbers of involved mobile devices for *DTA-Time* and *DTA-Number* were counted while the number of total tasks varied form 100 to 900, and the maximum size of input data was 2000kb for each task. The results in Fig.10(a) and Fig.10(b) show that the processing time of *DTA-Time* is much smaller since it tries to reduce the maximum parallel processing time as much as possible. On the other hand, smaller amount of mobile devices are involved by applying *DTA-Number* algorithm, so that the energy of majority mobile devices is saved, meanwhile, the communication cost is also reduced since fewer intermediate results are transmitted in a MEC system. In summary, *DTA-Time* and *DTA-Number* are suitable for different situations, where the users can select one of them flexibly according to the applications.

6.4 Performance of *HOTA-C*

The following section will evaluate the performance of our *HOTA-C* algorithm by comparing with three modified algorithms, which are *LP-HTA*, *HEFT* and *HOTA-L*. The *LP-HTA* is the algorithm proposed in section 3 in our paper, which is very efficient for processing the unordered holistic tasks in MEC systems according to Section 6.2. The *HEFT* [25] is a classical algorithm for assigning the ordered tasks in heterogeneous systems based on DAG, we modified it by considering Deadline and Resource Constraints. *HOTA-L* is the task assignment algorithm based on the topological sorting which is also widely used by many existing ordered task assignment algorithm [32]. Most parameters are the same with the ones given in Section 6.1 except that the range of local data size for each task is $[2000kb, 2500kb]$. Meanwhile, the maximum degree of given DAG, $\mathcal{G}$, is 5.

The first group of experiments investigates the impact of the intermediate result size on the energy consumption of the algorithms. In the experiment, the total number of tasks in a MEC system is set to be 32. The energy cost by four algorithms is calculated while the intermediate result size is varying from $0.01X$ to $2X$, where X denotes the size of the local data. The experimental results are presented in Fig.11. It shows that the energy consumed by *LP-HTA*, *HEFT* and *HOTA-L* increase rapidly with the growth of the intermediate result size, however, the energy consumption of *HOTA-C* algorithm is quite small since we use the cluster structure to reduce the communication cost as much as possible. Meanwhile, the energy cost by *LP-HTA* is largest comparing with the other three algorithms because it is the only one that is not designed for ordered task assignment.

The second group of experiments observes the influence of the number of tasks on the energy consumption of the algorithms. In the experiment, the energy cost by *HOTA-C*, *LP-HTA*, *HEFT* and *HOTA-L* were calculated while the number of the tasks in a MEC system increased from 16 to 80. Without loss of generality, we randomly set the size of intermediate results. The results are illustrated in Fig.12 shows that the energy consumption of *HOTA-C* is much lower than that of the other three algorithms because *HOTA-C* make full use of the resources owned by edge devices, including mobile devices and the base stations. Meanwhile, it also sufficiently consider the energy consumption caused by transmission as well. Therefore, our *HOTA-C* algorithm is also very efficient even when the load of the tasks increases.

The third group of experiments is to investigate the unsatisfied task rate of four selected algorithms. The unsatisfied task rate is denoted by the proportion of the tasks

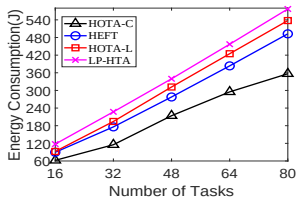


Fig. 12. Affected by Number of Tasks.

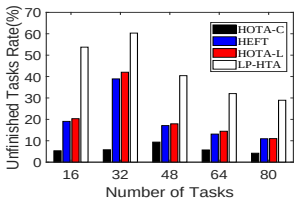


Fig. 13. Unfinished Tasks Rate.

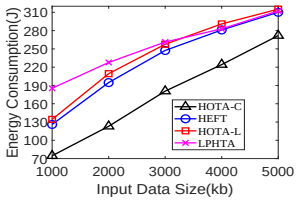


Fig. 14. Affected by Input Data Size.

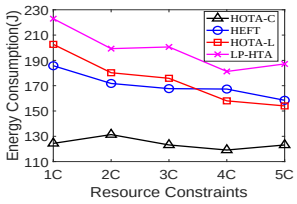


Fig. 15. Affected by Resource Constraint.

whose constraints can not be met among all the tasks. As shows in Fig.13, the unsatisfied task rate of the four algorithms was calculated when the number of tasks increased from 16 to 80. The result shows that the unsatisfied task rate of our *HOTA-C* is quite small comparing with *HOTA-L*,*HEFT* and *LP-HTA*. Since *HOTA-C* take advantage of the cluster structure as much as possible, the transmission delay is significantly reduced. Thus, only a small portion of tasks are unsatisfied, that is, our *HOTA-C* algorithm can guarantee the majority task to be processed on time.

The fourth group of experiments is to evaluate the energy consumption of *HOTA-C* affected by the local data size. In the experiment, the total number of tasks in a MEC system is set to be 32 and the energy consumed by *HOTA-C*, *HOTA-L*, *HEFT* and *LP-HTA* were calculated while the input data size of each task increased form 1000 kb to 5000 kb. According to the results illustrated in Fig.14, the energy consumption of *HOTA-C* is also the smallest which means that our *HOTA-C* is also suitable for scenarios with large amount of computation.

The fifth group of experiments is to investigate the energy consumption of *HOTA-C* affected by resource constraints. In this experiment, the total number of tasks of MEC system is 32 and the resource of each processor is varying from 1C to 5C, where C is the default resource size owned by processor. The results presented in Fig.15 indicate that the energy consumption of *HOTA-C* is extremely smaller than the other three algorithms. Such result implicates that our *HOTA-C* become more efficient when the resource of the mobile edges increases. Because our *HOTA-C* algorithm sufficiently uses to resource of edge devices, it is very suitable for the MEC systems with abundant resource in base stations and mobile devices.

7 Related Work

7.1 Related work about Unordered Task Assignment

The early ones that studied the task assignment in a MEC system mainly focus on serving a single user. To determine the offloading strategies of the tasks proposed by single users, a Binary offloading algorithms is given in [3]. To take advantage of parallel computing, a partial offloading problem were studied in [33], and an approximation algorithm with $(1+\epsilon)$ ratio bound was given to solve it. In [34], a variable-substitution technique was provided to jointly optimize the offloading ratio, transmission power and CPU-cycle frequency. These works has high performance to deal with the tasks raised by a single user, however, the situation considered by them seems a little simple. Meanwhile, they did not consider the data distribution during determine the offloading strategies.

Considering that a MEC system always serve for multiple users, [5] has formulated the computation offloading problem of multiple users via a single wireless access point as a computation offloading game, and proposed a decentralized mechanism that can achieve the Nash equilibrium to solve it. In [35], the author try to optimize the uplink and downlink scheduling using queuing theory. In [36], a decomposition algorithm was given to control the computation offloading selection, clock frequency control and transmission power allocation iteratively. The above algorithms are efficient to process the tasks raised by multiple users, however, since they were designed for a MEC system with a single base station, the cooperations among different base stations were not considered sufficiently. Furthermore, these algorithms ignored the data distribution either during task assignment. Besides, [8] introduced the centralized and distributed Greedy Maximal Scheduling algorithms to solve the computation offloading problem for multiple users with multiple tasks. However, these algorithms did not consider the data sharing during the task assignment, and ignored the delay constraint of each task as well, so that they are not suitable to solve the problem discussed in this paper.

To enhance the cooperations among different base stations, a new cooperation scheme among base stations was given in [37], the author tried to reduce the response latency through caching the results of the popular computation tasks. [38] proposed a game-theoretic algorithm to maximize revenues from all applications. All above works fully utilized the cooperations among base stations. However, these algorithms ignored the distribution of input data as well, and they are not suitable to the Data-Shared MEC systems. [39] provided a payment-based incentive mechanism. Such mechanism supports sharing computation resources among base stations and the authors also established a social trust network to manage the security risks among them, however, it still has the above problems that we pointed out. Moreover, the constraints on processing ability of each base station are not reasonable enough since they only consider the task arrival rate and didn't take the data and resources occupied by each task into account.

More recently, the architecture of a MEC system which involves the three levels and serves for multiple users are sufficiently investigated by [9] and [10]. In [9], a heuristic greedy offloading scheme was given for ultra dense networks. In [10], a distributed computation offloading algorithm that can achieve the Nash equilibrium was provided. Both of these works are efficient for a MEC system, however, they did not considered data distribution and the delay constraints of tasks during task assignment, thus they are not able to assign the tasks in a Data-Shared MEC system as well. Besides, the processing abilities of the Cloud are not

fully utilized, either.

7.2 Related work about Ordered Task Assignment

To the best of our knowledge, we are the first one to study the ordered task assignment algorithm in MEC systems. Therefore, in the rest of the section, we will summarize the research results of the ordered task assignment approaches in other distributed systems.

Generally, the directed acyclic graph (DAG) is widely used to denote the task dependencies in the distributed systems. Based on such structures, many heuristic and random algorithms have been proposed, which can also be divided into four categories.

First, a group of algorithms is based on the priority list, such as HEFT [25] and HCPT [40]. Such algorithms have high performance and low complexity. However, they assume that the computation and transmission time can be obtained in advance before scheduling, meanwhile, they did not consider the resource limitation of different processors and their aim is not to minimize the energy cost as well. Therefore, these algorithms are not suitable to solve our OTA problem in a MEC system.

Second, the grouping based heuristic algorithms are studied in [27] [41] for scheduling the ordered tasks in a distributed system with an unlimited number of processors, which is not practical for a MEC system. [42] studied the task scheduling problem in the system with a number of heterogeneous processors, where the processing speed and bandwidth of different processors are different. The given algorithm reduces the scheduling length of all the ordered task as much as possible. However, it did not consider the resource occupied by each task and the limitation of each processor either. Meanwhile, its optimal aim is also different from our OTA problem.

In order to further avoid the transmission among tasks, a heuristic method of fully using the idle time of each processor had been proposed by [43]. It allowed the task duplication, that is, one task may be carried out many times by different processors if they have idle time. Such method is efficient for the homogeneous systems, which is quite different from MEC systems discussed in our paper. Meanwhile, the idle time of different processors are not available ahead for a MEC system, so that it is not suitable to solve our OTA problem as well.

Furthermore, a group of random-search-based techniques have been also studied by researchers for ordered task scheduling, such as [44] and [45]. Most of them were designed according to the genetic algorithm, so that the complexities is extremely high since they will iterate multiple times. Therefore, such methods are not suitable for our OTA problem due to their high complexities as well as the lack of consideration on resource limitation for each processor.

Finally, the related work discussed in above section also cannot be applied to the OTA problem since the task considered by them are unordered.

8 Conclusion

This paper studies the task assignment problem in a Data-Share MEC system. We firstly distinguish the computation tasks as the holistic tasks and the divisible tasks, and then we consider the order dependencies among the holistic tasks. For the unordered holistic tasks, the HTA problem is proposed and proved to be NP-complete. Finally, a linear programming based approximation algorithm is proposed for it. For the unordered divisible tasks, the key problem is to rearrange the tasks according to the data distribution. Two approximation algorithms with $O(n)$ complexities are provided to solve it for different optimization goals. For the ordered tasks, heuristics ordered tasks assignment based on clustering is proposed. Both of theoretical analysis and experiments show that all proposed algorithms achieve high performance.

Acknowledgments

This work is partly supported by the National Natural Science Foundation of China under Grant No. 61972114, the National Natural Science Foundation of Heilongjiang Province under Grant No. YQ2019F007.

References

[1] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing - a key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.
[2] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in *International Conference on Intelligent Systems and Control*, 2016.
[3] W. Zhang, Y. Wen, K. Guan, K. Dan, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE TWC*, vol. 12, no. 9, pp. 4569–4581, 2013.
[4] K. Kumar, J. Liu, Y. H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Networks & Applications*, vol. 18, no. 1, pp. 129–140, 2013.
[5] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *Parallel & Distributed Systems IEEE Transactions on*, vol. 26, no. 4, pp. 974–983, 2014.
[6] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2016.
[7] T. Q. Dinh, Q. D. Tang, Jianhua La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE TCOM*, vol. 65, no. 8, pp. 3571–3584, 2017.
[8] W. Chen, D. Wang, and K. Li, "Multi-user multi-task computation offloading in green mobile edge cloud computing," *IEEE Transactions on Services Computing*, 2018.
[9] H. Guo, J. Liu, and J. Zhang, "Computation offloading for multi-access mobile edge computing in ultra-dense networks," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 14–19, 2018.
[10] L. Tang and S. He, "Multi-user computation offloading in mobile edge computing: A behavioral perspective," *IEEE Network*, vol. 32, no. 1, pp. 48–53, 2018.
[11] T. D. Burd and R. W. Brodersen, *Processor design for portable systems*. Kluwer Academic Publishers, 1996.
[12] J. Oueis, E. Calvanese-Strinati, A. De Domenico, and S. Barbarossa, "On the impact of backhaul network on distributed cloud computing," in *Wireless Communications and NETWORKING Conference Workshops*, 2014, pp. 12–17.
[13] "Amazon Cloud." [Online]. Available: http://www.cloudping.info/
[14] N. Karmarkar, "A new polynomial-time algorithm for linear programming," *Combinatorica*, vol. 4, no. 4, pp. 373–395, 1984.
[15] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Annual Cryptology Conference*, 2010, pp. 465–482.
[16] Y. Huang, X. Song, Y. Fan, Y. Yang, and X. Li, "Fair caching algorithms for peer data sharing in pervasive edge computing environments," in *ICDCS*, 2017.
[17] V. V. Vazirani, *Approximation algorithms*. Springer Science & Business Media, 2013.
[18] U. Feige, *A threshold of ln n for approximating set cover*. ACM, 1998.

[19] B. Kim, D. Kim, K. Kim, and K. Yi, "High-level automated driving on complex urban roads with enhanced environment representation," in *2015 15th International Conference on Control, Automation and Systems (ICCAS)*. IEEE, 2015, pp. 516–521.
[20] L. Liu, H. Li, and M. Gruteser, "Edge assisted real-time object detection for mobile augmented reality," 05 2019, pp. 1–16.
[21] P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan, "Mixed-integer nonlinear optimization," *Acta Numerica*, vol. 22, pp. 1–131, 2013.
[22] S. Burer and A. N. Letchford, "Non-convex mixed-integer nonlinear programming: A survey," *Surveys in Operations Research and Management Science*, vol. 17, no. 2, pp. 97–106.
[23] Z. Abrams and J. Liu, "Greedy is good: On service tree placement for in-network stream processing," in *ICDCS*, 2006, pp. 72–72.
[24] H. Arabnejad and J. G. Barbosa, "List scheduling algorithm for heterogeneous systems by an optimistic cost table," *IEEE Transactions on Parallel & Distributed Systems*, vol. 25, no. 3, pp. 682–694.
[25] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel & Distributed Systems*, vol. 13, no. 3, pp. 260–274, 2002.
[26] G. Xie, R. Li, and K. Li, "Heterogeneity-driven end-to-end synchronized scheduling for precedence constrained tasks and messages on networked embedded systems," *Journal of Parallel and Distributed Computing*, vol. 83, pp. 1–12, 2015.
[27] T. Yang and A. Gerasoulis, "Dsc: scheduling parallel tasks on an unbounded number of processors," *IEEE Transactions on Parallel & Distributed Systems*, vol. 5, no. 9, pp. 951–967, 2002.
[28] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Usenix Conference on Hot Topics in Cloud Computing*, 2010, pp. 4–4.
[29] O. Munoz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 10, pp. 4738–4755, 2015.
[30] A. E. Eshratifar and M. Pedram, "Energy and performance efficient computation offloading for deep neural networks in a mobile cloud computing environment," in *Proceedings of the 2018 on Great Lakes Symposium on VLSI*, 2018, pp. 111–116.
[31] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4g lte networks," in *ACM Mobisys*, 2012, pp. 225–238.
[32] A. EL-NATTAT, N. A. El-Bahnasawy, and A. EL-SAYED, "Enhanced leveled dag prioritized task scheduling algorithm in distributed computing system," in *The International Conference on Electrical Engineering*, vol. 10, no. 10th International Conference on Electrical Engineering ICEENG 2016. Military Technical College, 2016, pp. 1–13.
[33] Y. H. Kao, B. Krishnamachari, M. R. Ra, and B. Fan, "Hermes: Latency optimal task assignment for resource-constrained mobile computing," in *IEEE Conference on Computer Communications*, 2015, pp. 1894–1902.
[34] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE TCOM*, vol. 64, no. 10, pp. 4268–4282, 2016.
[35] M. Molina, O. Munoz, A. Pascual-Iserte, and J. Vidal, "Joint scheduling of communication and computation resources in multiuser wireless application offloading," in *IEEE International Symposium on Personal, Indoor, and Mobile Radio Communication*, 2015, pp. 1093–1098.
[36] S. Guo, B. Xiao, Y. Yang, and Y. Yang, "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing," in *INFOCOM*, 2016, pp. 1–9.
[37] M. S. Elbamby, M. Bennis, and W. Saad, "Proactive edge computing in latency-constrained fog networks," in *European Conference on Networks and Communications*, 2017, pp. 1–6.
[38] R. Yu, J. Ding, S. Maharjan, S. Gjessing, Y. Zhang, and D. H. K. Tsang, "Decentralized and optimal resource cooperation in geo-distributed mobile cloud computing," *IEEE Transactions on Emerging Topics in Computing*, vol. PP, no. 99, pp. 1–1, 2018.
[39] L. Chen and J. Xu, "Socially trusted collaborative edge computing in ultra dense networks," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, 2017, p. 9.
[40] A. A. Nasr, N. A. Elbahnasawy, A. Elsayed, A. A. Nasr, N. A. Elbahnasawy, and A. Elsayed, "Task scheduling optimization in heterogeneous distributed systems," vol. 107, no. 4, pp. 05–12, 2014.
[41] Y. K. Kwok and I. Ahmad, "Dynamic critical-path scheduling: an effective technique for allocating task graphs to multiprocessors," *IEEE Transactions on Parallel & Distributed Systems*, vol. 7, no. 5, pp. 506–521.
[42] H. Kanemitsu, M. Hanada, and H. Nakazato, "Clustering-based task scheduling in a large number of heterogeneous processors," *IEEE Transactions on Parallel & Distributed Systems*, vol. 27, no. 11, pp. 3144–3157, 2016.
[43] I. Ahmad and Y. kwong Kwok, "On exploiting task duplication in parallel program scheduling," *IEEE Transactions on Parallel & Distributed Systems*, vol. 9, no. 9, pp. 872–892.
[44] E. S. H. Hou, N. Ansari, and H. Ren, "Genetic algorithm for multiprocessor scheduling," *IEEE Trans Parallel & Distributed Systems*, vol. 5, no. 2, pp. 113–120, 1994.
[45] Y. Xu, K. Li, J. Hu, and K. Li, "A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues," *Information Sciences*, vol. 270, pp. 255–287, 2014.

Siyao Cheng is an Associate Professor in the School of Computer Science and Technology at Harbin Institute of Technology. She received the BS, Master and PhD degrees in computer science from Harbin Institute of Technology. Her research interests include big sensory data management, wireless sensor networks and Edge Computing.

Jiayan Huang is a Master Student in the School of Computer Science and Technology at Harbin Institute of Technology. She received the BS, Master and PhD degrees in computer science from Harbin Institute of Technology. Her research interests include Network Scheduling and Edge Computing.

Zhenyue Chen is a Master in the School of Computer Science and Technology at Harbin Institute of Technology. He received the BS, Master degrees in computer science from Harbin Institute of Technology. His research interests include wireless sensor networks and Edge Computing.

Jie Liu is a Chair Professor at Harbin Institute of Technology (HIT), China and the Dean of its AI Research Institute. He is an IEEE Fellow and an ACM Distinguished Scientist. Before joining HIT, he spent 18 years at Xerox PARC, Microsoft Research and Microsoft product teams. As a Principal Research Manager at MSR, he led the Sensing and Energy Research Group (SERG). In MSR-NExT and product groups, he incubated smart retail solutions, which became part of Microsoft Business AI offering. His research interests root in understanding and managing the physical properties of computing. He also has chaired a number of top-tier conferences in sensing and pervasive computing, and was Associate Editors for several top-tier journals.

Jianzhong Li is a Professor in the School of Computer Science and Technology at Harbin Institute of Technology. He was a visiting scholar at University of California, Berkeley, a staff scientist in the Information Research Group at the Lawrence Berkeley National Laboratory, and a visiting professor at University of Minnesota. His research interests include data management systems, sensor networks and data intensive computing.

Summary of Changes

This paper is an extended version of the paper "Task Assignment Algorithms in Data Shared Mobile Edge Computing Systems", published in ICDCS 2019. Comparing with the conference paper (10 pages), this submitted paper includes 18 pages, and the total ratio of expansion is more than 60%, including the newly added part (more than 50%) and the modified part (about 10%).

The detail information of the newly added part is as follows.

1. We adjust the optimal goal in Section 4.1. To achieve smaller processing time, the optimal goal in Section 4.1 is to minimize the parallel processing time. Therefore, new algorithm is proposed. The correctness and ratio bound of the algorithm is analyzed. Such content is shown in Pages 6 and 7 in the new manuscript.

2. Considering that there also exist many ordered tasks in an MEC system, we discuss the method to deal with them in Section 5 in the new manuscript. Unlike the unordered tasks, a new structure is needed to describe the order dependencies among tasks, so that the system models, including the computation and transmission models, are required to be updated. Such content is summarized as follow, which are also presented in Section 5.1 in the new manuscript.

   (a) A directed acyclic graph (DAG), denoted by $\mathcal{G}$, is added to describe the order relationship among tasks. Meanwhile, $pred(\mathcal{T}_{ij})$ and $suc(\mathcal{T}_{ij})$ is provided to define the immediate predecessor and successor sets of $\mathcal{T}_{ij}$ in $\mathcal{G}$;

   (b) New parameters, such as $\gamma_{rs,ij}$ and $\eta_{ij}(=\sum_{\mathcal{T}_{rs}\in pred\mathcal{T}_{ij}}\gamma_{rs,ij})$, are provided to denoted the size of the intermediate result for each $1 \le i \le n,\ 1 \le j \le N_i$. New computation model is proposed. Such model not only consider the sizes of local and external data, but also takes the sizes of intermediate results into account as well.

   (c) Since the intermediate results can distribute in the cloud, base stations, and mobile devices. Thus, more complicated cases should be regarded for determining the transmission time and cost. In the new manuscript, we totally discuss four cases, and detailed formulas to determine the transmission time and cost are presented in Page 8 and Page 9.

3. Based on the above modification, the problem of Ordered Task Assignment (OTA) is proposed for MEC systems in the new manuscript. We prove that such problem is NP-hard at the end of Section 5.1

4. To solve the OTA problem approximately, the heuristics algorithm is studied in Section 5.2, which is shown from Page 10 to Page 13. We totally use three Phases to construct such algorithm.

   (a) **Phase I.** To show the time information of different tasks, we construct a new DAG, denoted by $\mathcal{G}'$ according to $\mathcal{G}$, and use the average computation and transmission time for initialization;

(b) **Phase II.** To cluster the tasks, the *tlevel* and *blevel* for each task are firstly defined, and thus the priority of $\mathcal{T}_{ij}$ during clustering can be calculate by $PRIO(\mathcal{T}_{ij}) = tlevel(\mathcal{T}_{ij}) + blevel(\mathcal{T}_{ij})$. Finally, the detailed steps of Task Clustering Algorithm is presented by considering the priority of each task and order relationship among different tasks in Section 5.2.2.

(c) **Phase III.** To meet the deadline constraint and appoint the proper time for each task during scheduling, we first proposed two important time points, which are **Earliest Start Time** (EST) and **Latest Finish Time** (LFT) for each task. Based on EST, LFT and the task clusters returned in Phase II, the Task Assignment Algorithm based on Clusters is provided in Section 5.2.3 in the new manuscript. We also explain each step of such algorithm in detail in the new manuscript.

5. Meanwhile, an example is given in TABLE 1, Fig.4, Fig.5 and TABLE 2 to explain the procedure of each phase in the new manuscript.

6. The pseudocodes of Heuristics Ordered Tasks Assignment Algorithm ($HOTA$-C)is provided in Page 10 in the new manuscript.

7. Finally, the detailed analysis of $HOTA$-C algorithm, including the feasibility of its result and its complexity, is provided in Section 5.3 (Page 13) in the new manuscript.

8. Several groups of new experimental results are provided in the new manuscript to show the performance of $HOTA$-C algorithm. We compare our $HOTA$-C with $HOTA$-L,LP-HTA, and $HEFT$, which are representative works for task assignment in distributed system. New experimental result are presented in Section 6.4 (From Page 15 to Page 16) in the new manuscript, which are also summarize as follows.

   (a) Fig.11 is used to investigate the impact of the intermediate result size on the energy consumption of four algorithms, the results in it show that the energy consumed by our $HOTA$-C algorithm is quite small since it takes the advantages of clusters as much as possible.

   (b) Fig.12 is utilized to observe the influence of the number of tasks on the energy consumption of the four algorithms. The figure shows that the energy consumption of $HOTA$-C is much lower than that of the other three algorithms so that it is also very efficient even when the task load increases.

   (c) Fig.13 investigates the unsatisfied task rate of four selected algorithms. According to it, the unsatisfied task rate of our $HOTA$-C is quite small comparing with other three algorithm since the advantage of cluster structure is fully taken and the transmission delay is significantly reduced.

   (d) Fig.14 evaluates the energy consumption of $HOTA$-C affected by the local data size. According to the result, the energy consumption of $HOTA$-C is also the smallest which means that our $HOTA$-C is also suitable for scenarios with a large amount of computation.

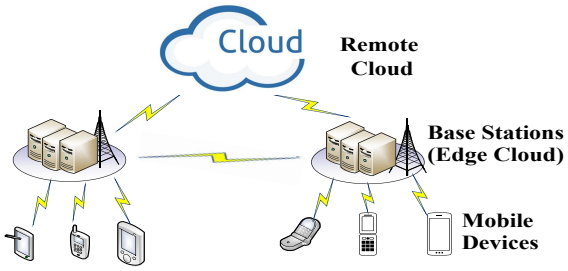


Figure 1: The three levels of a MEC system

(e) Fig.15 is used to investigate the impact of resource constraints on the energy consumption of four algorithms. The result presented in it indicates that the energy consumption of *HOTA-C* is extremely smaller than the other three algorithms because our *HOTA-C* algorithm sufficiently uses to the resource of edge devices.

The detail information of modified part is as follows.

1. We revised the abstract and introduction section, more detailed explanation is added. Meanwhile, the architecture of a MEC system is slightly modified, which is presented in Fig.1.

2. The performance of two Divisible Task Assignment algorithms are re-evaluated since a new algorithm, *i.e. DTA-Time*, is proposed. Therefore, Fig.9(a), Fig.9(b), Fig.10(a) and Fig.10(b) in Page 14 are update, and the analysis in Section 6.3 is also modified in new manuscript.

3. The section of Related Works is modified. Some works studying the ordered task scheduling problem in distributed systems are added and analyzed in Section 7.2 in the new manuscript.

4. Some contents in the conference paper are divided into multiple paragraphs to increase clarity.

5. Minor modifications about the words and grammar.

In summary, the newly added and modified contents are more than 60%.

Sincerely,

Siyao Cheng & Jiayan Huang & Zhenyue Chen & Jie Liu & Jianzhong Li
School of Computer Science
Harbin Institute of Technology

2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)

Task Assignment Algorithms in Data Shared Mobile Edge Computing Systems

Siyao Cheng, Zhenyue Chen, Jianzhong Li, Hong Gao
School of Computer Science and Tech, Harbin Institute of Technology
Emails: csy@hit.edu.cn, chenzyhit@126.com, lijzh@hit.edu.cn, honggao@hit.edu.cn

***Abstract*—The appearance of the Mobile Edge Computing (MEC) successfully solves the bottlenecks of traditional Cloud based networks as computation ability of each mobile edge is sufficiently utilized. Since the mobile edges, including mobile devices and base stations, have certain data processing abilities, it is not necessary to offload all the computation tasks to the remote could for handling. Therefore, it is quite important to decide the optimal task assignment in a MEC system, and a series of algorithms have been proposed to solve it. However, the existing algorithms ignored the data distribution during task assignment, so that the applied ranges of these algorithms are quite limit. Considering the data sharing is very important and common in a MEC system, this paper studies the task assignment algorithm in Data Shared Mobile Edge Computing Systems, and three algorithms are proposed to deal with holistic tasks and divisible tasks, respectively. The theoretical analysis on the hardness of the problem, the correctness, complexities and ratio bounds of the algorithms are also provided. Finally, the extensive experiment results were carried out. Both of the theoretical analysis and experiment results show that our algorithms have high performance in terms of latency and energy consumption.**

I. Introduction

With the development of IoT techniques and widely expansion of mobile devices [1] [2] [3], the applications of current wireless networks become more complicated, such as the intelligent manufacture, the driverless vehicles, smart health care.*etc*. Meanwhile, the QoS(Quality of Service) requirements desired by users, including the response delay, the computation accuracy, the congestion control, and the energy conservation *.etc*, become much stricter than before. Obviously, such requirements cannot be satisfied well in traditional Cloud based networks, where all the data need to be transmitted to the Cloud for processing. Furthermore, the data sampled by current IoT devices manifest an explosive growth, so that it is also not practical to transmit all the data to the Cloud.

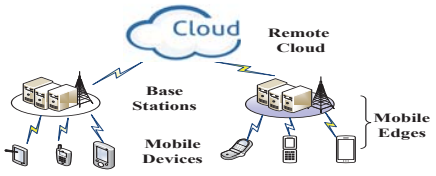


Fig. 1: The three levels of a MEC system

Fortunately, the appearance of the Mobile Edge Computing (MEC) provides a feasible way to solve the above bottlenecks. In a MEC system, the computation abilities of each mobile device and based station are sufficiently utilized, and thus it dramatically decreases the response delay as well as the transmission burden of the network. Similar to many previous studies [4] [5], the architecture of a MEC system considered in our paper have three levels as shown in Fig.1. The first level consists of a number of mobile devices, such as mobile phones, smart sensors *.etc*. These devices are mainly responsible for sampling data and communicating with users. The second level of a MEC system contains a series of base stations, in which a small scale could can be deployed. The base station and the surrounding mobile devices are connected by radio access networks, and they forms the mobile edges for a MEC system. The last level is usually to be the remote cloud which own the richer resources and have higher data processing abilities comparing with the mobile edges, which includes the mobile devices in first level and base stations in second level. However, the response latency is quite long and the energy cost is very high if all computation task are offloaded to the cloud for processing.

Since each mobile edge has certain data processing abilities, it is also not necessary to offload all the tasks to the remote could for handling. Based on such motivation, a series of task assignment algorithms have been proposed in MEC systems. The early ones investigate the problem of offloading the computation tasks for a single user in MEC systems. To solve it, [6] and [7] proposed the Binary offloading algorithm to optimize the latency and energy cost. These works firstly utilized the processing abilities of the mobile edges to improve the performance of the whole network. However, the situation discussed by them seems a little simple. Considering that a MEC system always serves multiple users, [8] has formulated the computation offloading problem of multiple users via a single base station as a computation offloading game, and design a distributed computation offloading mechanism that can achieve the Nash equilibrium to solve it. In [9], the authors have explored a distributed task offloading algorithm for multiple users in multi-channel wireless networks. The above two methods are more practical comparing with the early ones, however, the cooperations among different base stations are not considered. Due to such reason, a new framework that allow a mobile device to offload the tasks to multiple base stations was studied by [10], and semidefinite relaxation-based algorithms were provided to determine the task assignment. However, the situation of serving multiple users has not been addressed sufficiently. More Recently, a series of new task offloading algorithms that serve for multiple

2575-8411/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDCS.2019.00103

users with multiple tasks in a MEC system containing three levels were studied by [11], [12] and [13]. Such algorithms make full use of the MEC systems, however, rare of them take the data distribution and task deadline constraint into account during the task assignment.

As the necessary input, the data required by a computation task may be distributed in different mobile devices or even in different clusters. For example, in a intelligent traffic monitoring system, a user want to know the average flow rate of vehicles in the whole city, while the data sampled by his mobile device only show the vehicle flow rate in a small region. In a object tracking system, a mobile device is required to return the whole trajectory of the monitoring object, while it only has partial trajectory information. Therefore, the data sharing is quite important and common in a MEC system, and thus, the task assignment problem in such Data Shared Mobile Edge Computing System needs to be studied sufficiently. Furthermore, the limited computation abilities of mobile devices and base stations, the deadline constraint of each task are also ignored by the existing methods, which are very important during task assignment in a MEC system as well.

To overcome the above problems, this paper studied the task assignment algorithm in a Data-Shared MEC system. Considering the properties of the tasks, we distinguish the tasks into two categories, which are the holistic tasks and the divisible tasks. For the holistic tasks, the raw data transmission is inevitable since they cannot be processed distributedly, thus, we try to minimize the whole energy cost by transmission and computation. For the divisible tasks, which can be processed in distributed manner, we try to avoid the raw data transmission by migrating the computation and rearranging the tasks. Since the computation operations and partial results are much smaller than the raw data, lots of energy will be saved. In summary, the main contributions are as follows.

(1) The task assignment problem is firstly considered and formally defined in Data-Shared MEC Systems.

(2) For the holistic tasks, it is proved to be NP-complete for deciding the optimal task assignment in MEC systems. An approximation algorithm based on linear programming is proposed to solve it, and the detailed analysis on the correctness, the complexity and the ratio bound, is also given.

(3) For the divisible tasks, two algorithms are provided to meet different optimization goals, and the performance of the algorithms are also analyzed theoretically.

(4) The extensive experiments were carried out, where the experimental results show that all proposed algorithms were efficient in terms of latency and energy consumption.

The organization is as follows. Section II provides the system model and problem definitions. Sections III and IV gives task Assignment Algorithms for Holistic and Divisible tasks. Section V shows the experiment results. Section VI surveys related works and Section VII concludes the paper.

II. System Model

Without loss of generality, we assume that there exist k base stations,denoted by $B_1, B_2, ..., B_k$, and n users which are $U_1, U_2, ..., U_n$, in a MEC system. Each user U_i $(1 \leq i \leq n)$ has a corresponding mobile device, denoted by i. Each base station B_r $(1 \leq r \leq k)$ connects n_r mobile devices by radio access network, and these mobile devices form a cluster, where $\sum_{r=1}^{k} n_r = n$. Similar to the existing works such as [9], we also consider the quasi-static scenario during the task assignment, *i.e.* each mobile device connects the same base station in the period we considered.

According to the discussion in Section I, each mobile device will collect the computation tasks form users. For a computation task $\mathcal{T}_{ij}$(such as some data partitioned oriented task), which is the j-th computation task raised by user U_i, we use op_{ij} to denote the operations of task $\mathcal{T}_{ij}$ $(1 \leq i \leq n, 1 \leq j \leq m)$. Generally, a mobile device may not own all the data required by its computation tasks. Therefore, the input data of $\mathcal{T}_{ij}$ can be divided into two separated subsets, the local data LD_{ij} and the estimated external data ED_{ij}. Let $\alpha_{ij} = |LD_{ij}|$, $\beta_{ij} = |ED_{ij}|$, and L_{ij} denote the possible cluster (or mobile devices) that may own ED_{ij}. Besides the input data, each computation task also occupies some resources (*e.g.* memory) and has the deadline to meet, which are denoted by C_{ij} and T_{ij}, respectively. Thus, a computation task $\mathcal{T}_{ij}$ can be regarded as a tuple, that is, $\mathcal{T}_{ij} = (op_{ij}, LD_{ij}, ED_{ij}, L_{ij}, C_{ij}, T_{ij})$. For the convenience of discussion, we assume that every user raises the same number of tasks to a MEC system. All proposed algoritnms are also suitable for dealing with the opposite situation with minor revisions.

When the tasks are holistic, then all the local and external data should be collected to a single subsystem for processing, that is a computation task $\mathcal{T}_{ij}$ $(1 \leq i \leq n, 1 \leq j \leq m)$ could be carried out by one of three possible subsystems, *i.e.* the mobile device i, the base station connected to i or the remote cloud. We use the indicator variable x_{ijl} to show which subsystem is involved to deal with $\mathcal{T}_{ij}$, and x_{ijl} satisfies that

$$x_{ijl} = \begin{cases} 1, & \text{If } \mathcal{T}_{ij} \text{ is processed by subsystem } l \\ 0, & \text{Otherwise} \end{cases} \quad (1)$$

where Subsystem 1 (*i.e.* $l = 1$) means the mobile device i, Subsystem 2 (*i.e.* $l = 2$) is the base station connected to i and Subsystem 3 (*i.e.* $l = 3$) is regarded as the remote cloud.

Basically, the objective of assigning the computation tasks in a MEC system is to determine proper $\{x_{ijl} | 1 \leq i \leq n, 1 \leq j \leq m, l \in \{1, 2, 3\}\}$ which can minimize the energy cost of the system on condition that the deadlines of the tasks could be satisfied. There exist two main factors, *i.e.* computation and transmission, that influences the runtime and energy cost of each task. Therefore, we will show the models of computation and transmission in the following two subsections.

A. Computation Model

Since the first situation we considered is that the computation tasks are indivisible and holistic, only one subsystem is chosen to carried out $\mathcal{T}_{ij}$, so that $x_{ij1} + x_{ij2} + x_{ij3} = 1$. The situation that computation tasks are divisible will be considered in Section IV.

Let f_i represent the CPU frequency of mobile device i, $\lambda_{ij1}(y)$ be the function to define CPU cycles to process $\mathcal{T}_{ij}$ on condition that the input data size is y, where $\lambda_{ij1}(y)$ can be determined by the computation complexity of $\mathcal{T}_{ij}$. Thus, if $x_{ij1} = 1$, *i.e.* task $\mathcal{T}_{ij}$ is processed locally, then the computation time and energy cost for processing $\mathcal{T}_{ij}$ can be estimated by the following formulas,respectively.

$$t_{ij1}^{(C)} = \frac{\lambda_{ij1}(\alpha_{ij}+\beta_{ij})}{f_i},\ E_{ij1}^{(C)} = \kappa\lambda_{ij1}(\alpha_{ij}+\beta_{ij})f_i^2 \quad (2)$$

according to [6] and [14], where κ is a constant related to the hardware architecture.

Similarly, if $x_{ij2} = 1$ or $x_{ij3} = 1$, that is, the base station which connects with mobile device i or the remote cloud is selected to deal with task $\mathcal{T}_{ij}$, the computation time $t_{ij2}^{(C)}$ and $t_{ij3}^{(C)}$ can be determined by the following formulas,respectively.

$$t_{ij2}^{(C)} = \frac{\lambda_{ij2}(\alpha_{ij}+\beta_{ij})}{f_s},\ t_{ij3}^{(C)} = \frac{\lambda_{ij3}(\alpha_{ij}+\beta_{ij})}{f_c} \quad (3)$$

where $\lambda_{ij2}(\alpha_{ij}+\beta_{ij})$ and $\lambda_{ij3}(\alpha_{ij}+\beta_{ij})$ represents the CPU cycles of a base station and the cloud for dealing with the data with $\alpha_{ij}+\beta_{ij}$ size, f_s and f_c are the CPU frequencies of a base station and the cloud, respectively.

Since the energy cost of base station and the cloud during computation is extremely small comparing with that cost by transmission, therefore, they can be ignored.

B. Transmission Model

In order to obtain all the data for a task, each mobile device should exchange the data information with the connected base station by radio access network.

Let $r_i^{(U)}$ and $r_i^{(D)}$ be the upload and download rate of mobile device i $(1 \le i \le n)$. Both of them can be determined by [9] [10] using the Shannon Theory, that is $r_i^{(U)} = W_i^{(U)}\log_2(1+\frac{g_i^{(U)}P_i^{(T)}}{\varpi_0}), r_i^{(D)} = W_i^{(D)}\log_2(1+\frac{g_i^{(D)}P^{(S)}}{\varpi_0})$ where $g_i^{(U)}$ and $g_i^{(D)}$ are the uplink and downlink channel gains between mobile device i and the base station; $W_i^{(U)}$ and $W_i^{(D)}$ are the uplink and downlink channel bandwidths that the base station allocates for the mobile device i; $P_i^{(T)}$ and $P^{(S)}$ are the transmission powers of mobile device i and the base station; ϖ_0 is the power of the white noise.

Let $e_i^{(T)}(X)$, $e_i^{(R)}(X)$ and $e_{B,B}(X)$ denote energy cost of transmitting data with size X from mobile device i to the base station, from the base station to mobile device i, between two base stations, respectively. All of them also can be determined by the transmission rate and power according to [9].

Thus, if $x_{ij1} = 1$, then the mobile device i should retrieve the external data with the size β_{ij} from other mobile devices through the MEC system. Based on the definition of the task, L_{ij} will indicate which mobile device may own such external data. Therefore, the data retrieving time $t_{ij1}^{(R)}$ can be determined by the following formula.

$$t_{ij1}^{(R)} = \begin{cases} \frac{\beta_{ij}}{r_{L_{ij}}^{(U)}} + \frac{\beta_{ij}}{r_i^{(D)}}, & \text{If } L_{ij},\ i \text{ are in the same cluster} \\ \frac{\beta_{ij}}{r_{L_{ij}}^{(U)}} + \frac{\beta_{ij}}{r_i^{(D)}} + t_{B,B}(\beta_{ij}), & \text{Otherwise} \end{cases}$$

where $t_{B,B}(\beta_{ij})$ is the time spent for transmitting the data with size β_{ij} between two base stations.

The energy cost of retrieving the external data, $E_{ij1}^{(R)}$, satisfies that

$$E_{ij1}^{(R)} = \begin{cases} e_{L_i}^{(T)}(\beta_i) + e_i^{(R)}(\beta_i), & \text{If } L_{ij},\ i \text{ are in same cluster} \\ e_{L_i}^{(T)}(\beta_i) + e_i^{(R)}(\beta_i) + e_{B,B}(\beta_i) & \text{Otherwise} \end{cases}$$

Secondly, if $x_{ij2} = 1$, both of the local data and external data of $\mathcal{T}_{ij}$ should be transmitted to the base station for processing, and the result should be returned to mobile device i. Suppose that $\eta(y)$ represent the size of the computation result on condition that the input data size equals to y, then the information transmission time, $t_{ij2}^{(R)}$ satisfies the following formula.

$$t_{ij2}^{(R)} = \begin{cases} \max\{\frac{\beta_{ij}}{r_{L_{ij}}^{(U)}}, \frac{\alpha_{ij}}{r_i^{(U)}}\} + \frac{\eta(\alpha_{ij}+\beta_{ij})}{r_i^D}, \\ \qquad \text{If } L_{ij} \text{ and } i \text{ are in the same cluster} \\ \max\{\frac{\beta_{ij}}{r_{L_{ij}}^{(U)}} + t_{B,B}(\beta_j), \frac{\alpha_{ij}}{r_i^{(U)}}\} + \frac{\eta(\alpha_{ij}+\beta_{ij})}{r_i^D}, \text{ Else} \end{cases}$$

The total energy cost during transmission, denoted by $E_{ij2}^{(R)}$, can be calculated by Formula (4).

$$E_{ij2}^{(R)} = \begin{cases} e_{L_i}^{(T)}(\beta_i) + e_i^{(T)}(\alpha_i) + e_i^{(R)}(\eta(\alpha_{ij}+\beta_{ij})), \\ \qquad \text{If } L_{ij},\ i \text{ are in same cluster} \\ e_{L_i}^{(T)}(\beta_i) + e_i^{(T)}(\alpha_i) + e_i^{(R)}(\eta(\alpha_{ij}+\beta_{ij})) + e_{B,B}(\beta_i) \\ \qquad \text{Otherwise} \end{cases} \quad (4)$$

Thirdly, if $x_{ij3} = 1$, that is, the remote cloud is selected to deal with $\mathcal{T}_{ij}$. Thus, both of the local data and the external data are required to be delivered to the cloud for further processing. Let $t_{B,C}(X)$ and $e_{B,C}(X)$ be the transmission delay and energy cost of transmitting data with size X.

Then, the time and energy spent during transmission, denoted by $t_{ij3}^{(R)}$ and $E_{ij3}^{(R)}$, can be determined by the following two formulas when $x_{ij3} = 1$.

$$\begin{aligned} t_{ij3}^{(R)} &= \max\{\frac{\beta_{ij}}{r_{L_{ij}}^{(U)}}, \frac{\alpha_{ij}}{r_i^{(U)}}\} + \frac{\eta(\alpha_{ij}+\beta_{ij})}{r_i^D} \\ &\quad + t_{B,C}(\alpha_{ij}+\beta_{ij}+\eta(\alpha_{ij}+\beta_{ij})) \\ E_{ij3}^{(R)} &= e_{L_i}^{(T)}(\beta_i) + e_i^{(T)}(\alpha_i) + e_i^{(R)}(\eta(\alpha_{ij}+\beta_{ij})) \\ &\quad + e_{B,C}(\alpha_{ij}+\beta_{ij}+\eta(\alpha_{ij}+\beta_{ij})) \end{aligned}$$

Based on the above analysis, more data are required to transmit when $x_{ij3} = 1$, and the transmission time from a base station to the remote cloud is much larger than that between two base stations [15] [16], so that more energy will be consumed when $x_{ij3} = 1$, that is $E_{ij3}^{(R)} > E_{ij2}^{(R)}$.

C. Problem Definition

Based on the computation and transmission models, the total delay and energy cost for processing task $\mathcal{T}_{ij}$ when $l = 1, 2, 3$ are given as follows.

$$t_{ijl} = t_{ijl}^{(C)} + t_{ijl}^{(R)}, E_{ijl} = \begin{cases} E_{ijl}^{(R)} + E_{ijl}^{(C)} & \text{if } l = 1 \\ E_{ijl}^{(R)} & \text{if } l = 2, 3 \end{cases} \quad (5)$$

Meanwhile, the resources (*e.g.* memory, thread, virtual machine, processor) provided by mobile devices and base stations for computation are limited as discussed in Section I, so that we use max_i and max_S to denote upper bound of such resource. Thus, $\sum_{j=1}^{m} C_{ij}x_{ij1} \leq max_i$ and $\sum_{i=1}^{n}\sum_{j=1}^{m} C_{ij}x_{ij2} \leq max_S$, which means that the total resource occupied by the tasks assigned to mobile device i and a base station should be no more than max_i and max_S.

Thus, the problem of Holistic Task Assignment (HTA) in a MEC system is defined as follows.

Input:

1) A MEC system with n mobile devices,$\{1, 2, ...n\}$, and k base stations, $\{B_1, B_2, ..., B_k\}$;
2) A set of computation tasks $\{\mathcal{T}_{ij}|1 \leq i \leq n, 1 \leq j \leq m\}$, and the estimated CPU cycles and result size for each computation task.
3) Basic frequencies of mobile devices, base station and the cloud, $\{f_i|1 \leq i \leq n\}$, f_s, f_c;
4) The Network parameters, $r_i^{(U)}$, $r_i^{(D)}$, $t_{B,B}(X)$, $t_{B,C}(X)$ $e_i^{(T)}(X)$, $e_i^{(R)}(X)$, $e_{B,B}(X)$ and $e_{B,C}(X)$;
5) The limitations on computation resources of each mobile devices and the base station $\{max_i|1 \leq i \leq n\}$, max_S.

Output: The task assignment strategy $\{x_{ijl}|1 \leq i \leq n, 1 \leq j \leq m, l = 1, 2, 3\}$, where the total energy cost of the system, *i.e.* $\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{l=1}^{3} E_{ijl}x_{ijl}$, is minimized, and the following conditions are satisfied.

$$\sum_{l=1}^{3} t_{ijl}x_{ijl} \leq T_{ij} \quad \forall i \in N, \forall j \in M, \tag{C1}$$

$$\sum_{j=1}^{m} C_{ij}x_{ij1} \leq max_i \quad \forall i \in N, \tag{C2}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m} C_{ij}x_{ij2} \leq max_S \tag{C3}$$

$$\sum_{l=1}^{3} x_{ijl} = 1 \quad \forall i \in N, \forall j \in M, \tag{C4}$$

$$x_{ijl} \in \{0, 1\}, \forall i \in N, \forall j \in M, \forall l \in \{1, 2, 3\} \tag{C5}$$

The hardness and solution of such problem will be addressed in Section III.

III. Holistic Task Assignment Algorithm

Before introducing the algorithm, we firstly analyze the hardness of HTA problem.

Theorem 1. *The HTA problem is NP-complete.*

Proof Considering a special case of the HTA problem, that is, $max_i = 0$, $T_{ij} = \infty$ for all $1 \leq i \leq n$ and $1 \leq j \leq m$, which the deadline of each task can be ignored and the mobile devices does not have any ability for processing tasks.

Therefore, we have that $x_{ij1} = 0$ and $x_{ij3} = 1 - x_{ij2}$ and the HTA problem is equal to the following one

$$\begin{aligned} P1: \max \ & \sum_{i=1}^{n}\sum_{j=1}^{m}(E_{ij3} - E_{ij2})x_{ij2} \\ s.t. \ & \sum_{i=1}^{n}\sum_{j=1}^{m} C_{ij}x_{ij2} \leq max_S \\ & x_{ij2} \in \{0, 1\}, \forall i \in N, \forall j \in m \end{aligned}$$

Since $E_{ij3} > E_{ij2}$, the problem $P1$ can be regarded as a Knapsack problem with $n \times m$ items, the value and weight of item (i, j) equals to $E_{ij3} - E_{ij2}$ and C_{ij}, and the capacity of Knapsack is max_S. Since Knapsack problem is NP-complete, that is, the special case of the HTA problem is NP-complete, so that HTA problem is at least NP-complete. □

Since HTA problem is NP-complete, the following subsection will proposed an approximation algorithm to solve it.

A. Linear Programming based Algorithm

Based on Section II, there only exist three possible subsystems to process task $\mathcal{T}_{ij}$, which are mobile device i, the connected base station B_r and the remote cloud. Therefore, each cluster can be considered separately, so that this section will discuss how to assignment the task in a cluster.

Let $(\mathbf{y_1}, \mathbf{y_2}, ..., \mathbf{y_k})$ denoted the combination of k vectors. For example, $(\mathbf{y}, \mathbf{z}) = (y_1, y_2, ..., y_r, z_1, z_2, ..., z_l)$ if $\mathbf{y} = (y_1, y_2, ..y_r)$ and $\mathbf{z} = (z_1, z_2., , , z_l)$. Thus, we use $\xi_{ij} = (x_{ij1}, x_{ij2}, x_{ij3})$ to represent the assignment of task $\mathcal{T}_{ij}$, and then $\xi_i = (\xi_{i1}, \xi_{i2}, ..., \xi_{im})$ denotes the assignment result of all the tasks given by user U_i, and vector $\xi = (\xi_1, \xi_2, ..., \xi_{n_r})$ denote the assignment result of all the tasks in the cluster. By the same way, $\mathbf{e}_{ij} = (E_{ij1}, E_{ij2}, E_{ij3})$ denotes the possible energy cost of task $\mathcal{T}_{ij}$, and we set $\mathbf{e}_i = (\mathbf{e}_{i1}, \mathbf{e}_{i2}, \ldots, \mathbf{e}_{im})$, and $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{n_r})$.

Obviously, both of ξ and $\mathbf{e}$ are the vectors with $3n_rm$ size. Based on ξ and $\mathbf{e}$ and the definition of the HTA problem, the relax linear programming problem is formalized as follows.

$$\begin{aligned} P2: \min_{\xi} \ & \mathbf{e}\xi^T \\ s.t. \ & \mathbf{A}_1\xi^T \leq \mathbf{b}_1, \quad \mathbf{A}_2\xi^T \leq \mathbf{b}_2, \\ & \mathbf{A}_3\xi^T \leq \mathbf{b}_3, \quad \mathbf{A}_4\xi^T = \mathbf{b}_4, \\ & \xi[i] \in [0, 1], \forall i = 1, 2, \ldots, 3 \times n_r \times m \end{aligned}$$

where $\xi[i]$ is the i-th element of vector ξ, $\mathbf{b}_1 = (T_{11}, T_{11}, T_{11}, \ldots, T_{ij}, T_{ij}, T_{ij}, \ldots, T_{n_rm}, T_{n_rm}, T_{n_rm})^T_{1\times 3n_rm}$, $\mathbf{b}_2 = (max_1, max_2 \ldots, max_i, \ldots, max_{n_r})^T_{1\times n_r}$, $\mathbf{b}_3 = (max_S)$, $\mathbf{b}_4 = (1, 1, \ldots, 1)^T_{n_rm\times 1}$, $\mathbf{A}_1 = diag(t_{111}, t_{112}, t_{113}, ..., t_{n_rm1}, t_{n_rm2}, t_{n_rm3})_{3n_rm\times 3n_rm}$, $\mathbf{A}_3 = [0, C_{i1}, 0, 0, C_{i2}, 0, \ldots, 0, C_{n_rm}, 0]_{1\times 3n_rm}$,

$$\mathbf{A}_2 = \begin{bmatrix} \mathbf{a}_1 & \mathbf{0}_{1\times 3m} & \cdots & \mathbf{0}_{1\times 3m} \\ \mathbf{0}_{1\times 3m} & \mathbf{a}_2 & \cdots & \mathbf{0}_{1\times 3m} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{1\times 3m} & \mathbf{0}_{1\times 3m} & \cdots & \mathbf{a}_{n_r} \end{bmatrix}_{n_r\times 3n_rm},$$

$$\mathbf{A}_4 = \begin{bmatrix} \mathbf{a}_{11} & \mathbf{0}_{1\times 3} & \cdots & \mathbf{0}_{1\times 3} \\ \mathbf{0}_{1\times 3} & \mathbf{a}_{12} & \cdots & \mathbf{0}_{1\times 3} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} & \cdots & \mathbf{a}_{n_rm} \end{bmatrix}_{n_rm\times 3n_rm},$$

and $\mathbf{a}_i = (C_{i1}, 0, 0, \ldots, C_{im}, 0, 0)_{1\times 3m}$,and $\mathbf{a}_{ij} = (1, 1, 1)$.

Based on above symbols, the Holistic Task Assignment Algorithm for (*LP-HTA* for short) has six steps.

Step 1. Solve the linear programming problem $P2$ to obtain ξ by using the interior points method algorithm given in [17].

Step 2. Construct an equivalent fractional matrix $\mathbf{X}$ according to ξ. That is, $\mathbf{X}[i, j, l] = \xi[3m \times (i-1) + 3 \times (j-1) + l]$ for any $1 \leq i \leq n_r$, $1 \leq j \leq m$ and $l \in \{1, 2, 3\}$.

Step 3. If all elements of $\mathbf{X}$ are binary, then set $x_{ijl} = X[i,j,l]$, and return $\{x_{ijl}|1 \le i \le n_r, 1 \le j \le m, l \in \{1,2,3\}\}$ as the assignment result. Otherwise, let $q = \text{argmax}_{l\in\{1,2,3\}} \mathbf{X}[i,j,l]$, (*i.e.* $\mathbf{X}[i,j,q] = \max\{\mathbf{X}[i,j,1], \mathbf{X}[i,j,2], \mathbf{X}[i,j,3]\}$), then set

$$\widehat{x}_{ijl} = \begin{cases} 1 & \text{if } l = q \\ 0 & \text{Otherwise} \end{cases} \quad (6)$$

Step 4. Considering each $\widehat{x}_{ijq} = 1$, if $t_{ijq} \le T_{ij}$, set $x_{ijq} = 1$ and $x_{ijp} = 0$ for any $p \in \{1,2,3\}$ and $p \ne q$. Otherwise, find l which satisfies that $l = \text{argmax}_{p\in\{p|t_{ijp}\le T_{ij}\}} \mathbf{X}[i,j,p]$, then set $x_{ijl} = 1$ and $x_{ijr} = 0$ for any $r \in \{1,2,3\}$ and $r \ne l$. If we cannot find such l, then we cancel $\mathcal{T}_{ij}$ and inform users.

Step 5. For each mobile device i, reconsidering Constraint $C2$ in HTA problem. Compute $\sum_{j=1}^{m} C_{ij}x_{ij1}$. When $\sum_{j=1}^{m} C_{ij}x_{ij1} > max_i$, the following steps are carried out.

1) Select tasks from $S_1 = \{\mathcal{T}_{ij}|x_{ij1} = 1 \wedge t_{ij2} \le T_{ij}\}$ greedily according to the resource they occupy (*i.e.* C_{ij});
2) Set $x_{ij1} = 0$ and $x_{ij2} = 1$ for any selected task $\mathcal{T}_{ij}$, that is, remove the task $\mathcal{T}_{ij}$ to the base station for processing;
3) Repeat the above two steps until $\sum_{j=1}^{m} C_{ij}x_{ij1} \le max_i$ or $\{\mathcal{T}_{ij}|x_{ij1} = 1 \wedge t_{ij2} \le T_{ij}\} = \emptyset$;

If $\sum_{j=1}^{m} C_{ij}x_{ij1}$ is still larger than max_i, greedily select tasks with high resource occupation to cancel and inform users.

Step 6. For the base station, we do the similar operations as Step 5. First, check whether $\sum_{i=1}^{n_r}\sum_{j=1}^{m} C_{ij}x_{ij2} \le max_S$ is satisfied. If it is, then return the assignment result $\{x_{ijl}|1 \le i \le n_r, 1 \le j \le m, l \in \{1,2,3\}\}$. Otherwise,

1) Calculate $S_2 = \{\mathcal{T}_{ij}|x_{ij2} = 1 \wedge t_{ij3} \le T_{ij}\}$. If $S_2 \ne \emptyset$, select $\mathcal{T}_{ij}$ from S_2 greedily according to C_{ij}. Set $x_{ij2} = 0$ and $x_{ij3} = 1$, *i.e.* remove $\mathcal{T}_{ij}$ to Cloud to process.
2) Repeat the above step until $\sum_{i=1}^{n_r}\sum_{j=1}^{m} C_{ij}x_{ij2} \le max_S$ or $\{\mathcal{T}_{ij}|x_{ij2} = 1 \wedge t_{ij3} \le T_{ij}\} = \emptyset$

If $\sum_{i=1}^{n_r}\sum_{j=1}^{m} C_{ij}x_{ij2} \le max_S$ still can not be satisfied, then greedily select some tasks with high resource occupation, cancel them and inform the users.

B. Performance Analysis of the Algorithm

1) Correctness: Firstly, we need to verifies that *LP-HTA* algorithm provided a feasible solution for the HTA problem.

Suppose that $\{x_{ijl}|1 \le i \le n_r, 1 \le j \le m, l \in \{1,2,3\}\}$ be the assignment result returned by *LP-HTA* algorithm. As shown in the algorithm, we only set one of x_{ij1}, x_{ij2} and x_{ij3} to be 1, and set the other two to be 0, so that Constraints $C4$ and $C5$ in HTA problem are certainly satisfied.

Meanwhile, **Step 4** of the algorithm considers the deadline constraint of each task. That is, x_{ijl} is set to be 1 only if $t_{ijl} \le T_{ij}$, where $1 \le i \le n_r$, $1 \le j \le m$ and $l \in \{1,2,3\}$. Furthermore, the deadline constraint is also considered in **Step 5** and **Step 6** when we remove the tasks to base station or to the cloud. Thus, Constraint $C1$ of HTA problem is satisfied.

Finally, **Step 5** of the algorithm guarantees that $\sum_{j=1}^{m} C_{ij}x_{ij1} \le max_i$ for each mobile i $(1 \le i \le n_r)$, so that Constraint $C2$ is satisfied. Similarly, Constrain $C3$ in HTA problem is also meet by applying **Step 6** of algorithms

In summary, $\{x_{ijl}|1 \le i \le n_r, 1 \le j \le m, l \in \{1,2,3\}\}$ satisfies all the constraints, so it is a feasible solution for the HTA problem.

2) Complexity: Next, we will analyze the complexity of *LP-HTA* algorithm.

In **Step 1**, the interior points method is used for solve the linear programming problem and obtain ξ, which is a vector with $3n_rm$ size. The complexity is $O((n_rm)^{3.5})$ according to [17]. In **Step 2** and **Step 3**, we construct the equivalent fractional matrix $\mathbf{X}$ and $\{\widehat{x}_{ijl}|1 \le i \le n_r, 1 \le j \le m, l \in \{1,2,3\}\}$ according to ξ. The complexity of such construction is $O(n_rm)$. In **Step 4**, **Step 5** and **Step 6**, we adjust $\{\widehat{x}_{ijl}|1 \le i \le n_r, 1 \le j \le m, l \in \{1,2,3\}\}$ according the deadline and processing abilities constraints, and get $\{x_{ijl}|1 \le i \le n_r, 1 \le j \le m, l \in \{1,2,3\}\}$ finally. The complexity of such adjustment is still $O(n_rm)$. **In summary**, the total complexity of *LP-HTA* algorithm is $O((n_rm)^{3.5})$.

3) Ratio Bound: Before discussing the ratio bound of *LP-HTA* algorithm, the following lemma is proved.

Lemma 1. *The intermediate result* $\{\widehat{x}_{ijl}|1 \le i \le n_r, 1 \le j \le m, l \in \{1,2,3\}$ *calculated by Step 3 in the LP-HTA algorithm satisfies that* $\sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3} E_{ijl}\widehat{x}_{ijl} \le 3\sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3} E_{ijl}x_{ijl}^{OPT}$, *where* $\{x_{ijl}^{OPT}|1 \le i \le n_r, 1 \le j \le m, l \in \{1,2,3\}\}$ *be the optimal assignment for HTA problem.*

Proof. According to the definition of relax linear programming problem, *i.e.* Problem $P2$, we have $\mathbf{X}[i,j,1]+\mathbf{X}[i,j,2]+\mathbf{X}[i,j,3] = 1$. Therefore, $\max_{l=1,2,3} \mathbf{X}[i,j,l] \ge \frac{1}{3}$.

Let r satisfies that $\mathbf{X}[i,j,r] = \max_{l=1,2,3} \mathbf{X}[i,j,l]$. According to **Step 3**, we set $\widehat{x}_{ijr} = 1$, and $\widehat{x}_{ijl} = 0$ if $l \ne r$. Since $\mathbf{X}[i,j,r] = \max_{l=1,2,3} \mathbf{X}[i,j,l] \ge \frac{1}{3} = \frac{1}{3}\widehat{x}_{ijr}$, we have $\sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3} E_{ijl}\widehat{x}_{ijl} = \sum_{i=1}^{n_r}\sum_{j=1}^{m} E_{ijr}\widehat{x}_{ijr} \le 3\sum_{i=1}^{n_r}\sum_{j=1}^{m} E_{ijr}\mathbf{X}[i,j,r] \le 3\sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3} E_{ijl}\mathbf{X}[i,j,l]$

where $\sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3} E_{ijl}\mathbf{X}[i,j,l]$ is the optimal result of the linear programming problem.

Since $\{x_{ijl}^{OPT}|1 \le i \le n_r, 1 \le j \le m, l \in \{1,2,3\}\}$ be the optimal assignment of HTA problem, so that it is just a feasible solution of the linear programming problem. Therefore, $\sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3} E_{ijl}\mathbf{X}[i,j,l] \le \sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3} E_{ijl}x_{ijl}^{OPT}$. Thus, we have $\sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3} E_{ijl}\widehat{x}_{ijl} \le 3\sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3} E_{ijl}x_{ijl}^{OPT}$. □

According to *LP-HTA* algorithm, **Step 4**, **Step 5** and **Step 6** are used to adjust $\{\widehat{x}_{ijl}|1 \le i \le n_r, 1 \le j \le m, l \in \{1,2,3\}$ to satisfy Constraints $C1$, $C2$ and $C3$. In another words, some tasks are migrated among mobile devices, the base station and the cloud to meet the constraints. Let Δ be the growth of energy cost caused by the above migration, and $E_{LP}^{(OPT)} = \sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3} E_{ijl}\mathbf{X}[i,j,l]$ be the optimal result of the linear programming algorithm $P2$, then the ratio bound of *LP-HTA* algorithm meets the following theorem.

Theorem 2. *The ratio bound of LP-HTA Algorithm, i.e. R, is*

no more than $3+\frac{\Delta}{E_{LP}^{(OPT)}}$.

Proof. According to Lemma 1, we have $\sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3}E_{ijl}\widehat{x}_{ijl} \leq 3\sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3}E_{ijl}x_{ijl}^{OPT}$, where $\{\widehat{x}_{ijl}|1 \leq i \leq n_r, 1 \leq j \leq m, l \in \{1,2,3\}\}$ is intermediate result obtained by **Step 3** of *LP-HTA* algorithm.

Since Δ is the growth of energy cost caused by the task migration and $E_{LP}^{(OPT)} = \sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3}E_{ijl}\mathbf{X}[i,j,l] \leq \sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3}E_{ijl}x_{ijl}^{OPT}$ based on **Lemma 1**, we have $R = \frac{\sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3}E_{ijl}\widehat{x}_{ijl}+\Delta}{\sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3}E_{ijl}x_{ijl}^{OPT}} \leq 3+\frac{\Delta}{E_{LP}^{(OPT)}}$.□

The parameter Δ and $E_{LP}^{(OPT)}$ in **Theorem 2** can be determined during the execution of *LP-HTA* algorithm.

Furthermore, the energy cost by transmission usually is much larger than that cost by computation, so that $E_{ij1} < E_{ij2} < E_{ij3}$ since more data are transmitted if we use base station or cloud to deal with task $\mathcal{T}_{ij}$. Therefore, we have the following corollary.

Corollary 1. *The ratio bound of the LP-HTA algorithm, denoted by R, satisfied that* $R \leq \min\{\frac{\max_{1\leq i\leq n_r,1\leq j\leq m}E_{ij3}}{\min_{1\leq i\leq n_r,1\leq j\leq m}E_{ij1}}, 3+\frac{\Delta}{E_{LP}^{(OPT)}}\}$, *on condition that* $E_{ij1} < E_{ij2} < E_{ij3}$.

Proof. Since $E_{ij1} < E_{ij2} < E_{ij3}$, we have $\sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3}E_{ijl}x_{ijl} \leq \sum_{i=1}^{n_r}\sum_{j=1}^{m}E_{ij3}$ and $\sum_{i=1}^{n_r}\sum_{j=1}^{m}\sum_{l=1}^{3}E_{ijl}x_{ijl}^{OPT} \geq \sum_{i=1}^{n_r}\sum_{j=1}^{m}E_{ij1}$, where $\{x_{ijl}|1 \leq i \leq n_r, 1 \leq j \leq m, l \in \{1,2,3\}\}$ is the assignment result return by *LP-HTA* algorithm.

Thus, $R \leq \frac{\max_{1\leq i\leq n_r,1\leq j\leq m}E_{ij3}}{\min_{1\leq i\leq n_r,1\leq j\leq m}E_{ij1}}$. That is, $R \leq \min\{\frac{\max_{1\leq i\leq n_r,1\leq j\leq m}E_{ij3}}{\min_{1\leq i\leq n_r,1\leq j\leq m}E_{ij1}}, 3+\frac{\Delta}{E_{LP}^{(OPT)}}\}$ based on Theorem 2. □

Since $R \leq \min\{\frac{\max_{1\leq i\leq n_r,1\leq j\leq m}E_{ij3}}{\min_{1\leq i\leq n_r,1\leq j\leq m}E_{ij1}}, 3+\frac{\Delta}{E_{LP}^{(OPT)}}\} \leq \frac{\max_{1\leq i\leq n_r,1\leq j\leq m}E_{ij3}}{\min_{1\leq i\leq n_r,1\leq j\leq m}E_{ij1}}$ according to Corollary 1, and $n_r m$ is quite large comparing with $\frac{\max_{1\leq i\leq n_r,1\leq j\leq m}E_{ij3}}{\min_{1\leq i\leq n_r,1\leq j\leq m}E_{ij1}}$, thus the ratio bound of *LP-HTA* algorithm can be regarded as a constant.

IV. Divisible Tasks Assignment Algorithm

Besides the holistic computation tasks, there also exist many divisible tasks in a MEC system. We call a task to be divisible if and only if it can be implemented distributedly, *i.e.* the final result can be obtained by aggregating the partial results. For example, some statistics calculations, such as *Sum* or *Count*, can be regarded as divisible tasks. Considering that the partial results are always much smaller than the raw data, therefore, it will save much energy if we process such tasks in distributed manner. Furthermore, it is also useful to protect privacy if the tasks are processed distributedly in a MEC system [18].

Let D_i be the data owned by mobile device i, and $\{\mathcal{T}_{ij}|1 \leq i \leq n, 1 \leq j \leq m\}$ denote the set of all the tasks in a MEC system. Then, $D = \bigcup_{1\leq i\leq n,1\leq j\leq m}(LD_{ij} \bigcup ED_{ij})$ denote the total data required to be processed, where $D \subseteq \bigcup_{i=1}^{n} D_i$ and $D_i \bigcap D_j$ may not be empty since the monitoring regions of two devices may overlap with each other.

In order to reduce the energy cost as much as possible, we should rearrange the tasks according to $\{D_i|1 \leq i \leq n\}$. The aim of such rearrangement is that the new tasks assigned to each mobile device only need to deal with the local data. To achieve such aim and avoid the raw data transmission, the following two problems need to be solved.

(1). For each $1 \leq i \leq n$, how to determine the dataset that mobile device i needs to process?

(2). How to arrange the tasks using the above result?

Obviously, the first problem is more critical for the task arrangement in a MEC system. The following two subsections will discuss it based on different optimization goals, The last one will provide a solution for the second problem.

A. Data Division for Balancing the Workload

To determine the dataset for each mobile device i $(1 \leq i \leq n)$ to process, we need divide D into several disjoint subsets, and each mobile device only response for dealing with a subset. Firstly, we want to divide D as uniform as possible due to two reasons. First, the computation workload of each mobile device will be balanced if D is divided uniformly, so that the energy consumption of each mobile edge will be balanced as well. Second, since the mobile device can process the tasks in parallel manner, the longest time spent by a mobile device will be shorten effectively if D is divided uniformly, which results in a smaller total processing time of the whole system. Based on such motivation, the definition of the *Optimal Coverage of D with Smallest Set Size* is given as follows.

Definition 1. Given D, $\{D_i|1 \leq i \leq n\}$, $\mathcal{C}^{(S)} = \{C_1^{(S)}, C_2^{(S)}, ..., C_n^{(S)}\}$ is called as an *Optimal Coverage of D with Smallest Set Size* if and only if $\mathcal{C}^{(S)}$ satisfies the following conditions: **(1)** $C_i^{(S)} \subset D \bigcap D_i$; **(2)** $C_i^{(S)} \bigcap C_j^{(S)} = \emptyset$ for any $1 \leq i \neq j \leq n$, and $\bigcup_{i=1}^{n} C_i^{(S)} = D$; and **(3)** $\max_{1\leq i\leq n}|C_i^{(S)}|$ is minimized.

The first condition of Definition 1 means that $C_i^{(S)}$ can be processed by mobile device i locally since $C_i^{(S)} \subseteq D_i$, and mobile device i need not to deal with any data if $C_i^{(S)} = \emptyset$. The second condition indicates that D is fully processed since $\bigcup_{i=1}^{n} C_i^{(S)} = D$, meanwhile, the redundant computation is avoided as $C_i^{(S)} \bigcap C_j^{(S)} = \emptyset$. The last condition guarantees that the processing time has been reduced as much as possible since $\max_{1\leq i\leq n}|C_i^{(S)}|$ is minimized.

Let $D = \{d_1, d_2, ..., d_M\}$ and $UD_i = D \bigcap D_i$ for all $1 \leq i \leq n$, where d_i is regarded as a data item or a data block determined by [19]. For $1 \leq r \leq M, 1 \leq i \leq n$, let y_{ri} be a indicator variable satisfies that $y_{ri} = 1$ if and only if d_r is assigned to mobile device i for processing, and p_{ri} denote the cost of such assignment. Since we want to avoid the raw data transmission, $p_{ri} = \infty$ if $d_r \notin UD_i$, otherwise $p_{ri} = 1$. Thus, the problem of calculating $\mathcal{C}^{(S)} = \{C_1^{(S)}, C_2^{(S)}, ..., C_n^{(S)}\}$, can be formalized as the following 0-1 programming problem.

$$
\begin{aligned}
P3: \min\ & maxsize \\
s.t.\ & \sum_{i=1}^{n} y_{ri} = 1, && 1 \leq r \leq M, \\
& \sum_{r=1}^{M} y_{ri}p_{ri} \leq maxsize, && 1 \leq r \leq N, \\
& y_{ri} \in \{0,1\}, && 1 \leq r \leq M, 1 \leq i \leq n.
\end{aligned}
$$

where $C_i = \{d_r | y_{ri} = 1\}$ for all $1 \leq i \leq n$.

Since 0-1 programming problem is NP-complete [20], it is hard to determine $\mathcal{C}^{(\mathcal{S})} = \{C_1^{(S)}, C_2^{(S)}, ..., C_n^{(S)}\}$ in polynomial time. Then, a greedy algorithm with $O(n)$ complexity is presented as follows. The algorithm contains three steps.

Step 1. Let $C_1^{(S)} = \emptyset$, $C_2^{(S)} = \emptyset$,..., $C_n^{(S)} = \emptyset$.

Step 2. Determine r by $r = \text{argmin}_{1 \leq i \leq n} |UD_i \cap D|$, set $C_r^{(S)} = UD_r \bigcap D$, and $D = D - C_r^{(S)}$.

Step 3. Repeat Step 2 until $D = \emptyset$. Return $\mathcal{C}^{(\mathcal{S})} = \{C_1^{(S)}, C_2^{(S)}, ..., C_n^{(S)}\}$.

The following theorem and corollary guarantee that the ratio bound of the above greedy algorithm is $\frac{1}{1-e^{-1}}$.

Theorem 3. *$f(\mathcal{X}) = \max_{A \in \mathcal{X}} |A|$ is a submodular function, where $\mathcal{X} \subseteq 2^D$, and $A \subseteq D$.*

Proof. Let $\mathcal{X} \subseteq 2^D$ and $\mathcal{Y} \subseteq 2^D$ be two sets satisfy that $\mathcal{X} \subseteq \mathcal{Y}$. Thus, we have $f(\mathcal{X}) = \max\limits_{A \in \mathcal{X}} |A| \leq \max\limits_{B \in \mathcal{Y}} |B| = f(\mathcal{Y})$.

Let $G \subseteq D$ and $G \notin \mathcal{Y}$. Therefore, there exists three cases that needs to be discussed.

Case 1. If there at least exists $\exists A \in \mathcal{X}$ satisfies that $A| > |G|$, then $f(\mathcal{X} \cup \{G\}) - f(\mathcal{X}) = 0$. Since $\mathcal{X} \subseteq \mathcal{Y}$, we have $A \in \mathcal{Y}$, and thus $f(\mathcal{X} \cup \{G\}) - f(\mathcal{X}) = 0 = f(\mathcal{Y} \cup \{G\}) - f(\mathcal{Y})$

Case 2. If $|G| > \max_{A \in \mathcal{X}} |A| (= f(\mathcal{X}))$, however, $\exists B \in \mathcal{Y} - \mathcal{X}$ satisfies that $|B| > |G|$, then we have $f(\mathcal{X} \cup \{G\}) - f(\mathcal{X}) = |G| - f(\mathcal{X}) > 0 = f(\mathcal{Y} \cup \{G\}) - f(\mathcal{Y})$

Case 3. If $|G| > \max_{B \in \mathcal{Y}} |B| (= f(\mathcal{Y})$, then $f(\mathcal{X} \cup \{G\}) - f(\mathcal{X}) = |G| - f(\mathcal{X}) \geq |G| - f(\mathcal{Y}) = f(\mathcal{Y} \cup \{G\}) - f(\mathcal{Y})$ since $f(\mathcal{X}) \leq f(\mathcal{Y})$

In summary, we have $f(\mathcal{X} \cup \{G\}) - f(\mathcal{X}) \geq f(\mathcal{Y} \cup \{G\}) - f(\mathcal{Y})$, so that $f(\mathcal{X})$ is a submodular function. □

Since $f(\mathcal{X})$ is a submodular function, the following Corollary will be obtained.

Corollary 2. *The ratio bound of greedy algorithm is $\frac{1}{1-e^{-1}}$.*

B. Data Division for Minizing Involved Mobile Devices

Secondly, in order to save energy for majority mobile devices, some applications may require to minimize the number of devices for dealing with the tasks. Based on such motivation, the *Optimal Coverage of D with Smallest Set Number* is defined as follows.

Definition 2. $\mathcal{C}^{(\mathcal{N})} = \{C_{l_1}^{(N)}, C_{l_2}^{(N)}, ..., C_{l_r}^{(N)}\}$ is called as an *Optimal Coverage of D with Smallest Set Number* if and only if $\mathcal{C}$ satisfies: **(1)** $C_{l_i}^{(N)} \neq \emptyset$, $C_{l_i}^{(N)} \subset D \bigcap D_{l_i}$; **(2)** $C_{l_i}^{(N)} \bigcap C_{l_j}^{(N)} = \emptyset$ for any $1 \leq i \neq j \leq n$, and $\bigcup_{i=1}^{l} C_{l_i}^{(N)} = D$; **(3)** For all $\mathcal{C}'$ which satisfies the above two conditions, we have $|\mathcal{C}'| \geq |\mathcal{C}^{(\mathcal{N})}| (= r)$.

Considering D is a universe, and $\mathcal{S}^{(\mathcal{N})} = \{UD_1, UD_2, ..., UD_n\}$ is a family of subsets of D, where $UD_i = D \bigcap D_i$ as given in Section IV.A. Therefore, the problem of calculating *Optimal Coverage of D with Smallest Set Number*, *i.e.* $\mathcal{C}^{(\mathcal{N})} = \{C_{l_1}^{(N)}, C_{l_2}^{(N)}, ..., C_{l_r}^{(N)}\}$, can be regarded as the Set Cover Problem with input D and $\mathcal{S}^{(\mathcal{N})}$. Since the Set Cover Problem is proved to be NP-complete in [21], therefore, it is hard to obtain $\mathcal{C}^{(\mathcal{N})} = \{C_{l_1}^{(N)}, C_{l_2}^{(N)}, ..., C_{l_r}^{(N)}\}$ in polynomial time. The following part will provide a greedy algorithm to get approximation result The algorithm consists of three steps, and whose pseudocode is given in Algorithm 1.

First, let $UD_i = D \bigcap D_i$ for all $1 \leq i \leq n$, and $\mathcal{C}^{(\mathcal{N})} = \emptyset$.

Second, determine r by $r = \text{argmax}_{1 \leq i \leq n} |UD_i \cap D|$, let $\mathcal{C}^{(\mathcal{N})} = \mathcal{C}^{(\mathcal{N})} \bigcup \{UD_r \cap D\}$ and $D = D - (UD_r \cap D)$;

Third, repeat Step 2 until $D = \emptyset$. Return $\mathcal{C}^{(\mathcal{N})}$.

Obversely, the complexity of the above algorithm is $O(n)$. Its ration bound is $O(\ln n)$.

C. Task Rearrangement Method

After obtaining $\mathcal{C}^{(\mathcal{S})} = \{C_1^{(S)}, C_2^{(S)}, ..., C_l^{(S)}\}$ or $\mathcal{C}^{(\mathcal{N})} = \{C_1^{(N)}, C_2^{(N)}, ..., C_l^{(N)}\}$, the computation tasks, $\{T_{ij} | 1 \leq i \leq n, 1 \leq j \leq m\}$ will be rearranged based on them.

Take $\mathcal{C}^{(\mathcal{S})} = \{C_1^{(S)}, C_2^{(S)}, ..., C_l^{(S)}\}$ as an example. For each device i, the dataset that it needs to process is $C_i^{(S)}$, therefore, the information of the task $\mathcal{T}_{rl}$, including op_{rl}, C_{rl} and T_{rl}, are transmit to mobile device i if $C_i^{(S)} \bigcap (LD_{ij} \bigcup ED_{ij}) \neq \emptyset$. Through such arrangement, every mobile devices are assigned a series of new computation tasks. Then, the *LP-HTA* algorithm in Section III is applied to schedule these new tasks and obtain the partial results. Finally, the partial results are aggregated according to users' requirements.

As only the task information and partial results are required to transmit in a MEC system, much energy will be saved.

V. Experiment Results

A. Experiment Settings

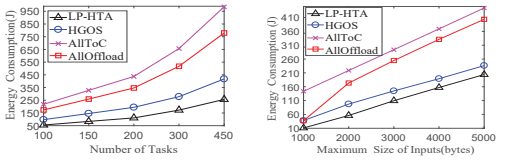
Similar as [22], we also assume that the energy cost, required CPU cycles and result size is linear to the size of input data in the simulated MEC system. That is, the required CPU cycles, the energy cost, and the result size can be estimated by λX, $\kappa \lambda X$ and ηX when the size of input data is X, where $\kappa = 10^{-27}$ (J/cycle), $\lambda = 330$ (cycle/byte), $\eta = 0.2$ based on [22]. The CPU frequencies of mobile devices varies from 1GHz to 2GHz. For a based station, its CPU frequency is set to be 4GHz, and the transmission delay between two base stations are 15ms [15]. For the cloud, we take Amazon T2.nano as example, and set its CPU frequency to be 2.4GHz, the delay between a base station and Cloud is 250ms based on [16]. Finally, each mobile device connects with the base station by 4G or WiFi randomly, where the download, upload, transmitting and receiving powers of simulated wireless network are summarized in Table I based on [23] and [24].

TABLE I: parameters of wireless networks

NetWork	Download speed	Upload speed	P^T	P^R
4G	13.76 Mbps	5.85 Mbps	7.32 W	1.6W
Wi-Fi	54.97 Mbps	12.88 Mbps	15.7 W	2.7 W

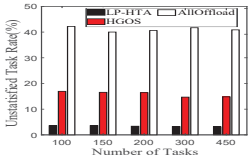
B. Performance of LP-HTA Algorithm

In this section, we will investigate the performance of our *LP-HTA* algorithm by comparing with *HGOS* and two baseline algorithms, *i.e. AllToC* and *AllOffload*, where *HGOS* denotes the Heuristic Greedy Offloading Scheme proposed in [12],



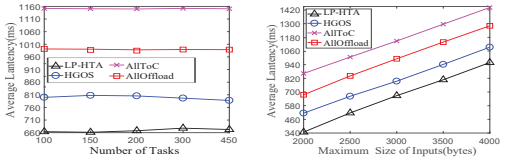
(a) Affected by Number of Tasks (b) Affected by Size of Input Data

Fig. 2: Energy Cost of *LP-HTA*



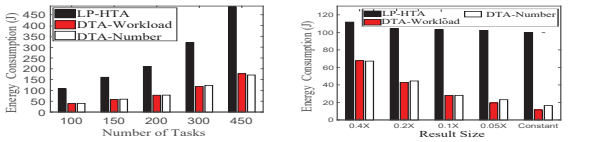
(a) Unsatisfied Task Rate

Fig. 3: The Rate of Unsatisfied Task



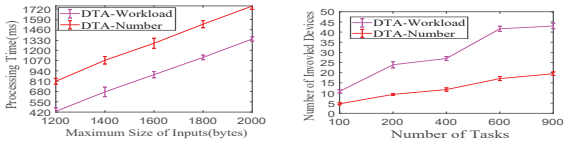
(a) Affected by Number of Tasks (b) Affected by Size of Input Data

Fig. 4: Latency of *LP-HTA*



(a) Affected by Number of Tasks (b) Affected by Size of Task Result

Fig. 5: Energy Cost of *LP-HTA*, *DTA-Workload*, *DTA-Number*



(a) Processing Time (b) Number of Invovled Devices

Fig. 6: The comparison of *DTA-Workload* and *DTA-Number*

AllToC and *AllOffload* means that all the tasks are offloaded to the cloud and to the base stations and the cloud, respectively. The reasons of choosing the above three methods are as follows. Firstly, we compare *LP-HTA* with *HGOS* [12] since it is the latest and efficient task assignment algorithms in a MEC system. Secondly, *AllToC* and *AllOffload* are selection for comparison because they are most classical and representative methods when the computation abilities of the mobile devices are not considered. The size of the external data is set to 0 to 0.5 times the local data.

The first group of experiments investigate the impact of the number of tasks on the energy cost of *LP-HTA*. In the experiments, the energy consumed by *LP-HTA*, *HGOS*, *AllToC* and *AllOffload* were calculated while the number of the tasks in a MEC system increased from 100 to 450, and the maximum input data size for each task is 3000kb. According to the results illustrated in Fig.2(a), the energy consumption of *LP-HTA* is much smaller than that of *AllToC* and *AllOffload* as the computation abilities of the mobile edges, including mobile devices and the base stations are sufficiently used. Furthermore, the energy consumed by our *LP-HTA* is also lower than that consumed by *HGOS*. Since the constraints of each mobile edge is fully considered during task offloading, the task assignment results of our *LP-HTA* is more reasonable and efficient comparing with *HGOS*, and thus much energy will be saved. Finally, we find that the energy consumed by *LP-HTA* increases slowly with the increment of the tasks, which means that our *LP-HTA* algorithm is very suitable to process the computation intensive tasks.

The second group of experiments observe the energy consumption of *LP-HTA* affected by the input data size. In the experiments, the energy cost of *LP-HTA*, *HGOS*,*AllToC* and *AllOffload* were calculated while the maximum input data size of each task increased form 1000 kb to 5000 kb, and the number of total tasks is 100. The results presented in Fig.2(b) indicate that the energy consumption of *LP-HTA* is also the smallest even when the amount of data required to be processed increases, which means that our *LP-HTA* is also suitable to deal with the data-intensive tasks.

The third group of experiments is to investigate the unsatisfied task rate of all the task assignment algorithms. The unsatisfied task rate is denoted by the proportion of the tasks whose delay constraints can not be met among all the tasks. Since the unsatisfied task rate of *AllToC* is quite high due to the large latencies of transmitting all the task to the remote cloud, we only compare *HGOS*, *AllOffload* and our *LP-HTA* in the experiments. As shows in Fig.3, the unsatisfied task rate of the three algorithms were calculated when the number of tasks increased form 100 to 450. The results shows that the unsatisfied task rate of our *LP-HTA* is quite small comparing with *HGOS* and *AllOffload*. Since we take the delay constraint of each task into account during task assignment, only a small portion of task are unsatisfied. Therefore, our *LP-HTA* algorithms are more efficient and effective to process the tasks that has strict delay constraints. Furthermore, although the energy cost of *HGOS* are close to *LP-HTA* based on Fig.2, it has quite large unsatisfied task rate, so that the performance of *HGOS* is not good enough comparing with *LP-HTA* as large amount of deadline constraints cannot be met.

The fourth group of experiments is to investigate the impact of the amount of tasks on the latency of *LP-HTA*. In the experiments, the average latency of *LP-HTA*, *HGOS*, *AllToC* and *AllOffload* were calculated while the number of tasks grows from 100 to 450, and the maximum size of input data is 3000kb. From results in Fig.4(a), we can see that the average latency of *LP-HTA* is extremely small comparing with *AllToC* and *AllOffload* since the mobile devices are fully used and some of task are processed immediately by *LP-HTA*. Moreover, the average latency of *LP-HTA* is also much lower than that of *HGOS* as *LP-HTA* considers both of task deadline constraints and the limited computing resources of each mobile edge sufficiently during task assignment. Thus,LP-HTA is efficient to process the latency-intensive tasks in MEC systems.

The fifth group of experiments is used to analyze the

relationship between the latency of *LP-HTA* and the size of input data. The average latency of *LP-HTA*, *HGOS*, *AllToC* and *AllOffload* were calculated while the maximum size of the input data for each task varies from 1000kb to 5000kb, and the number of total tasks is 100. Based on the results in Fig.4(b), the average latency of *LP-HTA* is still the smallest comparing with *HGOS*, *AllToC* and *AllOffload*. However, the advantage of *LP-HTA* on latency is not so much obvious comparing with *HGOS*, which is because more tasks will exceed the processing abilities of mobile devices with the increment of input data size for both *LP-HTA* and *HGOS*, so that they are offloaded to base station or to the cloud, and the latency will enlarge accordingly. In spite of that, the *LP-HTA* still utilize the processing abilities of the mobile edges, including mobile devices and base stations, as much as possible. Meanwhile, the *LP-HTA* also guarantees that much more deadline constraints of the tasks are satisfied even when the average latency is enlarged according to the results in Fig.3. Such results also verified that our *LP-HTA* is efficient and effective to deal with the data-intensive tasks.

C. Performance of DTA

The section will evaluate of the performance of Divisible Task Assignment algorithms, and we use *DTA-Workload* and *DTA-Number* to stand for the algorithms proposed in Sections IV(a) and IV(b), respectively. In the following experiments, the comparisons among *DTA-Workload*, *DTA-Number* and *LP-HTA* are considered since *LP-HTA* has the best performance according to the analysis in the above section

The first group of experiments compares the energy cost by *DTA-Workload*,*DTA-Number* and *LP-HTA* for processing the tasks with different amount. The energy cost of three methods were computed while the number of tasks increased form 100 to 450, the maximum size of input data was 3000kb for each task, and the ratio of the result size to the amount of input data was 0.2. Fig.5(a) shows that the energy cost of *DTA-Workload* and *DTA-Number* are quite small especially when the amount of tasks increases. Since more raw data are avoided to transmit by *DTA-Workload* and *DTA-Number* when the amount of tasks increases, lots of energy will be saved.

The second group of experiments observes the energy cost of the above three algorithms while the result size is varying. In the experiments, the result size is set to be $0.4X$, $0.2X$, $0.1X$, $0.05X$ and constant, where X is the size of input data. The number of total tasks in a MEC system was 100, and the maximum size of input data was 3000kb for each task. Fig.5(a) presents the energy cost of three algorithm for different result size. It shows that *DTA-Workload* and *DTA-Number* consume extremely small energy when the result size decreases. Considering that the sizes of final result and partial results are alway much smaller than that of raw data, *DTA-Workload* and *DTA-Number* will achieve high performance for processing many kinds of tasks.

The last two groups of experiments are going to compare *DTA-Workload* and *DTA-Number* separately. In Fig.6(a), the processing time of *DTA-Workload* and *DTA-Number* were calculated while the maximum size of input data increased from 1200kb to 2000kb, and the number of total tasks was 200. In Fig.6(b), the numbers of involved mobile devices for *DTA-Workload* and *DTA-Number* were counted while the number of total tasks varied form 100 to 900, and the maximum size of input data was 2000kb for each task. The results in Fig.6(a) and Fig.6(b) show that the processing time of *DTA-Workload* is much smaller since it divides the input data as uniform as possible, and the total processing workload are balanced, so that the processing time is shorten accordingly. On the other hand, smaller amount of mobile devices are required by *DTA-Number*, so that the energy of majority mobile devices is save. Therefore, *DTA-Workload* and *DTA-Number* are suitable for different situations, where the users can select one of them flexibly according to the applications.

VI. Related Works

The early ones that studied the task assignment in a MEC system mainly focus on serving a single user. To determine the offloading strategies of the tasks proposed by single users, a Binary offloading algorithms is given in [6]. To take advantage of parallel computing, a partial offloading problem were studied in [25], and an approximation algorithm with $(1+\epsilon)$ ratio bound was given to solve it. In [26], a variable-substitution technique was provided to jointly optimize the offloading ratio, transmission power and CPU-cycle frequency. These works has high performance to deal with the tasks raised by a single user, however, the situation considered by them seems a little simple. Meanwhile, they did not consider the data distribution during determine the offloading strategies.

Considering that a MEC system always serve for multiple users, [8] has formulated the computation offloading problem of multiple users via a single wireless access point as a computation offloading game, and proposed a decentralized mechanism that can achieve the Nash equilibrium to solve it. In [27], the author try to optimize the uplink and downlink scheduling using queuing theory. In [28], a decomposition algorithm was given to control the computation offloading selection, clock frequency control and transmission power allocation iteratively. The above algorithms are efficient to process the tasks raised by multiple users, however, since they were designed for a MEC system with a single base station, the cooperations among different base stations were not considered sufficiently. Furthermore, these algorithms ignored the data distribution either during task assignment. Besides, [11] introduced the centralized and distributed Greedy Maximal Scheduling algorithms to solve the computation offloading problem for multiple users with multiple tasks. However, these algorithms did not consider the data sharing during the task assignment, and ignored the delay constraint of each task as well, so that they are not suitable to solve the problem discussed in this paper.

To enhance the cooperations among different base stations, a new cooperation scheme among base stations was given in [29], the author tried to reduce the response latency through caching the results of the popular computation tasks. [30]

proposed a game-theoretic algorithm to maximize revenues from all applications. All above works fully utilized the cooperations among base stations. However, these algorithms ignored the distribution of input data as well, and they are not suitable to the Data-Shared MEC systems. [31] provided a payment-based incentive mechanism. Such mechanism supports sharing computation resources among base stations and the authors also established a social trust network to manage the security risks among them, however, it still has the above problems that we pointed out. Moreover, the constraints on processing ability of each base station are not reasonable enough since they only consider the task arrival rate and didn't take the data and resources occupied by each task into account.

More recently, the architecture of a MEC system which involves the three levels and serves for multiple users are sufficiently investigated by [12] and [13]. In [12], a heuristic greedy offloading scheme was given for ultra dense networks. In [13], a distributed computation offloading algorithm that can achieve the Nash equilibrium was provided. Both of these works are efficient for a MEC system, however, they did not considered data distribution and the delay constraints of tasks during task assignment, thus they are not able to assign the tasks in a Data-Shared MEC system as well. Besides, the processing abilities of the Cloud are not fully utilized, either.

VII. Conclusion

This paper studies the task assignment problem in a Data-Share MEC system. We firstly distinguish the computation tasks as the holistic tasks and the divisible tasks. For the holistic tasks, the HTA problem is proposed and proved to be NP-complete. Finally, a linear programming based approximation algorithm is proposed. For the divisible tasks, the key problem is to rearrange the tasks according to the data distribution. Two approximation algorithms with $O(n)$ complexities are provided to solve it under different optimization goals. Both of theoretical analysis and experiments show that all proposed algorithms achieve high performance.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant Nos. 61632010, U1509216.

References

[1] S. Cheng, Y. Li, Z. Tian, W. Cheng, and X. Cheng, "A model for integrating heterogeneous sensory data in iot systems," *Computer Networks*, vol. 150, pp. 1–14, 2019.

[2] S. Cheng, Z. Cai, and J. Li, "Curve query processing in wireless sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 11, pp. 5198–5209, 2015.

[3] S. Cheng, Z. Cai, J. Li, and H. Gao, "Extracting kernel dataset from big sensory data in wireless sensor networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 4, pp. 813–827, 2017.

[4] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computingła key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[5] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in *International Conference on Intelligent Systems and Control*, 2016.

[6] W. Zhang, Y. Wen, K. Guan, K. Dan, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE TWC*, vol. 12, no. 9, pp. 4569–4581, 2013.

[7] K. Kumar, J. Liu, Y. H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Networks & Applications*, vol. 18, no. 1, pp. 129–140, 2013.

[8] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *Parallel & Distributed Systems IEEE Transactions on*, vol. 26, no. 4, pp. 974–983, 2014.

[9] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2016.

[10] 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 TCOM*, vol. 65, no. 8, pp. 3571–3584, 2017.

[11] W. Chen, D. Wang, and K. Li, "Multi-user multi-task computation offloading in green mobile edge cloud computing," *IEEE Transactions on Services Computing*, 2018.

[12] H. Guo, J. Liu, and J. Zhang, "Computation offloading for multi-access mobile edge computing in ultra-dense networks," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 14–19, 2018.

[13] L. Tang and S. He, "Multi-user computation offloading in mobile edge computing: A behavioral perspective," *IEEE Network*, vol. 32, no. 1, pp. 48–53, 2018.

[14] T. D. Burd and R. W. Brodersen, *Processor design for portable systems*. Kluwer Academic Publishers, 1996.

[15] J. Oueis, E. Calvanese-Strinati, A. De Domenico, and S. Barbarossa, "On the impact of backhaul network on distributed cloud computing," in *Wireless Communications and NETWORKING Conference Workshops*, 2014, pp. 12–17.

[16] "Amazon Cloud." [Online]. Available: http://www.cloudping.info/

[17] N. Karmarkar, "A new polynomial-time algorithm for linear programming," *Combinatorica*, vol. 4, no. 4, pp. 373–395, 1984.

[18] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Annual Cryptology Conference*, 2010, pp. 465–482.

[19] Y. Huang, X. Song, Y. Fan, Y. Yang, and X. Li, "Fair caching algorithms for peer data sharing in pervasive edge computing environments," in *ICDCS*, 2017.

[20] C. H. Papadimitriou, "On the complexity of integer programming," *Journal of the Acm*, vol. 28, no. 4, pp. 765–768, 1981.

[21] U. Feige, *A threshold of ln n for approximating set cover*. ACM, 1998.

[22] O. Munoz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 10, pp. 4738–4755, 2015.

[23] A. E. Eshratifar and M. Pedram, "Energy and performance efficient computation offloading for deep neural networks in a mobile cloud computing environment," in *Proceedings of the 2018 on Great Lakes Symposium on VLSI*, 2018, pp. 111–116.

[24] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4g lte networks," in *ACM Mobisys*, 2012, pp. 225–238.

[25] Y. H. Kao, B. Krishnamachari, M. R. Ra, and B. Fan, "Hermes: Latency optimal task assignment for resource-constrained mobile computing," in *IEEE Conference on Computer Communications*, 2015, pp. 1894–1902.

[26] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE TCOM*, vol. 64, no. 10, pp. 4268–4282, 2016.

[27] M. Molina, O. Muoz, A. Pascual-Iserte, and J. Vidal, "Joint scheduling of communication and computation resources in multiuser wireless application offloading," in *IEEE International Symposium on Personal, Indoor, and Mobile Radio Communication*, 2015, pp. 1093–1098.

[28] S. Guo, B. Xiao, Y. Yang, and Y. Yang, "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing," in *INFOCOM*, 2016, pp. 1–9.

[29] M. S. Elbamby, M. Bennis, and W. Saad, "Proactive edge computing in latency-constrained fog networks," in *European Conference on Networks and Communications*, 2017, pp. 1–6.

[30] R. Yu, J. Ding, S. Maharjan, S. Gjessing, Y. Zhang, and D. H. K. Tsang, "Decentralized and optimal resource cooperation in geo-distributed mobile cloud computing," *IEEE Transactions on Emerging Topics in Computing*, vol. PP, no. 99, pp. 1–1, 2018.

[31] L. Chen and J. Xu, "Socially trusted collaborative edge computing in ultra dense networks," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, 2017, p. 9.