Energy-Efficient Computation Offloading for Mobile Edge Networks: A Graph Theory Approach

Junlin Liu[1], Xing Zhang[1,*], Xin Li[1], Yongdong Zhu[2]
[1]Wireless Signal Processing and Network Laboratory
Beijing University of Posts and Telecommunications, Beijing 100876, China
[2]Zhejiang Lab, Zhejiang, China
[*]E-mail: zhangx@ieee.org

***Abstract*—Computation offloading is deemed as a promising technology for ensuring user experience and realizing load balance. However, it is challenging to utilize network resources efficiently due to lack of collaborative management ability of isolated edge devices. In this paper, we propose a computation offloading scheme to minimize the total energy consumption for mobile edge networks. Specifically, we formulate the problem as a mixed integer non-linear program and transform it to two sub-problems, namely task offloading sub-problem and resource allocation sub-problem. We leverage the improved graph theory algorithm to figure out the computation offloading sub-problem, and use the binary search algorithm along with priority assignment to solve the resource allocation sub-problem. The numerical results reveal that maximum-alternative-differences-first Gale Sherply (MADF-GS) algorithm performs the best among all GS algorithms, which combines low time complexity with excellent performance, and it saves at least 66.7% energy consumption in comparison with the conventional scheme.**

***Index Terms*—Mobile edge computing, computation offloading, resource allocation, graph theory**

I. Introduction

The rapid development of Internet of Things (IoT) technology gives birth to massive advanced applications with delay-sensitive and computation-intensive tasks. However, mobile cloud computing (MCC) is impotent to make rapid decisions in many application scenarios. In order to achieve lower energy consumption, shorter delay, greater security, higher reliability and lower bandwidth requirements, mobile edge computing (MEC) emerges and gains its popularity. By deploying servers at the edge, the service response and execution efficiency have been improved significantly [1]. As the essential part of mobile edge network, computation offloading has made a great contribution to maximize resource utilization. Based on this technology, we can dynamically allocate resources and schedule tasks to meet users' demands.

Compared with mobile cloud computing, resource constraints in MEC urge scholars to study task offloading and resource allocation to maximize the effective utilization of network resources. In [2], C. You *et al.* studied computation offloading in TDMA and OFDMA system, comprehensively considered the priority function design in the case of limited resource and infinite resources; S. Mu *et al.* proposed three distributed algorithms to solve the task offloading problem in multi-user multi-server scenario [3]. In [4], a framework for computation offloading in SD-UDN system was designed, and the offloading decision variables were approximated as continuous functions to solve. J. Zhang *et al.* exploited computation replication to mitigate interference and fading during task offloading, and researched the tradeoff between communication and computation resource [5]; In [6], the author modeled the interaction between devices and servers as a Stackelberg game, and proposed an efficient decentralized algorithm for computing Nash equilibrium. H. A. Alameddine *et al.* considered the edge-to-edge delay, and proposed a novel and ready-to-use algorithm based on Benders decomposition to solve MINLP problem [7]. Furthermore, some scholars have introduced deep reinforcement learning (DRL) theory into computation offloading, and designed adaptive strategies based on DQN, DDPG and etcetera [8]–[11].

However, the pioneering work did not consider the joint optimization of continuous computation offloading and limited resource allocation in mobile edge network. In addition, we may have to assign the offloading proportion and allocated resource carefully to meet the resource constraint. In this paper, we explore the computation offloading problem in container-cluster based mobile edge network with the consideration of battery capacity, resource constraint and users' demands. The main contributions of this paper can be formulated as follows:

- We use binary search algorithm to obtain the optimal solution of offloading proportion and computation resource allocation for mobile edge networks, and design a priority-based assignment scheme with limited resource.
- We propose two task offloading algorithms based on graph theory, and verify that MADF-GS has the optimal comprehensive performance. Specifically, Kuhn Munkres (KM) algorithm is capable to save the most energy consumption, while GS algorithm can greatly save the running time at the expense of certain performance.
- We analyze the algorithms' performance with different number of UEs, and solve the optimal discount factor γ for GS algorithms.

The remainder of this paper is organized as follows. In Section II, we describe the system model and problem formulation. In Section III, we subdivide the problem to two sub-problems and design the algorithm. In Section IV, we evaluate the proposed algorithm with numerical results. Finally, conclusion and future work are given in Section V.

II. System Model and Problem Formulation

A. Network Model

As is illustrated in Figure 1, we consider a mobile edge network with multiple UEs and MEC servers. Each server integrates a base station (BS) and deploys a set of containers, and we assume each UE access to the nearest BS. The container is the smallest units to execute a task, and it shares the resource of the server. We denote F as the computation resource of MEC servers, measured by CPU frequency; Q as the storage resource, measured by the size of hard discs.

The system operates in a time-slotted structure. In each time slot, the UE needs to process a delay-sensitive and computing-intensive task, which can be offloaded to container for collaboration. We assume each UE can only offload its task to at most one container in MEC server, and each container can execute at most one task simultaneously.

Let $\mathcal{M} \triangleq \{1, 2, \cdots, M\}$ represent the set of UEs, and $\mathcal{N} \triangleq \{1, 2, \cdots, N\}$ denote the set of containers in MEC servers. The maximum battery capacity of UE m is $E_m^{\max}$. We denote $f_m, f_n, \kappa_m, \kappa_n$ as the computing speed (i.e. CPU frequency) and energy efficiency coefficient of UE m and container n, respectively.

The task for UE m is described as a tuple of three parameters $< W_m, D_m, T_m >$, where W_m indicates the workload, measured in the number of CPU cycles. D_m represents the size of input data. Similar with [12], we assume D_m is proportional to task's workload, i.e. $D_m = \alpha W_m$, where $\alpha (0 \leq \alpha \leq 1)$ is determined by the nature of the task. T_m is the deadline. The task will be rejected if it can't process within its deadline.

We introduce the task offloading pair (m, n) to represent the task of UE m is offloaded to container n. We assume the task can be partitioned continuously, which indicates the UE can offload any proportion of the task to the container. Denote $\lambda_{m,n} (0 \leq \lambda_{m,n} \leq 1)$ as the offloading proportion of the task, and then the rest workload $(1-\lambda_{m,n})W_m$ will process locally.

B. Time Latency

1) Local Computing: The time latency of local computing is equivalent to the local computation delay, which is given by:

$$t_{m,l} = \frac{(1 - \lambda_{m,n}) W_m}{f_m}. \tag{1}$$

2) Task Offloading: For UE m, the time latency of task offloading mainly consists of upload delay $t_{m,n}^{\text{up}}$ and computation delay $t_{m,n}^{\text{comp}}$. Similar to many studies such as [13], the download delay is ignored because the output data is always much smaller than the initial size of the task. Additionally, the edge-to-edge delay is negligible because of the excellent quality of wireless links between servers.

Denote h_m as the channel coefficient, which is constant during offloading duration, and P_m the transmission power of UE m. The transmission rate between UE m and container n can be expressed as:

$$R_{m,n} = B_m \log_2 \left(1 + \frac{P_m h_m^2 d^{-\zeta}}{\sigma_n^2}\right), \tag{2}$$

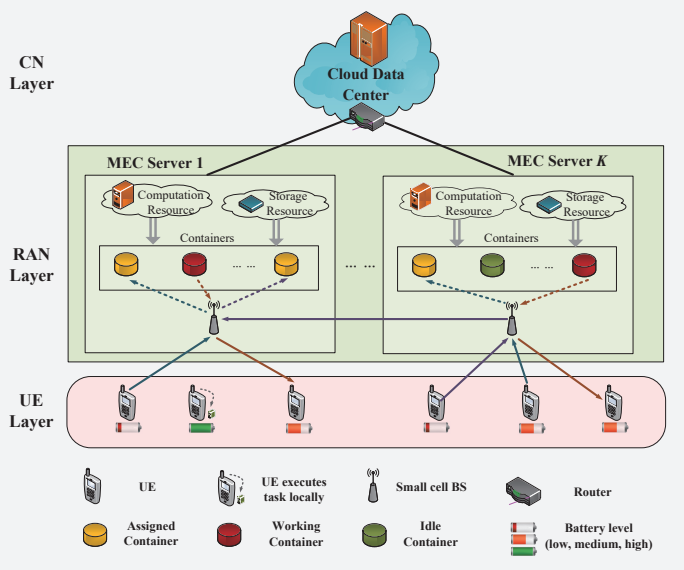


Fig. 1. Computation offloading model for mobile edge network. Each UE can execute task locally, offload task to the access server, and offload task to other servers. The upload link is used for offloading tasks, and the download link is used for transmitting the results. The cloud data center is responsible for collecting and analyzing the data and logs from servers.

where B_m denotes the uploading bandwidth, d is the distance between the UE and the container, ζ denotes the path loss exponent, σ_n^2 is the variance of noise power.

Then, the task offloading delay can be written as:

$$t_{m,n} = t_{m,n}^{\text{up}} + t_{m,n}^{\text{comp}} = \frac{\lambda_{m,n} D_m}{R_{m,n}} + \frac{\lambda_{m,n} W_m}{f_n}. \tag{3}$$

Due to the parallel computing at the UE and container, the total time latency for UE m is determined by the larger one between $t_{m,l}$ and $t_{m,n}$, which is given by

$$t_m = \max\{t_{m,l}, t_{m,n}\}. \tag{4}$$

C. Energy Consumption

1) Local Computing: For UE m, the energy consumption per CPU cycles is $\kappa_m f_m^2$, and then the energy consumption for local computing can be expressed as:

$$E_{m,l} = (1 - \lambda_{m,n}) \kappa_m f_m^2 W_m. \tag{5}$$

We introduce E_m to represent UE m executes its total task locally, i.e. $\lambda_{m,n} \equiv 0 (\forall n \in \mathcal{N})$, which is given by

$$E_m = \kappa_m f_m^2 W_m. \tag{6}$$

2) Task Offloading: For UE m, the energy consumption of task offloading consists of uploading cost $E_{m,n}^{\text{up}}$ and computing cost $E_{m,n}^{\text{comp}}$, which can be deduced by

$$E_{m,n} = E_{m,n}^{\text{up}} + E_{m,n}^{\text{comp}} = P_m t_{m,n}^{\text{up}} + \lambda_{m,n} \kappa_n f_n^2 W_m. \tag{7}$$

Therefore, the total energy consumption for UE m is the sum of $E_{m,l}$ and $E_{m,n}$, which is given by

$$\begin{aligned} E_m &= E_{m,l} + E_{m,n} \\ &= (1 - \lambda_{m,n}) \kappa_m f_m^2 W_m + P_m t_{m,n}^{\text{up}} + \lambda_{m,n} \kappa_n f_n^2 W_m. \end{aligned} \tag{8}$$

D. Problem Formulation

We define the binary decision variable $x_{m,n}$ to represent whether the task of UE m is offloaded to container $n(x_{m,n} = 1)$ or not$(x_{m,n} = 0)$. In order to take the local computing into consideration, we use $\mathcal{N}'$ to represent the set of generalized containers, which includes both containers and UEs, i.e. $\mathcal{N}' = \mathcal{N} \cup \mathcal{M}$. We specify $x_{m,N+m} = 0$ when UE m process its entire task locally.

For convenience of description, we introduce the auxiliary variable $\hat{f}_{m,n}$ to represent the allocated computation resource of container n when serving UE m. The relationship between f_n and $\hat{f}_{m,n}$ can be expressed as $f_n = \sum_{m\in M} x_{m,n}\hat{f}_{m,n}$.

Under the constraints of battery capacity, resource constraint and users' demands, the computation offloading problem is formulated as follows:

$$\mathbf{P}: \min_{\boldsymbol{X},\boldsymbol{\lambda},\hat{\boldsymbol{f}}} \sum_{m\in\mathcal{M}}\sum_{n\in N'} x_{m,n}\left(E_{m,l}+E_{m,n}\right) \tag{9a}$$

$$\text{s.t.} \sum_{n\in\mathcal{N}'} x_{m,n}t_m \le T_m, \forall m\in\mathcal{M}, \tag{9b}$$

$$E_{m,l} \le E_m^{\max}, \forall m\in\mathcal{M}, \tag{9c}$$

$$\sum_{m\in\mathcal{M}}\sum_{n\in\mathcal{N}} x_{m,n}\hat{f}_{m,n} \le F, \tag{9d}$$

$$\sum_{m\in\mathcal{M}}\sum_{n\in\mathcal{N}} \alpha x_{m,n}\lambda_{m,n}W_m \le Q, \tag{9e}$$

$$\sum_{n\in\mathcal{N}'} x_{m,n} + x_{m,N+m} = 1, \forall m\in\mathcal{M}, \tag{9f}$$

$$\sum_{m\in\mathcal{M}} x_{m,n} \le 1, \forall n\in N, \tag{9g}$$

where $\boldsymbol{X} \triangleq \{x_{m,n}\}$, $\boldsymbol{\lambda} \triangleq \{\lambda_{m,n}\}$, $\hat{\boldsymbol{f}}^* \triangleq \left\{\hat{f}_{m,n}\right\}$ indicates the task offloading decision, offloading proportion and allocated computation resource, respectively. The computation offloading problem is a non-convex mixed-integer non-linear program (MINLP), which can be proved NP-hard according to [14].

III. Problem Transformation and Computation Offloading Scheme

In this section, we propose an efficient computation offloading scheme for aforementioned problem $\mathbf{P}$. Considering the difficulty of solving problem directly, we decompose the problem into two sub-problems: 1) $\mathbf{P_T}$, task offloading sub-problem; 2) $\mathbf{P_R}$, resource allocation sub-problem.

We firstly define the objective function as follows:

$$G(\boldsymbol{X},\boldsymbol{\lambda},\hat{\boldsymbol{f}}) = \sum_{m\in\mathcal{M}}\sum_{n\in N'} x_{m,n}E_{m,n}. \tag{10}$$

The problem $\mathbf{P}$ will be solved in three phases. **Firstly**, with given task offloading decision $\boldsymbol{X^0}$, i.e. when $x_{m,n}$ is fixed, we find the optimal offloading proportion $\boldsymbol{\lambda^*}$ and computation resource $\hat{\boldsymbol{f}}^*$. **Secondly**, we solve the optimal task offloading decision $\boldsymbol{X}^*$ with $\boldsymbol{\lambda}^*$ and $\hat{\boldsymbol{f}}^*$. **Finally**, we examine whether the sum of allocated resource exceed the resource capacity, and adjust the allocated resource by priority.

A. Optimal Offloading Proportion and Resource Allocation

In this subsection, we assume the network resources are sufficient. Given task offloading decision $\boldsymbol{X^0}$, the optimization problem can be reformulated as

$$\mathbf{P_R}: \min\ G(\boldsymbol{\lambda},\hat{\boldsymbol{f}}) \quad \text{s.t. (9b), (9c).} \tag{11}$$

The optimal value of $(\lambda, \hat{f})$ is independent for different task offloading pair (m, n). Therefore, we can decouple the problem above into $M \times N$ sub-problems:

$$\mathbf{P_R 1}: \min\ G(\lambda,\hat{f}) \tag{12a}$$

$$\text{s.t. } \lambda - \frac{R_m\hat{f}T_m}{W_m(R_m+\alpha\hat{f})} \le 0, \tag{12b}$$

$$\lambda_{\min} \le \lambda \le 1, \tag{12c}$$

$$0 \le f, \tag{12d}$$

where $\lambda_{\min}$ is derived by (9b) and (9c), which is written as

$$\lambda_{\min} = \max\left\{1-\frac{T_m f_m}{W_m}, 1-\frac{E_m^{\max}}{\kappa_m f_m^2 W_m}, 0\right\}. \tag{13}$$

The Lagrange function of $\mathbf{P_R 1}$ can be obtained as follows:

$$L(\lambda,\hat{f},\beta) = G(\lambda,\hat{f}) + \beta\left[\lambda - \frac{R_m\hat{f}T_m}{W_m(R_m+\alpha\hat{f})}\right]. \tag{14}$$

By using KKT conditions, we can find that the optimal value of $\hat{f}^*$ is uniquely determined by λ^*. Actually, $\hat{f}^*$ increases monotonically w.r.t. λ^*. We can denote $\hat{f}^*$ as a function of λ^*, which is given by

$$\hat{f}^* = \Gamma(\lambda^*) = \frac{\lambda^* W_m R_m}{R_m T_m - \alpha\lambda^* W_m}. \tag{15}$$

We substitute (15) into $L(\lambda, \hat{f}, \beta)$ and then the derivative w.r.t λ can be rewritten as:

$$\frac{\partial L}{\partial\lambda} = -\kappa_m f_m^2 W_m + \alpha\frac{P_m}{R_{m,n}}W_m + \frac{\kappa_n W_m R_{m,n}^2}{\alpha^2}\cdot\varphi(\lambda), \tag{16}$$

where $\varphi(\lambda) = \frac{\lambda^2(\lambda-3k)}{(\lambda-k)^3}$, and $k = \frac{T_m R_{m,n}}{\alpha W_m} > 1$.

It can be verified that L is a convex function of λ, and $\frac{\partial L}{\partial\lambda}$ increases monotonically along with λ in interval $(0, 1)$. Therefore, we can adopt the binary search method to find the optimal λ^* with given interval $[\lambda_{\min}, 1]$ and accuracy indicator ϵ, then the optimal f^* can be obtained according to (15).

B. Optimal Task Offloading Policy

In this subsection, we transform the task offloading decision problem into a weighted bipartite graph matching problem, and use the graph theory algorithm to solve it efficiently. A weighted bipartite graph is generally defined as $G = (V, E, \omega)$, where V is the set of vertices, including two disjoint vertex subsets V_M, V_N. E is the set of undirected edges, each connecting a vertex in V_M and a vertex in V_N. ω is the weight set of corresponding edges. We take $\mathcal{M}$ and $\mathcal{N}'$ as two disjoint vertex sets, and the weight of the edge is equal to the opposite value of the pair's energy consumption.

We can use KM algorithm to solve this problem, which has been proved to find the optimal matching of complete bipartite graph in polynomial time. We need to add $N - M$ virtual vertices in V_M as a step of preprocessing. Given M UEs and N containers, the time complexity of KM algorithm is $O((M + N)^3)$. In large-scale mobile edge network, the running time of KM algorithm will be non-negligible. Therefore, we need to find a strategy to minimize the impact of algorithm execution latency.

C. Sub-optimal Task Offloading Policy

In this subsection, we design a sub-optimal task offloading policy inspired by the classical stable matching algorithm based on graph theory, namely GS (Gale Shapley) algorithm. Note that the definition of *applicants*, *colleges* and *stable matching* can be seen in [15].

The main idea of GS algorithm is to repeat the process below until achieving the *stable matching*: the *applicant* sends request to the *college* by preference, and the *college* accepts or rejects the *applicant* by preference. Therefore, the main factor determining the performance of GS algorithm is the preference rules (or rather, **ranking function**) of *applicants* and *colleges*.

We take the set of UEs and containers as *applicants* and *colleges*, respectively. The UE's ranking function is denoted as $P(n)$, and the container's as $Q(m)$. The UE's ranking function P can be set to the negative energy consumption since UE expects to choose the container that consumes the least energy to process the task, written by

$$P^*(n) = \arg\max_{n \in \mathcal{N}} \left(-|E_{m,n}|\right). \tag{17}$$

The container's ranking function Q should comprehensively consider the gap between the energy consumption of executing the UE's task and that of the next (or even the next k) container after rejecting the current UE, so as to ensure that the matching process does not fall into the local optimal (that is, not short-sighted) and achieve the global optimal (namely long-term consideration). We design a series of practicable ranking functions for the container as follows, and the abbreviations of these functions are used to represent the algorithm.

1) Original GS(ORG-GS): The negative energy consumption of UE's task.

$$Q^*(m) = \arg\max_{m \in \mathcal{M}} \left(-|E_{m,n}|\right). \tag{18}$$

2) Maximum Local Difference First(MLDF-GS): The energy consumption difference between offloading to the container and executing locally.

$$Q^*(m) = \arg\max_{m \in \mathcal{M}} \left(|E_{m,N+m} - E_{m,n}|\right). \tag{19}$$

3) Maximum Sub-optimal Difference First(MSDF-GS): The energy consumption difference between offloading to the optimal container and offloading to the sub-optimal container.

$$Q^*(m) = \arg\max_{m \in \mathcal{M}} \left(|E_{m,n^{(1)}} - E_{m,n}|\right). \tag{20}$$

where n^i indicates the i-th optional container for UE m.

Algorithm 1: General Form for GS Algorithm

Input: Energy consumption matrix E;
UEs set: $\mathcal{M} = \{1, 2, \cdots, M\}$;
containers set: $\mathcal{N} = \{1, 2, \cdots, N\}$;
Output: Task offloading decision $\boldsymbol{X}^*$

Initialize $\boldsymbol{X}^* = \boldsymbol{O}, E' = E$;
repeat
  Randomly choose m from $\mathcal{M}$;
  Find n^* according to (17) by replacing E with E';
  $E'_{m,n^*} \leftarrow \infty$;
  if $E_{m,n} > E_{m,N+m}$ **then**
    remove m from $\mathcal{M}$;
    $x_{m,N+m} \leftarrow 1$;
  else
    if $n \in \mathcal{N}$ **then**
      remove m from $\mathcal{M}$, remove n from $\mathcal{N}$;
      $x_{m,n} \leftarrow 1$
    else
      find m' coupled with n in *match*;
      compute $Q(m), Q(m')$ by (18)$\sim$(22);
      if $Q(m) > Q(m')$ **then**
        remove m from $\mathcal{M}$, insert m' to $\mathcal{M}$;
        $x_{m',n} \leftarrow 0,\ x_{m,n} \leftarrow 1$;
until $\mathcal{M} = \varnothing$;
return $\boldsymbol{X}^*$

4) Maximum Alternative Differences First(MADF-GS): The weighted sum of the energy consumption difference between optimal container and offloading to other optional containers.

$$Q^*(m) = \arg\max_{m \in \mathcal{M}} \left(\sum_k \gamma^{k-1} |E_{m,n^{(k)}} - E_{m,n}|\right). \tag{21}$$

where $0 < \gamma < 1$ is a discount factor.

5) Maximum Differentials First(MDF-GS): The weighted differential sum of energy consumption of offloading to its optional containers.

$$Q^*(m) = \arg\max_{m \in \mathcal{M}} \left(\sum_k \gamma^{k-1} |E_{m,n^{(k)}} - E_{m,n^{(k-1)}}|\right). \tag{22}$$

We propose a general form of GS algorithm in Algorithm 1. Given M UEs and N containers, the time complexity of GS algorithms is $O(MN)$.

D. Priority-based Resource Allocation Adjustment

In this subsection, we design a priority-based resource allocation adjustment strategy. The offloading proportion and computation resource will reduce at the same time according to (15) to meet the task's deadline constraint when exceeding the resource capacity. To comprehensively evaluate the resource bottleneck between storage and computation resource, we define the priority function as follows:

Algorithm 2: Computation Offloading Algorithm

Input: $\kappa_m, \kappa_n, \alpha, W_m, f_m, T_m, P_m, R_{m,n}, E_m^{\max}, \epsilon$
Output: Decision indicator: $\boldsymbol{X}^*, \boldsymbol{\lambda}^*, \boldsymbol{f}^*$

Find $\boldsymbol{\lambda}^*, \hat{\boldsymbol{f}}^*$ according to Binary Search Algorithm;
Find $\boldsymbol{X}^*$ according to Algorithm 1 or KM Algorithm;
Initialize $Q_{\text{sum}} = \sum_m \sum_n (\alpha x_{m,n} \lambda_{m,n} W_m)$, and $F_{\text{sum}} = \sum_m \sum_n (x_{m,n} \hat{f}_{m,n})$;
if $Q_{\text{sum}} > Q$ or $F_{\text{sum}} > F$ **then**
  Compute $\phi_{m,n}$ according to (23);
  Sort the task offloading pair (m, n) in ascending order based on the value of $\phi_{m,n}$ as Φ;
  for pair (m, n) with the lowest priority in Φ **do**
    $Q_0 \leftarrow \alpha \lambda_{m,n} W_m, F_0 \leftarrow \hat{f}_{m,n}$;
    $\Delta Q \leftarrow Q_{\text{sum}} - Q, \Delta F \leftarrow F_{\text{sum}} - F$;
    if $\Delta Q > Q_0$ or $\Delta F > F_0$ **then**
      $x_{m,n} \leftarrow 0$, $\lambda_{m,n} \leftarrow 0$, $\hat{f}_{m,n} \leftarrow 0$;
      $x_{m,N+m} \leftarrow 1$;
      $Q_{\text{sum}} \leftarrow Q_{\text{sum}} - Q_0, F_{\text{sum}} \leftarrow F_{\text{sum}} - F_0$;
      Remove pair (m, n) from Φ
    else
      $\lambda' \leftarrow \frac{Q_0 - \Delta Q}{\alpha W_m}, f' \leftarrow F_0 - \Delta F$;
      if $\Gamma(\lambda') < F_0 - \Delta F$ **then**
        $\lambda_{m,n} \leftarrow \lambda', \hat{f}_{m,n} \leftarrow \Gamma(\lambda_{m,n})$
      else
        $\hat{f}_{m,n} \leftarrow f', \lambda_{m,n} \leftarrow \Gamma^{-1}(\hat{f}_{m,n})$
      break
Obtain $\boldsymbol{f}^*$ from $\hat{\boldsymbol{f}}^*$;
return $\boldsymbol{X}^*, \boldsymbol{\lambda}^*, \boldsymbol{f}^*$

Definition 1: The priority function $\phi_{m,n}$ of task offloading pair (m, n) is the weighted sum of storage resource priority function $\phi_{m,n}^r$ and computation resource priority function $\phi_{m,n}^f$. Pair with larger value is assigned with higher priority.

$$\phi_{m,n} = \begin{cases} \sigma_r \phi_{m,n}^r + \sigma_f \phi_{m,n}^f, & \lambda_{m,n} > 0, \\ 0, & \lambda_{m,n} = 0, \end{cases} \tag{23}$$

where $0 \le \sigma_r, \sigma_f \le 1$, $\sigma_r + \sigma_f = 1$. $\phi_{m,n}^r$ and $\phi_{m,n}^f$ are the solution of Lagrange coefficients in problem $\mathbf{P}$.

$$\phi_{m,n}^r = \frac{\kappa_m f_m^2 - \kappa_n \hat{f}_{m,n}^2}{\alpha} - \frac{P_m}{B_m \log_2 \left(1 + \frac{P_m h_m^2 d^{-\zeta}}{\sigma_n^2}\right)}, \tag{24}$$

$$\phi_{m,n}^f = -\lambda_{m,n} \kappa_n W_m \hat{f}_{m,n}. \tag{25}$$

The complete process of computation offloading algorithm is demonstrated in Algorithm 2.

IV. Numerical Results

Our baseline is random offloading scheme (RAO), namely UE offloads its task to MEC server randomly. We assume the effective communication range between UEs and MEC servers is 80m. The energy coefficient of UE is 1×10^{-26}, and the server's is 1×10^{-27}. The task's workload is 2.5×10^8 cycles, and its deadline is 12ms. The proportional parameter $\alpha = 0.2$. The CPU frequency of MEC server is 20GHz, and the computing speed of UE is 400MHz. The energy consumption limit of UE is 150mJ. The wireless channel is Rayleigh channel with path loss exponent $\zeta = 3.5$. The channel bandwidth is 2MHz and the noise power is 10^{-15}W.

TABLE I
Average Running Time (s) vs. Number of UEs

Number of UEs	50	100	200	400
KM	0.0022	0.0041	0.0287	0.3977
MDF-GS	0.0019	0.0036	0.0145	0.0810
MADF-GS	0.0018	0.0035	0.0137	0.0839
MSDF-GS	0.0013	0.0021	0.0067	0.0388
MLDF-GS	0.0008	0.0014	0.0040	0.0207
GS	0.0008	0.0013	0.0040	0.0202

Table I gives the results of average running time versus the number of UEs. It can be observed that the GS algorithm has much shorter average running time than KM algorithm, especially when the number of UEs is large (e.g. 400). This is because the time complexity of KM algorithm is one order of magnitude higher than that of GS algorithms. Therefore, the horrible running time performance of KM algorithm makes it hard to be widely used in the large-scale network. We can see that MADF-GS and MDF-GS have longer running time, since they make more attempts to approach the global optimum.

Figure 2 shows the performance of MSDF-GS, MADF-GS and MDF-GS under different γ. It shows that the energy consumption of MSDF-GS is stable, but MADF-GS and MDF-GS increases when γ is large, especially MDF-GS. This is because these two algorithms overestimate the influence of the worst container's energy consumption, which is contrary to our goal. We can find that the optimal value of γ^* is 0.68, and the best performance of MADF-GS and MDF-GS can be obtained by setting γ^* in advance.

Figure 3 shows the energy consumption as the numbers of UE changes, where subfigure 3(a) evaluates the average energy consumption, and subfigure 3(b) compares the deviation between GS algorithms and KM algorithm. It can be observed that RAO performs worst because it can't make full use of the network status and task features. The performance of ORG-GS is far behind other GS algorithms, since its sorting function is too greedy and easy to fall into local optimum. The performance of MSDF-GS and MLDF-GS is acceptable, because both of them starts to consider the energy consumption difference, which is effective to avoid shortsightedness. MADF-GS and MDF-GS perform better and are basically the same, because it can be proved that the ranking functions of the two are actually equivalent when the number of UEs is large. Besides, nearly all curves in subfigure 3(a) tend to be stable when the number of UEs increases, because servers' resources are limited and some tasks are forced to be executed locally. In conclusion, MADF-GS is the best GS algorithm because it performs well with different γ, approaches the optimum in energy consumption and has the shorter running time.

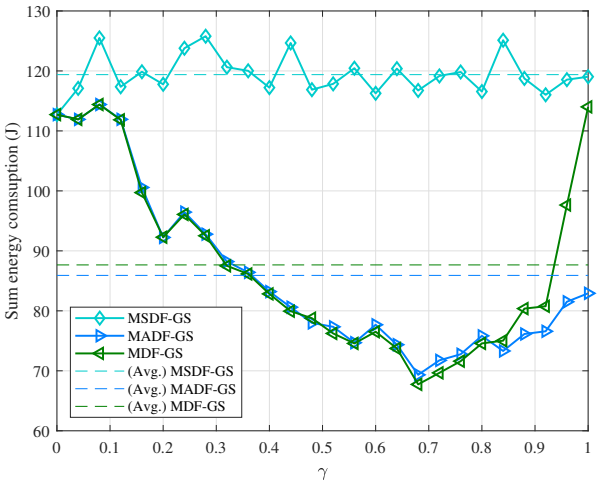
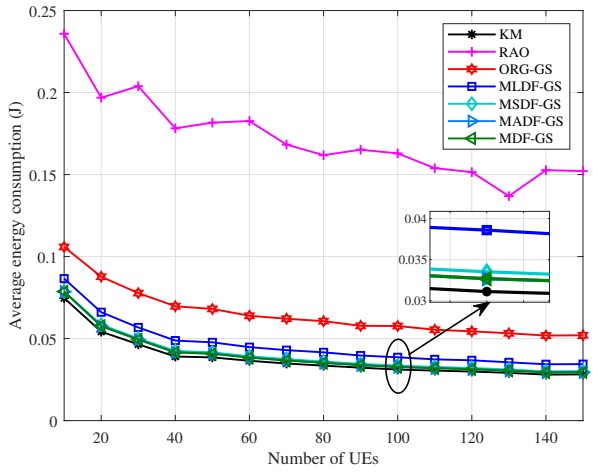
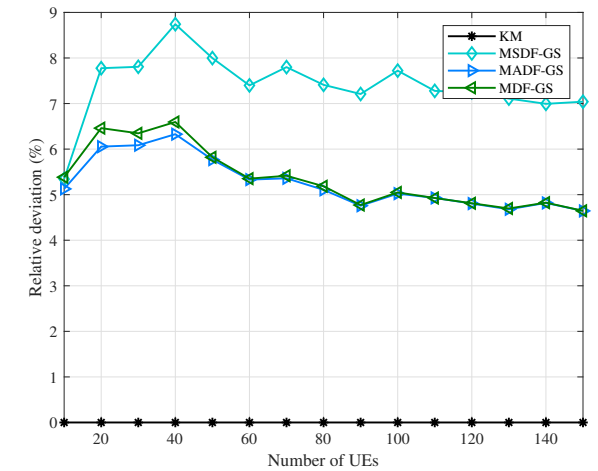


Fig. 2. The sum energy consumption versus γ.



(a) Average energy consumption.



(b) Performance deviation in comparison with KM.

Fig. 3. The energy consumption performance versus the number of UEs.

V. Conclusion

In this paper, we propose a computation offloading scheme to minimize the total energy consumption in mobile edge network. To address the issue, we figure out the optimal offloading proportion and computation resource allocation with binary search algorithm, leverage the graph theory to obtain the result of task offloading decision, and design a priority-based scheme to reassign the allocated resource dynamically. Simulation results show that the proposed algorithm has remarkable performance than randomly offloading, outperforms particularly in the condition of limited resource. Future work is in process, in which we consider the case that the network can't be modeled as a fully connected graph, and the time latency for allocating resources to containers is non-negligible.

Acknowledgment

This work is supported by the Zhejiang Laboratory Open Project Fund 2020LCOAB01 and by the National Science Foundation of China under Grant 61631005, 62071063.

References

[1] S. Wang, X. Zhang, Y. Zhang *et al.*, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

[2] C. You, K. Huang, H. Chae *et al.*, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, 2017.

[3] S. Mu, Z. Zhong, D. Zhao *et al.*, "Joint job partitioning and collaborative computation offloading for internet of things," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 1046–1059, 2019.

[4] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 587–597, 2018.

[5] J. Zhang, H. Guo, J. Liu *et al.*, "Task offloading in vehicular edge computing networks: A load-balancing solution," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 2092–2104, 2020.

[6] S. Jošilo and G. Dán, "Wireless and computing resource allocation for selfish computation offloading in edge computing," in *Proc. IEEE INFOCOM*, 2019, pp. 2467–2475.

[7] H. A. Alameddine, S. Sharafeddine, S. Sebbah *et al.*, "Dynamic task offloading and scheduling for low-latency iot services in multi-access edge computing," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 668–682, 2019.

[8] J. Zhang, J. Du, C. Jiang *et al.*, "Computation offloading in energy harvesting systems via continuous deep reinforcement learning," in *IEEE Int. Conf. Commun.*, 2020, pp. 1–6.

[9] M. Li, J. Gao, L. Zhao *et al.*, "Deep reinforcement learning for collaborative edge computing in vehicular networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 4, pp. 1122–1135, 2020.

[10] J. Shi, J. Du, J. Wang *et al.*, "Distributed V2V computation offloading based on dynamic pricing using deep reinforcement learning," in *IEEE Wireless Commun. Networking Conf.*, May 2020, pp. 1–6.

[11] X. Qiu, W. Zhang, W. Chen *et al.*, "Distributed and collective deep reinforcement learning for computation offloading: A practical perspective," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 5, pp. 1085–1101, 2021.

[12] Y. Wang, M. Sheng, X. Wang *et al.*, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct 2016.

[13] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856–868, 2019.

[14] 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, 2012.

[15] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *American Mathematical Monthly*, vol. 69, no. 1, pp. 9–15, 1962.