Task Offloading and Resource Allocation for Container-enabled Mobile Edge Computing

Ao Zhou, Sisi Li, Shangguang Wang
State Key Laboratory of Network and Switching Technology
Beijing University of Posts and Telecommunications
Beijing, China
{aozhou, sisili, sgwang}@bupt.edu.cn

***Abstract*—A growing number of mobile services demand intensive computation resources and high energy consumption. Traditional cloud-based task offloading incurs excessive transmission delay. With mobile edge computing, we can offload tasks to the nearby edge servers, thus mitigating long data transmission latency. Current task offloading and resource allocation approaches ignore the extra overhead brought by container instance startup. This naturally leads to a long overall latency and high energy consumption. Motivated by the limitations of existing literature, we address task offloading and resource allocation problem considering container instance startup time. We first examine the startup latency of containers started from different types of images, and present its impact on overall performance. Then, we formulate the problem as optimization problem. The problem is hard to address because mutual effects of offloading decision, resource allocation decision and container instance startup contribute to the overall performance. We resort to Gibbs sampling to decouple offloading decision and resource allocation. Finally, taking into account all potential conditions brought up by container instance startup, an optimal computation resource allocation and uplink power assignment scheme are derived through bisection, and Karush-Kuhn-Tucher (KKT) conditions. Comprehensive evaluations are conducted to verify the performance of the proposed approach, and demonstrate its superiority over current notable solutions.**

***Index Terms*—Mobile edge computing, Task offloading, Resource allocation, Container.**

I. Introduction

With the proliferation of smart mobile devices (MDs), a growing number of computation-intensive and energy-hungry mobile services have emerged and attracted great interests. These mobile services commonly consume significant amount of computation resources with a tremendously high energy consumption. However, the MDs are constrained in terms of computation capacity and battery capacities, due to the size limitation.

Offloading computations via a wireless network to a remote cloud data center has emerged as a solution to relieve the battery of the MDs. However, offloading to the remote cloud data center has the drawbacks of high overhead and long backhaul latency. Mobile edge computing (MEC) is emerging as a promising solution to address this issue. The key advantage of MEC is to provide cloud computing capabilities at the network edge to meet the requirements of all mobile applications. Each cellular Base Station (BS) is equipped with an edge cloud, and the MDs within the coverage area of the BS can offload their tasks to the edge cloud. Hence, the tasks are allowed for execution in close proximity to the MDs. The architecture can substantially reduce end-to-end latency, and release the traffic burden on backhaul networks.

Due to the data communication required between the MDs and the edge cloud in the uplink wireless channels, offloading tasks to the edge cloud incurs extra overheads in terms of latency and energy consumption. Moreover, the computation resources at the edge cloud is limited, which calls for a notable mechanism to allocate computation resources coordinately among multiple MDs. Hence, offloading decision (a task should be executed locally or be remotely at edge cloud) and resource allocation (computation resource allocation and uplink transmission power assignment) become critical issues toward enabling efficient computation offloading. The problem has been addressed by many researchers [1] [2] [3].

However, current approaches ignore the extra overhead brought by container instance startup. To improve resource utilization, the edge cloud adopts light weight virtualization solutions (e.g., container) to deploy application. Hence, application is started by a fine-grained virtualization instance(e.g. container instance started from docker container image). For each MD, a container instance is brought up on the edge cloud to offer all necessary software module and computing environment desired. A task can start to be processed only after the corresponding container instance has been launched. This factor is significant in computation offloading, and further increases the complexity of the original optimization problem. For example, increasing the uplink transmission power would not reduce the end-to-end latency when the data uploading latency is smaller than the instance startup time; allocating more computation resource can compensate the negative impact of long container instance startup time. Resource limitation and competition among users naturally make this problems more noticeable.

To tackle these difficulties, we propose a **t**wo-layer **o**ffloading **d**ecision and **r**esource **a**llocation approach (TODRA) for container-enabled mobile edge computing. Considering the variety of potential conditions caused by instance startup, we seek to optimize the task offloading decision, uplink transmit power and computation resource allocation.

The main contributions of this paper are listed as follows.

- We have made an empirical study on AliCloud to show

the impact of instance startup time on overall performance. We have made the experiment on 57 Docker container images belonging to five broad categories.
- We address the task offloading and resource allocation problem with the consideration of instance startup time in MEC. By analyzing the impact of instance startup time on the end-to-end latency, the task offloading decision, uplink transmit power and computation resource allocation are comprehensively considered in optimization. We formulate the problem as mix-integer nonlinear programming (MINLP). The problem is difficult to solve for the mutual effects of multiple factors.
- To address this issue, we propose a two-layer offloading decision and resource allocation approach, which makes the offloading decision and resource allocation decision in an iterative manner. In the first layer, Gibbs sampling is adopted to make the offloading decision. After obtaining the offloading decision from the first layer and analyzing all potential conditions caused by instance startup, we make the joint computation resource and uplink power allocation decision by harnessing bisection, and KKT conditions in the second layer.
- We carry out extensive numerical simulations to evaluate the performance of the proposed approach, and the experiment results illustrate that the proposed approach can improve offloading efficiency over traditional approaches.

The rest of this paper is organized as follows. In Sections 2, we give a brief review of prior works. Some empirical studies are given in Section 3 to illustrate our motivation. The system model is introduced in Section 4. We formulate the problem in Section 5. In Section 6, the solution of the problem is analyzed. Simulation results are presented in Section 7. In Section 8, conclusions are drawn.

II. Related Works

Current research can be viewed from two main perspectives: offloading to a remote public cloud, and to edge clouds.

One solution to reduce energy consumption and enhance computation capability of MDs is to offload computation-intensive tasks to the remote cloud. Nir et al. [4] propose a monetary cost and energy-aware offloading scheduler for multi-tasks. Moreover, Nir et al. [5] propose an energy-aware offloading scheduler for multi-tasks. In order to migrate computation-intensive tasks from MDs to public cloud effectively, Xia et al. [6] propose an offloading decision-making system to save energy on the MDs. Zheng et al. [7] and Chen [8] adopt the game theoretic to achieve efficient computation offloading for MDs. Another solution is to offload computation-intensive tasks to edge clouds [9] [10] [11]. Yang et al. [12] propose a distributed computation offloading decision-making strategy for a multi-device and multi-server system in small-cell networks. Yi et al. [13] propose an algorithm that jointly determines the optimal computation and transmission scheduling strategy. To minimize the energy consumption of all MDs, a computation offloading decision making and spectrum selection mechanism is proposed by Guo et al. [14]. Meng et al. [15] investigate online deadline-aware task scheduling problem in MEC. Mao et al. [1] allocate the communication and computation resource allocation to optimize the energy efficiency in wireless powered MEC. Lyu et al. [2] propose a semi-distributed algorithm to maximize the weighted sum utility of energy consumption and delay. Tran et al. [3] discuss the joint resource scheduling and offloading decision making to maximize the weighted sum utility in multi-server edge computing networks.

However, approaches developed in these works cannot be applied for solving the problem in this paper, because none of them realize the impact of the container instance startup latency. As will be seen shortly, our proposed approach avoids the problem effectively.

III. Measurement Study

In this section, we begin with some empirical studies to prove that instance startup time has significant impact on overall performance and illustrate the limitations of current offloading approaches.

A. Experiment Setup

Virtualization platform. In our measurement, we employ Docker for service deployment. In Docker, each application is packed into a Docker container image, and application per container instance becomes the main deployment pattern. Docker container images become container instances at runtime. We refer "docker container image" as "image" for short.

Image pulling. Due to the large size and various types, it is impractical to cache all images over the edge clouds. We evaluate the overall performance under three image pulling schemes. a) Remote pulling. Images are pulled from the remote cloud, or Docker Hub. b) Edge pulling. An image is pulled from a nearby edge server if the image has been cached on it. c) No pulling. Image do not need to be pulled if it has been cached on the target edge server.

Hardware. We purchase one cloud server and two edge servers from AliCloud. The cloud server acts as a remote storage server, and the two edge servers are employed for container instance startup and image caching, respectively. The edge server for container instance startup is equipped with a 8 core Intel Xeon processor running 2.5 GHz with 11GB of RAM.

Image selection. It is impractical to experiment on all types of images. 57 images are selected in Docker Hub, and the images are divided into five broad categories including Linux Distro, Database, Language, Web Server and other [16]. The booting latency is measured by either running a simple "hello world" program in the container instance or waiting until the container instance echos "hello".

B. Results

Booting Time of Different Images. The booting time of different images is shown in Fig. 1. The reserved memory is specified to 512MB and 1GB. The CDF of container instance booting time with reserved memory from 512MB to 11GB

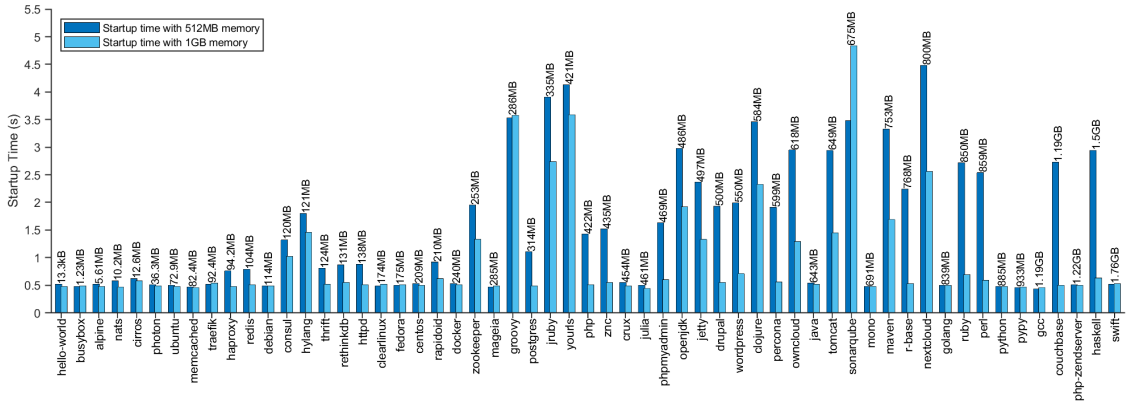


Fig. 1. Booting time of different images.

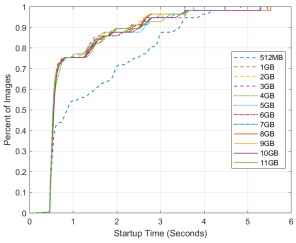


Fig. 2. Startup time.

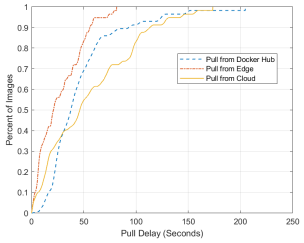
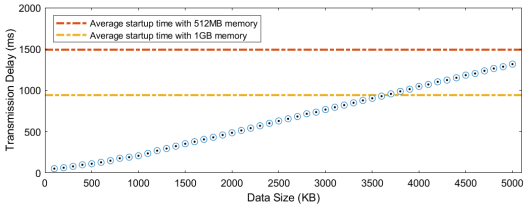
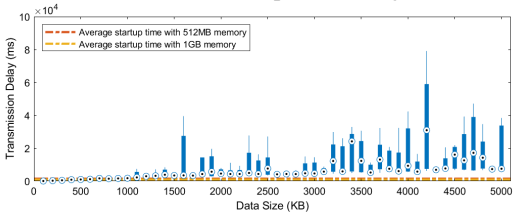


Fig. 3. Pull time.



(a) Data transmission from phone to edge via WiFi.



(b) Data transmission from phone to edge via 4G.

Fig. 4. Data transmission time.

is shown in Fig. 2. The booting time of different images varies. The average booting time is 1.4921s. The mean value of booting time under 1GB memory is 0.9421s, and the maximum and the minimum value are 3.7612s and 0.4530s.

Pull Time of Different Images. We measure the pull time for each image mentioned above. Fig. 3 shows the CDF of the pull time. The best performance is achieved by pulling images from nearby edge servers, and its average pull time is 26.6967s. Due to the Docker Hub server and cloud server are both far away from the edge, the required time of fetching images from them are similar and long. The mean pull time of them are 46.5937s and 54.9539s, respectively.

Data Transmission Time of Different Data Size. The data of task needs to be uploaded to the edge server before processed. The size of the data to be processed varies from 100KB to 5000KB. Taking the fluctuation of network quality into account, multiple data transmission tests are conducted at different times. Fig. 4a and Fig. 4b illustrate the experiment results. Obviously, the transmission time increases with the size of the data.

The dotted lines in Fig. 4 are the average container instance startup time of 57 types of images. The dotted lines intersect with the data transmission time curve, which indicates that the startup time can be larger or less than the transmission time. For example, the container startup time for "python" is 0.48s with 1GB memory. When the data size is larger than 2000KB, the container instance startup time may be less than the data transmission latency in WiFi network.

C. Key Observations

1) The startup time can be larger or less than the transmission time, depending on the type of image, the location where the required image is stored and the size of data to be uploaded. 2) Both container instance startup and data transmission can be the performance bottleneck, depending on which one is more time-consuming. 3) Ignoring the startup time may lead to the wrong offloading decision and high end-to-end task processing cost. 4) Increasing the uplink transmission power would not contribute to the overall performance improvement when the startup time is longer than the data transmission time; that is to say, energy of the mobile device is wasted.

IV. System Model

A. Mobile Edge Computing

We consider a scenario where multiple MDs (i.e users) are within the coverage area of one BS. The set of MDs are denoted as $\mathbb{I}=\{1,2,...,I\}$. The BS is co-located with an edge cloud, which is denoted as $\hat{d}$. Let F^{Edge} represent the overall

computation resource of the edge cloud $\hat{d}$. Supposing that each MD i has one task to be processed. The task from MD i is described as $\{b_i, c_i\}$, where b_i denotes the size of data to be processed, and c_i denotes the number of CPU cycles needed to complete the task. Due to the factor that the size of the computation outcome is much small, we neglect the download time. This assumption is similar to other studies [3] [17] [2]. Each task can be executed locally or remotely at the edge cloud $\hat{d}$. The edge cloud adopts container, a light weight virtualization solution, to deploy application. We need to launch a container instance on the edge cloud to offer all necessary computing environment desired for MDs. The data from the MD start to be processed once the corresponding container instance has been booted and the data has been received.

B. Communication Model

The orthogonal frequency division multiple access (OFDMA) is adopted, and the spectrum of BS is divided into M homogeneous orthogonal channels. The data transmission rate can be calculated as follows:

$$R_i = W\log_2(1 + \frac{p_i g_i}{N_0}), \tag{1}$$

where W denotes the channel bandwidth, p_i is the transmission power of MD i, g_i denotes the channel gain from the MD to the BS with a consideration of path loss and fading, and N_0 denotes the Gaussian noise power. According to the equation, we can adjust the value of p_i to control the data transmission rate.

C. Computation Model

Each task can be executed locally or remotely at the edge cloud. Therefore, we describe two computation model in this section. We now define an offloading decision indicator x_i, and x_i is set to 1 when the corresponding task of MD i is offloaded to the edge cloud.

1) Local Computation: The overhead of local computing is composed of two parts: execution latency and energy consumption. Let f_i^{Local} denotes the computation capability (CPU cycles per second) of MD i. The task completion time can be written as follows:

$$T_i^{Local} = \frac{c_i}{f_i^{Local}}. \tag{2}$$

According to [3], the local energy consumption is a superlinear function of frequency, and the unit energy consumption ε_i^{Local} is given as:

$$\varepsilon_i^{Local} = \alpha(f_i^{Local})^{\gamma}, \tag{3}$$

where $\alpha = 10^{-11}$,$\gamma = 2$. The energy consumption is given as:

$$E_i^{Local} = \varepsilon_i^{Local} T_i^{Local} = \alpha(f_i^{Local})^{\gamma} \frac{c_i}{f_i^{Local}} = \alpha c_i (f_i^{Local})^{\gamma\text{-}1}. \tag{4}$$

2) Mobile Edge Computation: The latency consists of the time to prepare container and the time of task execution. The preparing time is determined by the time of data uploading, and the time of instance startup. The time of data uploading is calculated by the following:

$$T_i^{R,Edge} = \frac{b_i}{R_i} = \frac{b_i}{W\log_2(1 + \frac{p_i g_i}{N_0})}. \tag{5}$$

The preparing process consists of container image file pulling and container instance launching. In the image file pulling stage, the image is transferred from the storage node to the target node. In the instance launching stage, an instance is startup from the fetched image file. If the required image file is cached at the edge network, the image file can be fetched from the storage edge cloud, thereby reducing the startup latency. However, due to the limited storage space of edge cloud, not all image files can be cached at the same time. When an image file is not cached at the edge network, the image file should be fetched from the remote cloud, where all images are stored. The set of edge clouds at the edge network where $\hat{d}$ belongs to are denoted as $\mathbb{D} = \{1, 2, ..., D\}$. The transmission bandwidth between two edge clouds is given by $l(d_1, d_2)$, $d_1, d_2 \in \mathbb{D}$. The time to fetch the image from the remote cloud is denoted by h_t. Moreover, we denote $\mathbb{S} = \{1, 2, ..., S\}$ as the set of image files, λ_s as the size (in bit) of image file s, and θ_s as the required instance booting time from s. We now define an image demand indicator $\xi_{i,s}$, and $\xi_{i,s}$ is set to 1 when s is required to startup the instance for MD i. Otherwise, $\xi_{i,s}$ is set to 0. $\eta_{s,d}$ is employed to indicate whether image s is cached on the edge cloud d. Now, the preparing time is given as:

$$\begin{aligned} T_i^{S,Edge} = \min_{d\in D} & \frac{\sum\limits_{s\in S} \xi_{i,s}\lambda_s}{\sum\limits_{s\in S} \xi_{i,s}\eta_{s,d} l(\hat{d}, d)} + \\ & (1 - \sum_{s\in S, d\in D} \xi_{i,s}\eta_{s,d}) ht + \sum_{s\in S} \xi_{i,s}\theta_s. \end{aligned} \tag{6}$$

We denote f_i^{Edge} as the computation resources allocated to MD i. The remote execution time is given as:

$$T_i^{C,Edge} = \frac{c_i}{f_i^{Edge}}. \tag{7}$$

A task can be processed by the edge cloud only after the instance has been startup, and the data have been uploaded. Therefore, the task completion time is written as follows:

$$T_i^{Edge} = \max\{T_i^{R,Edge}, T_i^{S,Edge}\} + T_i^{C,Edge}. \tag{8}$$

The transmission power of MD i is denoted as p_i. Then the transmission energy consumption of MD i is calculated as follows:

$$E_i^{Edge} = p_i T_i^{R,Edge} = \frac{p_i b_i}{W\log_2(1 + \frac{p_i g_i}{N_0})}. \tag{9}$$

3) Problem Formulation: Similar to [2] [3], we define ω_i^T, ω_i^E to denote the weights of completion time and energy consumption, respectively. Different MDs can choose different weights. The MD can put more weight on energy consumption (i.e., a larger ω_i^E) to save more energy. This mechanism can

provide more flexibility and meet MD-specific demands. The utility function of MD i is given by:

$$U_i = x_i \left(\omega_i^T \frac{T_i^{Local} - T_i^{Edge}}{T_i^{Local}} + \omega_i^E \frac{E_i^{Local} - E_i^{Edge}}{E_i^{Local}} \right). \quad (10)$$

The task offloading and resource allocation problem becomes a utility maximization problem. We formulate the problem as follows:

P:

$$\max_{X,F,P} U = \sum_{i \in \mathbb{I}} U_i \quad (11)$$

s.t.

$$\sum_{i \in \mathbb{I}} f_i^{Edge} \leq F^{Edge} \quad (11a)$$

$$f_i^{Edge} > 0, \forall i \in \mathbb{I}, x_i > 0 \quad (11b)$$

$$\sum_{i \in \mathbb{I}} x_i \leq M \quad (11c)$$

$$p_i \in [0, p_i^{\max}], \forall i \in \mathbb{I} \quad (11d)$$

$$x_i \in \{0, 1\}, \forall i \in \mathbb{I} \quad (11e)$$

$X = \{x_1, x_2, ..., x_I\}$ denotes the offloading decision profile, $F = \{f_1^{Edge}, f_2^{Edge}, ..., f_I^{Edge}\}$ denotes the computation resource allocation profile, and $P = \{p_1, p_2, ..., p_I\}$ denotes the uplink transmission power profile. (11a) states that the overall allocated computation resources should be less than maximum computation resources of the edge cloud. (11b) specifics that we need to allocate computation resources for offloaded tasks. (11c) guarantees that at most M MDs are allowed to transmit simultaneously. (11d) ensures that the uplink transmission power must be positive, and must not exceed the maximum transmission power. (11e) implies that a task can be either locally executed or offloaded to the edge cloud. Now, the problem is formulated as a mix-integer nonlinear programming (MINLP).

V. Offloading and Resource Allocation Approach

To address this issue, we propose a **t**wo-layer **o**ffloading **d**ecision and **r**esource **a**llocation algorithm, which make the offloading decision and resource allocation decision in an iterative manner. In the first layer, the Gibbs sampling is adopted to make the offloading decision. After obtain the offloading decision from the first layer, we make the joint computation resource and uplink power allocation decision based on bisection, and KKT conditions in the second layer.

A. Offloading Decision Making based on Gibbs Sampling

We now exploit the Gibbs Sampling to obtain the offloading decision, and the algorithm workflow is shown in Algorithm 1. In each iteration, a randomly selected MD i virtually changes its current offloading decision x_i^* to $x_i^{\#}$, and correspondingly, the offloading decision profile of all MDs is represented as X^* and $X^{\#}$. With the given policy $X^{\#}$, **P** is reduced to the resource allocation problem as follows:

$$\max_{X,F,P} \sum_{i \in \mathbb{I}} \left(\omega_i^T \frac{T_i^{Local} - T_i^{Edge}}{T_i^{Local}} + \omega_i^E \frac{E_i^{Local} - E_i^{Edge}}{E_i^{Local}} \right). \quad (12)$$

Then the optimal resources allocation strategy under $X^{\#}$ is derived by solving (12). Thirdly, transition probability ϑ is calculated based on U^* and $U^{\#}$. x_i^* is updated to $x_i^{\#}$ with probability ϑ, and keeps unchanged with probability $(1 - \vartheta)$. The above process repeats until the stopping criterion is satisfied.

The algorithm explores a new offloading decision with a certain probability to avoid being trapped in a local optimum. The parameter $\delta > 0$ is used to control exploration versus exploitation. A smaller value of δ implies that the algorithm keep a new decision with higher probability.

Lemma 1: The algorithm converges with a higher probability to the global optimal solution with the decrease of δ. The algorithm converges to the global optimal solution with probability 1 when the value of δ tends towards 0.

Proof: Please see Appendix A.

Algorithm 1: Offloading Decision based on Gibbs Sampling

Input: the parameters of P
Output: the optimal offloading decision and resource allocation strategy

1 Initialize X^*, F^*, P^* to **0**;
2 $U^* = U(X^*, F^*, P^*)$;
3 Randomly pick a MD $i \in \mathbb{I}$, and $x_i^{\#}$ = 1-x_i^*;
4 $X^{\#} = X^* \backslash \{x_i^*\} \bigcup \{x_i^{\#}\}$;
5 **if** $X^{\#}$ *is feasible* **then**
6 Obtain $F^{\#}$ and $P^{\#}$ by minimizing (12);
7 $U^{\#} = U(X^{\#}, F^{\#}, P^{\#})$;
8 $\vartheta = \frac{1}{1+e^{-(U^{\#} - U^*)/\delta}}$;
9 With probability ϑ, assign $X^{\#}$, $F^{\#}$, $P^{\#}$ to X^*, F^*, P^*;
10 With probability $(1 - \vartheta)$, X^*, F^*, P^* keep unchanged;
11 **end**
12 **if** *the stopping criterion is satisfied* **then**
13 Return X^*, F^*, P^*;
14 **end**
15 **else**
16 Goto Line 2;
17 **end**

B. Computation Resource and Uplink Power Allocation

In the second layer, the offloading decision from the first layer is obtained. The problem is reduced to (12). Substituting (1),(2),(3) into (12), we can rewritten the object function as (13). The first part becomes a constant, the second part is related to computation resource allocation, and the third

$$\underbrace{\sum_{i\in\mathbb{I}}(\omega_i^T+\omega_i^E)}_{the\quad first\quad part}+\underbrace{\min_{F}\sum_{i\in\mathbb{I}}\frac{\omega_i^T f_i^{Local}}{f_i^{Edge}}}_{the\quad second\quad part}+\underbrace{\min_{P}\sum_{i\in\mathbb{I},\hat{x}_i>0}\left(\frac{\omega_i^T f_i^{Local}}{c_i}\max\{\frac{b_i}{W\log_2(1+\frac{p_i g_i}{N_0})},T_i^{S,Edge}\}+\frac{\omega_i^E p_i b_i}{\alpha c_i(f_i^{Local})^{\gamma\text{-}1}W\log_2(1+\frac{p_i g_i}{N_0})}\right)}_{the\quad third\quad part}. \tag{13}$$

part is related to uplink power allocation. Because F, P are decomposed from each other now, we can decompose (13) into two dependent subproblems:

P1:

$$\min_{F}\sum_{i\in\mathbb{I},\hat{x}_i>0}\frac{\omega_i^T f_i^{Local}}{f_i^{Edge}} \tag{14}$$

s.t.

$$\sum_{i\in\mathbb{I}} f_i^{Edge}\le F^{Edge} \tag{14a}$$

$$f_i^{Edge}>0,\forall i\in\mathbb{I},x_i>0 \tag{14b}$$

P2:

$$\min_{P}\left(\begin{array}{l}\frac{\omega_i^T f_i^{Local}}{c_i}\max\{\frac{b_i}{W\log_2(1+\frac{p_i g_i}{N_0})},T_i^{S,Edge}\}\\+\frac{\omega_i^E p_i b_i}{\alpha c_i(f_i^{Local})^{\gamma\text{-}1}W\log_2(1+\frac{p_i g_i}{N_0})}\end{array}\right) \tag{15}$$

s.t.

$$C4: p_i\in[0,p_i^{\max}],\forall i\in\mathbb{I} \tag{15a}$$

P1 is to find the optimal computation resource allocation strategy, and **P2** is to find the optimal uplink power for each MD i ($i\in\mathbb{I}$ and $\hat{x}_i>0$).

Algorithm 2: Transmission Power Allocation

Input: P3 and P4
Output: the optimal transmission power

```
Initialize p_L = 0 and p_U = p_i^{max}(t);
Solve P3 by using Algorithm 3, and assign the
  solution to p_i^{L*};
Solve P4 by using Algorithm 4, and assign the
  solution to p_i^{R*};
if Γ_3(p_i^{L*}) < Γ_4(p_i^{R*}) then
    Return p_i^{L*};
end
else
    Return p_i^{R*} ;
end
```

1) Optimization of Computation Resources Allocation:

Lemma 2: P1 is a convex optimization problem.

Proof: See Appendix B.

We can adopt the Lagrangian optimization and Karush-Kuhn-Tucker (KKT) condition to solve the problem. The Lagrangian function of **P1** is as follows:

$$L(F,\beta)=\sum_{i\in\mathbb{I},\hat{x}_i>0}\frac{\omega_i^T f_i^{Local}}{f_i^{Edge}}+\beta\left(\sum_{i\in\mathbb{I},\hat{x}_i>0}f_i^{Edge}-F^{Edge}\right). \tag{16}$$

Based on the KKT condition, we can obtain that the optimal f_i^{Edge*} satisfies:

$$\frac{\partial L(F,\beta)}{\partial f_i^{Edge}}\Big|_{f_i^{Edge*}}=-\frac{\omega_i^T f_i^{Local}}{\left(f_i^{Edge*}\right)^2}+\beta=0, \tag{17}$$

$$f_i^{Edge*}=\sqrt{\frac{\omega_i^T f_i^{Local}}{\beta}}. \tag{18}$$

Substituting (18) into (16), we can obtain that:

$$L(\beta)=2\sum_{i\in\mathbb{I},\hat{x}_i>0}\sqrt{\beta\omega_i^T f_i^{Local}}-\beta F^{Edge}, \tag{19}$$

$$\frac{\partial L(\beta)}{\partial\beta}|_{\beta^*}=\sum_{i\in\mathbb{I},\hat{x}_i>0}\sqrt{\frac{\omega_i^T f_i^{Local}}{\beta^*}}-F^{Edge}=0, \tag{20}$$

$$\beta^*=\frac{\left(\sum_{i\in\mathbb{I},\hat{x}_i>0}\sqrt{\omega_i^T f_i^{Local}}\right)^2}{\left(F^{Edge}\right)^2}. \tag{21}$$

Substituting (21) into (18), we can obtain that:

$$f_i^{Edge*}=F^{Edge}\frac{\sqrt{\omega_i^T f_i^{Local}}}{\sum_{i\in\mathbb{I},\hat{x}_i>0}\sqrt{\omega_i^T f_i^{Local}}}. \tag{22}$$

2) Optimization of Uplink Power Allocation: Two cases of **P2** are discussed. When the data uploading time is smaller than the preparing time, **P2** can be transformed into:

P3:

$$\min_{P}\left(\frac{\omega_i^T f_i^{Local}T_i^{S,Edge}}{c_i}+\frac{\omega_i^E p_i b_i}{\alpha c_i(f_i^{Local})^{\gamma\text{-}1}W\log_2(1+\frac{p_i g_i}{N_0})}\right) \tag{23}$$

s.t.

$$\frac{b_i}{W\log_2(1+\frac{p_i g_i}{N_0})}\le T_i^{S,Edge} \tag{23a}$$

$$p_i\in[0,p_i^{\max}],\forall i\in\mathbb{I} \tag{23b}$$

The lower bound is deduced by (23a):

$$p_i \geq \left(\frac{N_0}{g_i}\right) 2^{\frac{b_i}{WT_i^{S,Edge}}} - \frac{N_0}{g_i}. \tag{24}$$

When the data uploading time is larger than the preparing time, **P2** can be transformed into:

P4:

$$\min_{P} \left(\frac{\omega_i^T f_i^{Local} b_i}{c_i W \log_2(1+\frac{p_i g_i}{N_0})} + \frac{\omega_i^E p_i b_i}{\alpha c_i (f_i^{Local})^{\gamma\text{-}1} W \log_2(1+\frac{p_i g_i}{N_0})} \right) \tag{25}$$

s.t.

$$\frac{b_i}{W\log_2(1+\frac{p_i g_i}{N_0})} > T_i^{S,Edge} \tag{25a}$$

$$p_i \in [0, p_i^{\max}], \forall i \in \mathbb{I} \tag{25b}$$

The upper threshold is calculated by (25a):

$$p_i \leq \left(\frac{N_0}{g_i}\right) 2^{\frac{b_i}{WT_i^{S,Edge}}} - \frac{N_0}{g_i}. \tag{26}$$

We denote $\Gamma_3(p_i), \Gamma_4(p_i)$ as the optimization function of **P3** and **P4**, respectively. We would solve **P3**, **P4**, and select the better one as the optimal solution. The algorithm is shown in Algorithm 2.

Lemma 3: $\Gamma_3(p_i)$ is quasiconvex in the domain.

Proof: See Appendix C.

Lemma 4: $\Gamma_4(p_i)$ is quasiconvex in the domain.

Proof: For quite similar to the proof of Lemma 3, we omit the detailed description.

Algorithm 3: Transmission Power Allocation for P3

Input: P3
Output: the optimal transmission power

if $\left(\frac{N_0}{g_i}\right) 2^{\frac{b_i}{WT_i^{S,Edge}}} - \frac{N_0}{g_i} > p_i^{max}$ **then**
  Return ∞ ;
end
if $\Gamma_3(\left(\frac{N_0}{g_i}\right) 2^{\frac{b_i}{WT_i^{S,Edge}}} - \frac{N_0}{g_i})' >= 0$ **then**
  Return $\left(\frac{N_0}{g_i}\right) 2^{\frac{b_i}{WT_i^{S,Edge}}} - \frac{N_0}{g_i}$;
end
Initialize $p_L = \left(\frac{N_0}{g_i}\right) 2^{\frac{b_i}{WT_i^{S,Edge}}} - \frac{N_0}{g_i}$ and $p_U = p_i^{max}$;
while p_U-$p_L > \epsilon$ **do**
  p_M=(p_L+p_U)/2;
  if $\Gamma_3(p_M)' > 0$ **then**
    $p_U = p_M$;
  end
  else
    $p_L = p_M$;
  end
end
p_i^*=(p_L+p_U)/2;
Return p_i^*;

Since the $\Gamma_3(p_i), \Gamma_4(p_i)$ is quasiconvex and its constraints are linear, the optimal solution can be obtained at either the boundary points or the stationary points. For **P3**, if the derivative value of $\Gamma_3(p_i)$ at $\left(\frac{N_0}{g_i}\right) 2^{\frac{b_i}{WT_i^{S,Edge}}} - \frac{N_0}{g_i}$ is greater than 0, then $\Gamma_3(p_i)$ is monotonically increasing and the minimum function value is obtained at $\left(\frac{N_0}{g_i}\right) 2^{\frac{b_i}{WT_i^{S,Edge}}} - \frac{N_0}{g_i}$. Otherwise, we use the bisection method to find the stationary point. With the same scheme, we can get the optimal value of **P4**. The details of the solutions for **P3** and **P4** are presented in Algorithm 3 and Algorithm 4, respectively.

Algorithm 4: Transmission Power Allocation for P4

Input: P4
Output: the optimal transmission power

if $\Gamma_4(\left(\frac{N_0}{g_i}\right) 2^{\frac{b_i}{WT_i^{S,Edge}}} - \frac{N_0}{g_i})' <= 0$ **then**
  Return $\left(\frac{N_0}{g_i}\right) 2^{\frac{b_i}{WT_i^{S,Edge}}} - \frac{N_0}{g_i}$;
end
Initialize $p_L = 0$ and $p_U = \left(\frac{N_0}{g_i}\right) 2^{\frac{b_i}{WT_i^{S,Edge}}} - \frac{N_0}{g_i}$;
while p_U-$p_L > \epsilon$ **do**
  p_M=(p_L+p_U)/2;
  if $\Gamma_4(p_M)' > 0$ **then**
    $p_U = p_M$;
  end
  else
    $p_L = p_M$;
  end
end
p_i^*=(p_L+p_U)/2;
Return p_i^*;

VI. Performance Evaluation

A. Experiment Setup

In this section, numerical results are provided to evaluate the performance of the proposed offloading decision and resource allocation approach. The simulation settings are as follows unless otherwise stated. There are 40 MDs, 20 sub-channels for uplink communications. We consider similar network settings with [2] [3]. The noise power of each channel is set to $10^{-12}W$. The maximum transmit power for each MD is set as 24 dBm. The sub-channel power gain is modeled as $(d_i)^{-3}h_0$, where d_i denotes the distance from the MD i to the BS. h_0 denotes the Rayleigh fading coefficient, which follows the exponential distribution with the mean value of 1. For the task from MDs $\in \mathbb{I}$, both the data size b_i and the required number of CPU cycles c_i follow the uniform distribution within [500, 2000] KB and [1000, 1500]M CPU cycles, respectively. The total computation resource of the MDs follow the uniform distribution within [0.5, 1.5]G CPU cycles/s. The total computation resource of the edge cloud is set as 20G CPU cycles/s. The probability of image files being

required follows a Zipf-like distribution [18]. Without loss of generality, we rank these image files in a descending order according to the popularities. The request probability of the i^{th} image file is denoted as

$$\rho_i = \frac{1/i^{\beta}}{\sum_{s=1}^{S} 1/s^{\beta}}, \tag{27}$$

where the Zipf parameter β represents the skewness in the MDs' preference, and $\beta = 0.6$ [18]. The caching of image files at edge clouds $\mathbb{D}$ follows the same distribution. All random variables are independent for different MDs, modeling heterogeneous mobile edge computing environment.

We compare our approach with the following benchmark schemes.

- Local Computation Only (LCO). The MDs can only process the computation data locally.
- Computation Offloading Only (COO). The tasks offloading is scheduled to occupy all the sub-channels, and then resource is joint allocated among users considering the preparing time.
- Submodular-based Offloading Decision and Resource Allocation (SODRA) [2] [19]. The offloading decision is obtained by submodular set function optimization, and the preparing time is ignored.
- Heuristic Offloading Scheduling and Resource Allocation (HOSRA) [3]. The resource allocation is addressed with convex and quasi-convex optimization techniques, and offloading strategy is solved with heuristic algorithm. But the preparing time is ignored.
- Independent Offloading and Joint Resource Allocation (IOJRA) [20]. Each mobile user independently make offloading decision, and joint resource allocation considering the preparing time is employed.

B. Experiments Results

1) Convergence: Fig. 5 shows the convergence process of the proposed algorithm. In general, a faster convergence speed can be obtained with a smaller δ. However, the value of δ affects not only the convergence speed, but also the convergence results. Keeping decreasing the parameter δ impedes the identification of global optimum and results in the convergence to inferior solutions. In our experiment, the value of δ is set to be 0.001 to achieve a good trade-off between the solution quality and the convergence rate.

2) Impact of Device Number: Fig. 6 shows the evaluation of system utility as the number of users increase from 10 to 40. In the case of LCO, the system utility is always 0, and it does not change with increasing users. COO always offloads users as many as possible to make full use of channel resource. IOJRA make offloading decisions for each user independently rather than from a global perspective, which enables to select more tasks to offload. Both of them ignore the fierce competition for the limited computation and power resource among offloading users, which leads to the degradation of system utility with more users. The performance of IOJOA starts to degrade when the number of users is greater than 9 and even get

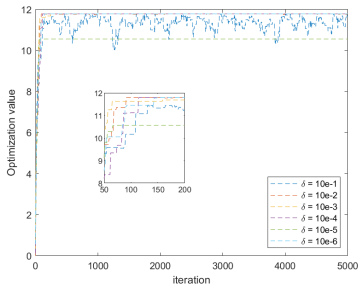


Fig. 5. Impact of δ on the convergence.

worse than COO when the number of users exceeds 15. The performance of SODRA and HOSRA versus users are almost the same after multiple statistical simulations. Their curves overlap, while the strategies of them are various in each simulation. TODRA, SODRA, HOSRA are the three approaches that can continuously improve the system utility with the increase of users, while TODRA are superior and maintain about 40% higher utility. The main reason is TODRA takes the preparing time into account, including image file transmission and instance startup, while the other two do not. This demonstrates that considering the preparing time is essential in the offloading model.

Fig. 7 presents the number of offloading tasks versus the number of users. As the number of users increases, the number of offloading users in COO linearly increase at first since it offloads all the tasks. When the number of users exceeds the number of sub-channels, COO can not offload more users. Due to the limited resource, the SODRA, HOSRA maintain the lower number of offloading users. TODRA is the most "rational" algorithm, which does not blindly occupy channel resource considering the cost of preparation time. As the number of users increase, computation and communication resource become scarcer, the approaches have to give up some users and select a part of users to maximize the objective.

3) Analysis of per-user metrics: **Distribution of user utility.** Fig. 8 shows the CDF of the user utility achieved by the approaches when the number of users is 30. In LCO, all the users have the same utility of 0 since there is no user to be offloaded. All the algorithms have at least two-thirds of users with 0 utility since there is only 20 sub-channels. COO randomly select 20 users for offloading, which leads to about 28% users obtaining negative utility. SODRA and HOSRA make offloading and resource allocation decision without considering the preparing time, thus suffering about 12% of users achieving negative utility. IOJRA could guarantee a positive utility for each user when offloading individually. However, when the resource is shared among all offloading users, each user's resources are reduced and original utility cannot be ensured, and the utility of some users even reduce to less than 0. TODRA offloads about 52% of user tasks and jointly schedules resource, which make all the users get a non-negative utility.

Distribution of time consumption. Fig. 9 illustrates the

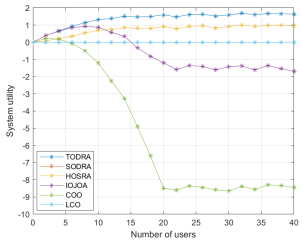


Fig. 6. System utility.

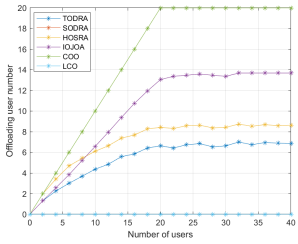


Fig. 7. Offloading user numbers.

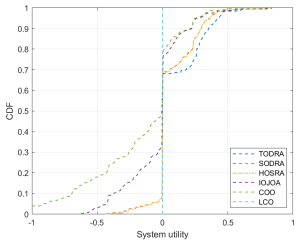


Fig. 8. CDF of system utility.

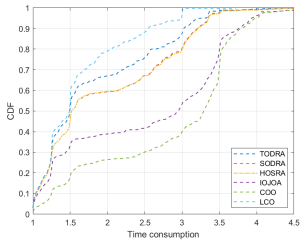


Fig. 9. CDF of time consumption.

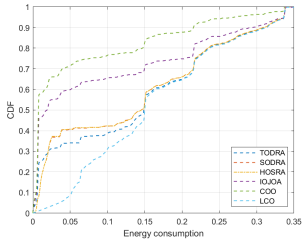


Fig. 10. CDF of energy consumption.

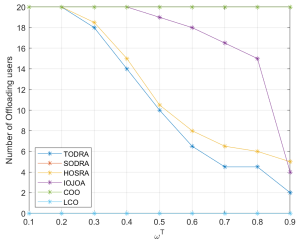


Fig. 11. Offloading user numbers with ω_i^T.

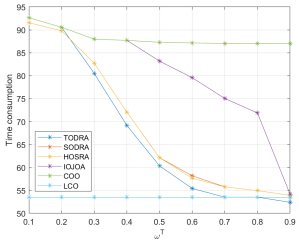


Fig. 12. Time consumption with ω_i^T.

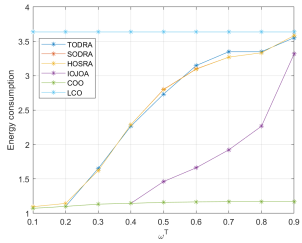


Fig. 13. Energy consumption with ω_i^T.

CDF of time consumption of users calculated by the approaches. Offloading tasks can decrease the computation delay with more sufficient resource in edge server, while it increase the transmission latency and preparing latency. CCO and IOJRA suffer the worst time cost as they offload users as many as possible. LCO has the lowest user delay at the cost of energy consumption as showed in Fig. 10. The performance of TODRA, SODRA, HOSRA are similar and are between the best and the worst. The reason is that they assign the resource from a global perspective and take the trade-off between energy and time into consideration.

Distribution of energy consumption. Fig. 10 shows the CDF of energy consumption of users. LCO has the worst performance with the highest energy consumption since offloading can save energy in general. On the contrary, COO is the most energy-saving algorithm as it offloads the most tasks. IOJRA makes offloading decisions independently as long as the remote computation has higher utility than local computation for the select task. It fails to select the user task set with better utility to offload, which results in a crossing point among TODRA and SODRA. In addition, though the performance of SODRA, HOSRA versus users are almost the same in Fig. 6 and Fig. 7, their resource allocation approach is distinct for the specific situation, which leads to the subtle difference of the distribution of energy consumption, time consumption and user utility.

4) Energy-Time Consumption Trade-off: Fig. 11, 12, 13 present the number of offloading user tasks, time consumption and energy consumption of the algorithms against different ω_i^T, respectively. We let the mean value of ω_i^T increase from 0.1 to 0.9 with N = 30, and $\omega_i^E = 1 - \omega_i^T$. With the increase of ω_i^T, users prefer lower time consumption. Since offloading tasks can save energy at the cost of time consumption, TODRA, SODRA, HOSRA offload less tasks with higher ω_i^T to reduce the user latency. Correspondingly, more users are decided to compute tasks locally, which leads to the growing of the energy consumption. For LCO, it completes all the task locally, so the time consumption and energy consumption would not change with the weighted parameter. COO always offloads 20 users. It adjusts the computation and communication resource allocation under different ω_i^T though the change of time and energy consumption is subtle. It has to mention that the IOJRA seems not to be sensitive to the change of ω_i^T when it increases from 0.1 to 0.4. This may be caused by the independent decision making policy. The task is chosen to be offloaded as long as the remote computation has higher utility than local computation until all the sub-channels are occupied, which means there is at least 20 users achieving positive single-user utility when ω_i^T is between 0.1 and 0.4.

VII. Conclusion

In this paper, the joint task offloading and resource allocation with the consideration of container startup time in mobile edge computing has been studied. To illustrate the significant impact of container startup time on overall performance, we have made a measurement study in Alibaba edge node. The offloading decision making and resource allocation is formulated as an optimization problem. To reduce the complexity, a two-layer offloading decision and resource allocation algorithm is designed. In the first layer, the Gibbs sampling is adopted to make the offloading decision. In the second layer, bisection and KKT conditions is exploited to obtain the optimal computation resource allocation and uplink power assignment scheme. Numerical simulation results show that

our proposed approach can largely reduce the latency and save the overall energy consumption over traditional approaches.

APPENDIX A
PROOF OF LEMMA 1

The offloading decision evolves as a I-dimensional Markov chain. We begin with two MDs in the system. Let $\{x_1, x_2\}$ denote the offloading decision of the two MDs, and D denote the offloading decision space. In each iteration, we change the offloading decision to$\{x_1^{\#}, x_2\}$ with the following probability:

$$\Pr(\{x_1^{\#}, x_2\}|\{x_1, x_2\}) = \frac{1}{2} \times \frac{1}{1 + e^{-(U(\{x_1^{\#}, x_2\}) - U(\{x_1, x_2\}))/\delta}} \tag{28}$$

$U(\{x_1, x_2\})$ denotes the utility function value when the offloading decision is $\{x_1, x_2\}$. Let $\Pr^*(\{x_1, x_2\})$ denote the stationary distribution for each state. We can derive from the stationary condition of Markov chain that:

$$\begin{aligned} &\Pr^*(\{x_1, x_2\}) \Pr(\{x_1, x_2^{\#}\}|\{x_1, x_2\}) \\ &= \Pr^*(\{x_1^{\#}, x_2\}) \Pr(\{x_1, x_2\}|\{x_1^{\#}, x_2\}) \end{aligned} \tag{29}$$

By substituting (1) into (2), we have:

$$\begin{aligned} &\frac{1}{2} \times \frac{1}{1 + e^{-(U(\{x_1^{\#}, x_2\}) - U(\{x_1, x_2\}))/\delta}} \times \Pr^*(\{x_1, x_2\}) \\ &= \frac{1}{2} \times \frac{1}{1 + e^{-(U(\{x_1, x_2\}) - U(\{x_1^{\#}, x_2\}))/\delta}} \times \Pr^*(\{x_1^{\#}, x_2\}) \end{aligned} \tag{30}$$

$$\begin{aligned} &\frac{1}{2} \times \frac{e^{U(\{x_1^{\#}, x_2\})/\delta}}{e^{U(\{x_1^{\#}, x_2\})/\tau} + e^{(U(\{x_1, x_2\}))/\delta}} \times \Pr^*(\{x_1, x_2\}) \\ &= \frac{1}{2} \times \frac{e^{U(\{x_1, x_2\})/\delta}}{e^{U(\{x_1, x_2\})/\tau} + e^{(U(\{x_1^{\#}, x_2\}))/\delta}} \times \Pr^*(\{x_1^{\#}, x_2\}) \end{aligned} \tag{31}$$

Observing that the equation is symmetry. Therefore, the stationary distribution is:

$$\Pr^*(\{x_1, x_2\}) = \Upsilon e^{U(\{x_1, x_2\})/\delta} \tag{32}$$

Based on the probability conservation law, we have:

$$\sum_{\{x_1^{\#}, x_2^{\#}\} \in D} \Pr^*(\{x_1^{\#}, x_2^{\#}\}) = 1 \tag{33}$$

Therefore, the stationary distribution is:

$$\Pr^*(\{x_1, x_2\}) = \frac{e^{U(\{x_1, x_2\})/\delta}}{\sum\limits_{\{x_1^{\#}, x_2^{\#}\} \in D} e^{U(\{x_1^{\#}, x_2^{\#}\})/\delta}} \tag{34}$$

Let $\{x_1^*, x_2^*\}$ denotes the globe optimal state. The stationary distribution of $\{x_1^*, x_2^*\}$ is given by:

$$\begin{aligned} \Pr^*(\{x_1, x_2\}) &= \frac{e^{U(\{x_1^*, x_2^*\})/\delta}}{\sum\limits_{\{x_1^{\#}, x_2^{\#}\} \in D} e^{U(\{x_1^{\#}, x_2^{\#}\})/\delta}} \\ &= \sum_{\{x_1^{\#}, x_2^{\#}\} \in D} e^{\frac{U(\{x_1^*, x_2^*\}) - U(\{x_1^{\#}, x_2^{\#}\})}{\delta}} \end{aligned} \tag{35}$$

Because $U(\{x_1^{\#}, x_2^{\#}\}) \leq U(\{x_1^*, x_2^*\})$, we can observe that $\Pr^*(\{x_1, x_2\})$ increases with the decrease of δ, and $\lim\limits_{\delta \to 0} \Pr^*(\{x_1, x_2\}) = 1$. The above analysis can be straightforwardly extends to I-dimensional Markov chain. We can make the conclusion that the algorithm converges to the optimal solution with probability 1 when the value of δ tends to 0.

APPENDIX B
PROOF OF LEMMA 2

The domain of P1 is convex. We denote $\Phi(F)$ as the optimization function of P1. Then, the second-order derivative of $\Phi(F)$ is given as:

$$\frac{\partial^2 \Phi(F)}{\partial f_i^{Edge} f_j^{Edge}} = \begin{cases} \dfrac{2\omega_i^T f_i^{Local}}{\left(f_i^{Edge}\right)^3} > 0, i \neq j \\ 0, i == j \end{cases} \tag{36}$$

The Hessian matrix of $\Phi(F)$ is diagonal, and its elements are larger than 0. Therefore, the Hessian matrix of $\Phi(F)$ is positive definite, and $\Phi(F)$ is convex. We can conclude that P1 is convex optimization.

APPENDIX C
PROOF OF LEMMA 3

For simplicity, we define two variable as follows:

$$y = \frac{\omega_i^E b_i}{\alpha c_i (f_i^{Local})^{\gamma\text{-}1} W} \tag{37}$$

$$z = \frac{g_i}{N_0} \tag{38}$$

$\Gamma_3(p_i)$ can be simplified as :

$$\Gamma_3(p_i) = \frac{\omega_i^T f_i^{Local} T_i^{S,Edge}}{c_i} + \frac{y p_i}{\log_2(1 + z p_i)} \tag{39}$$

$\Gamma_3(p_i)$ is twice differentiable in the domain. We now check that for a point $\hat{p}_i$ satisfies $\Gamma_3(\hat{p}_i)' = 0$ implies$\Gamma_3(\hat{p}_i)'' \geq 0$. The first-order derivative of $\Gamma_3(p_i)$ is:

$$\Gamma_3(p_i)' = \frac{y \log_2(1 + z p_i) - y p_i \frac{z}{\ln 2(1 + z p_i)}}{\left(\log_2(1 + z p_i)\right)^2} \tag{40}$$

When $\Gamma_3(\hat{p}_i)' = 0$, we can obtain:

$$y \log_2(1 + z\hat{p}_i) - y p_i \frac{z}{\ln 2(1 + z\hat{p}_i)} = 0 \tag{41}$$

$$\begin{aligned} \Gamma_3(p_i)'' = &\frac{\left(\frac{y z^2 p_i}{\ln 2(1 + z p_i)^2}\right) \left(\log_2(1 + z p_i)\right)^2}{\left(\log_2(1 + z p_i)\right)^4} \\ &- \frac{\left(y \log_2(1 + z p_i) - y p_i \frac{z}{\ln 2(1 + z p_i)}\right) \frac{2 z \log_2(1 + z p_i)}{\ln 2(1 + z p_i)}}{\left(\log_2(1 + z p_i)\right)^4} \end{aligned} \tag{42}$$

The second-order derivative of $\Gamma_3(p_i)$ is shown in (42). Substituting $\hat{p}_i$ into (42), we can obtain:

$$\Gamma_3(\hat{p}_i)'' = \frac{y z^2 \hat{p}_i}{\left(\log_2(1 + z\hat{p}_i)\right)^2 \ln 2(1 + z\hat{p}_i)^2} > 0 \tag{43}$$

Therefore, $\Gamma_3(P)$ is quasiconvex in the domain.

ACKNOWLEDGMENT

This work was supported in part by National Key R&D Program of China (2020YFB1805502), NSFC (61922017 and 61921003).

REFERENCES

[1] S. Mao, S. Leng, S. Maharjan, and Y. Zhang, "Energy efficiency and delay tradeoff for wireless powered mobile-edge computing systems with multi-access schemes," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 1855–1867, 2019.

[2] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 4, pp. 3435–3447, 2017.

[3] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, 2019.

[4] 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, pp. 2795–2808, October 2016.

[5] M. Nir and A. Matrawy, "Centralized management of scalable cyber foraging systems," *Procedia Computer Science*, vol. 21, no. 1, pp. 265 – 273, 2013.

[6] F. Xia, F. Ding, J. Li, X. Kong, L. T. Yang, and J. Ma, "Phone2cloud: Exploiting computation offloading for energy saving on smartphones in mobile cloud computing," *Information Systems Frontiers*, vol. 16, pp. 95–111, Mar 2014.

[7] J. Zheng, Y. Cai, Y. Wu, and X. Shen, "Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach," *IEEE Transactions on Mobile Computing*, vol. 18, pp. 771–786, April 2019.

[8] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, pp. 974–983, April 2015.

[9] J. P. V. Champati and B. Liang, "Delay and cost optimization in computational offloading systems with unknown task processing times," *IEEE Transactions on Cloud Computing*, vol. pp, pp. 1–12, Dec 2019.

[10] H. Wu, Y. Sun, and K. Wolter, "Energy-efficient decision making for mobile cloud offloading," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2018.

[11] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. S. Shen, "Toffee: Task offloading and frequency scaling for energy efficiency of mobile devices in mobile edge computing," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2019.

[12] L. Yang, H. Zhang, X. Li, H. Ji, and V. C. M. Leung, "A distributed computation offloading strategy in small-cell networks integrated with mobile edge computing," *IEEE/ACM Transactions on Networking*, vol. 26, pp. 2762–2773, Dec 2018.

[13] C. Yi, J. Cai, and Z. Su, "A multi-user mobile computation offloading and transmission scheduling mechanism for delay-sensitive applications," *IEEE Transactions on Mobile Computing*, vol. 19, pp. 29–43, Jan 2020.

[14] F. Guo, H. Zhang, H. Ji, X. Li, and V. C. M. Leung, "An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing," *IEEE/ACM Transactions on Networking*, vol. 26, pp. 2651–2664, Dec 2018.

[15] J. Meng, H. Tan, C. Xu, W. Cao, L. Liu, and B. Li, "Dedas: Online task dispatching and scheduling with bandwidth constraint in edge computing," *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pp. 2287–2295, April 2019.

[16] T. Harter, B. Salmon, R. Liu, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Slacker: Fast distribution with lazy docker containers," *14th USENIX Conference on File and Storage Technologies (FAST 16)*, pp. 181–195, 2016.

[17] X. Zhang, Y. Mao, J. Zhang, and K. B. Letaief, "Multi-objective resource allocation for mobile edge computing systems," *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1–5, 2017.

[18] T. Zhang, Y. Wang, Y. Liu, W. Xu, and A. Nallanathan, "Cache-enabling uav communications: Network deployment and resource allocation," *IEEE Transactions on Wireless Communications*, 2020.

[19] U. Yaqub and S. Sorour, "Multi-objective resource optimization for hierarchical mobile edge computing," *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2018.

[20] W. Zhang, Y. Wen, and D. O. Wu, "Collaborative task execution in mobile cloud computing under a stochastic wireless channel," *IEEE Transactions on Wireless Communications*, vol. 14, no. 1, pp. 81–93, 2015.