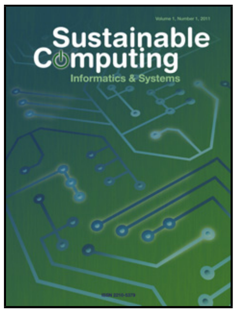

Title: Energy-Efficient Computation Offloading and Resource Allocation for Delay-sensitive Mobile Edge Computing

Author: Quyuan Wang Songtao Guo Jiadi Liu Yuanyuan Yang

PII: S2210-5379(18)30078-7
DOI: https://doi.org/doi:10.1016/j.suscom.2019.01.007
Reference: SUSCOM 305

To appear in:

Received date: 27 February 2018
Revised date: 15 July 2018
Accepted date: 9 January 2019

Energy-Efficient Computation Offloading and Resource Allocation for Delay-sensitive Mobile Edge Computing

Quyuan Wang[a], Songtao Guo[a,*], Jiadi Liu[a], Yuanyuan Yang[a,b,*]

[a]*Chongqing Key Laboratory of Nonlinear Circuits and Intelligent Information Processing, College of Electronic and Information Engineering, Southwest University, Chongqing, 400715, China*
[b]*Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794, USA*

Abstract

Mobile edge computing is an emerging computing paradigm to augment computational capabilities of mobile devices by offloading computation-intensive tasks from resource-constrained smart mobile device onto edge clouds nearby with potential computation capability. However, in general, edge clouds have limited computation resource and energy. Thus it is critical to achieve high energy efficiency while ensuring satisfactory user experience. In this paper, we first formulate the computation offloading problem into the system energy-efficiency cost minimization problem by taking into account the completion time and energy. Then, by solving the optimization problem, we propose a distributed algorithm consisting of clock frequency configuration, transmission power allocation, channel rate scheduling and offloading strategy selection. In this algorithm, the clock frequency for local execution and the transmission power in edge cloud execution are optimized jointly. Furthermore, to optimize the queue delay, we formulate an M/M/n queue model with individual capacity between different windows and propose an algorithm for optimal task offloading rate. Finally, the experimental results show that our algorithm can achieve energy-efficient offloading performance compared to other existing algorithms.

Keywords: Mobile edge computing, Energy efficiency, Computation Offloading, Resource Allocation, Distributed algorithm.

1. Introduction

In the era of mobile computing and Internet of Things, mobile devices have become an essential part of modern life. The widespread used of smart mobile devices also has contributed to the development of computation-intensive mobile applications with advanced features. What we face now is the contradiction between inadequate processing capacity of edge devices and the users' ever-growing needs for better performance. Mobile edge computing (MEC) is such a promising technology to solve the conflicts between the computation-intensive applications and the resource-limited mobile devices by shortening the distance between users and clouds [1–5]. Compared to Mobile cloud computing (MCC) [6–8], MEC has some obvious advantages. For instance, it is able to shorten the distance among data transmission, pre-process the complex tasks before offloading

*Corresponding author.
Email addresses: songtao_guo@163.com (Songtao Guo), yuanyuan.yang@stonybrook.edu (Yuanyuan Yang)

to save the cost, and offer significantly small jitter among offloading. The main objective of MEC is to reduce latency by transferring the computation and storage capacity from the core Wide Area Network (WAN) to the edge network. The direct interaction between mobile devices and edge clouds brings ultra-low latency and longer battery lifetime of mobile devices.

Although MEC transfers the computations to the edge cloud and utilizes task partitioning and offloading technique to enhance the performance significantly [9–14], it still faces some challenges for realizing more reasonable MEC system. On the one hand, since there may exist multiple edge clouds nearby mobile users in MEC system, it is essential to make the appropriate offloading selection among the edge clouds as well as local device and central cloud. On the other hand, the energy efficiency has always been the main concern in MEC research. In MEC, mobile devices often need to share the resources of edge clouds with others to carry out their tasks. Thus, it is of importance how to balance the energy consumption among mobile devices. In addition, since there are a large number of tasks in MEC system need to be offloaded under the constraint of limited channel resources, therefore, energy saving in the transmission is also critical for improving system energy efficiency.

In MEC framework, the core of computation offloading is making decision on which computation tasks should be offloaded and how to schedule the limited resources to ensure offloading successfully. Some previous works [9–11, 15, 16] only considered that a task can be executed on mobile devices (local execution), or be offloaded onto the MEC server (edge cloud execution). However, these works ignored the central cloud in MEC system. Because edge clouds have the limited computation and storage capacity, they may consume more energy and bring additional latency when the task is too complicated or its data size is too large. This motivates us to not only utilize edge clouds but also take advantage of the central cloud to improve the system performance. In particular, in our work, the central cloud only plays the auxiliary role in computation offloading. When the task is too complicated or there is no available edge cloud nearby mobile device, the task will be offloaded onto central cloud to meet user's needs. However, if the connection between the mobile device and the central cloud is lost, the mobile device will locally execute tasks with the help of edge clouds or wait for reconnection within the tolerable time. Hence different from the previous work, our offloading selection strategy has three selection objections, i.e., edge cloud, local device and central cloud.

To achieve cost-saving offloading or energy-efficient offloading, there are several works using Game theory or joint allocation on radio and computational resources [10, 13, 17, 18], which mainly focus on CPU frequency and data rate. However, in such an era of frequent communication, not only the computing resource is shortage, but also the communication resource is limited, such as frequency spectrum. If there are too many tasks to be offloaded onto the edge cloud over the same spectrum/channel at the same time, it will generate confliction/congestion, which may lead to fast drop in task offloading rate. Therefore, it is critical how to exploit limited communication resource to schedule more tasks. On the basis of the above considerations, we apply the queuing theory to optimize the task offloading rate so as to reduce the queuing delay. In order to make reasonable offloading decision and achieve energy efficiency, we focus on the following issues in this work:

1. How to make an optimal decision on selecting local execution or cloud computing. That is to make an optimal decision for reducing total energy-efficiency cost, which is the weighted sum of completion time and energy consumption .

2. How to schedule local resources for local execution and allocate cloud and channel resources for cloud computing.
3. How to make full use of existing channels to ensure that tasks are offloaded quickly and effectively.

The objective of this paper is to propose an energy-efficient offloading strategy to minimize total cost by optimizing offloading selection and resource allocation in mobile edge computing system under the assistance of central cloud. Here, the cost includes energy consumption and processing time. Compared with previous work, our contributions in this paper can be summarized as follows:

- We propose a task offloading and resource scheduling mechanism with three offloading destinations, i.e., local device, edge cloud and central cloud. Since there are both edge clouds and central clouds in the real environment, it is much essential to utilize the central cloud to help dealing with complex tasks.

- We formulate the energy-efficient offloading problem into a cost minimization problem by considering application completion deadline and processing capability constraints. For solving the optimization problem, we propose a distributed algorithm containing strategy selection, clock frequency control and transmission power allocation.

- We introduce the queueing theory to formulate an M/M/n queue model with individual capacity between different windows. We give the optimal task offloading rate and queue delay by using convex optimization method.

- Experimental results show that compared with other existing algorithms, our task offloading and resource scheduling algorithm can reduce about 30% of the cost. Moreover, we find that the cost does not increase dramatically with the increase of the number of tasks and input data size.

To the best of our knowledge, this work is the first work on dynamic task offloading and resource scheduling that minimizes total cost taking into account both the CPU clock frequency control on local device, and the transmission power allocation and the task offloading rate on mobile edge computing with the assistance of the central cloud.

The rest of the paper is organized as follows. In Section 2 we review the related works on mobile cloud computing and the mobile edge computing. Section 3 proposes an offloading scheme, constructs the system model and formulates the optimization problem. In Section 4, we propose the distributed algorithm by solving the optimization problem based on convex method and queue theory. In Section 5, we conduct the extensive experiments to evaluate our proposed algorithms. Finally, Section 6 concludes this paper and gives our future works.

2. Related Work

Computation offloading for MEC system has attracted significant attention in recent years [9, 10, 12, 13, 16, 17, 19–24]. In[19], You et al. proposed a resource allocation policy based on time division multiple access (TDMA) and orthogonal frequency-division multiple access (OFDMA) to minimize the mobile energy consumption for MEC system. In [20], Drolia et al. studied the tradeoff between offloading computation to an infrastructure cloud and retaining the computation within a mobile edge cloud. Wang et al. studied the dynamic service migration problem at the network edge and utilized Markov Decision Process to formulate a sequential decision making problem in [21]. However, the above works only concentrate on the offloading rather than the joint optimization on communication and computation resources.

Furthermore, there have been some works on the joint optimization of radio and computational resources MEC system. In [9], Zhao et al. jointly optimized the offloading selection, radio resource allocation, and computational resource allocation. They also formulated the energy consumption minimization problem as a mixed integer nonlinear programming (MINLP) problem, which is subject to specific application latency constraints. Mao et al. in [10] developed an online joint radio and computational resource management algorithm for multi-user MEC systems, which aims at minimizing the long-term average weighted sum power consumption of the mobile devices and the MEC server, subject to the task buffer stability constraint to make offloading more effective. Dinh et al. in [13] aimed to minimize both total task execution latency and the mobile device energy consumption by jointly optimizing the task allocation decision and the mobile devices' CPU frequency. In [17], Chen et al. proposed an efficient computation offloading algorithm that can achieve Nash equilibrium by using game theory approach. Although the above works give more attention on radio and computation resources, they neither consider the role of the central cloud, nor address how to allocate channels with limited channel resources.

Different from the previous works, our paper gives a queue model to optimize the task offloading rate so as to minimize the queue delay while optimizing the clock frequency and transmission power to minimize the energy consumption and latency with the help of the central cloud for MEC system.

3. System Model and Problem Formulation

In this section, we first introduce the system model. We assume that the set $\mathcal{J} = \{1, 2, ..., J\}$ denotes computation-intensive tasks. As shown in Fig.1, our MEC system consists of some mobile phones, multiple access points or base stations and one central cloud. In our MEC system, the edge cloud includes the fixed edge gateway, e.g. base stations, servers in buildings and powerful processing equipment nearby, which assists mobile devices to offload computing tasks, and some mobile phones with abundant resources. Other mobile phones with limited resources is referred to as clients, which can send their requests and offload their tasks onto central cloud through wireless access points. The clients can also offload their tasks onto edge cloud if their connections are available. We assume that there is only one central cloud which is used to help edge clouds compute some complex tasks. It is worth noting that in our study, we just consider that the clients maintain a low speed motion relative to the edge clouds to ensure stable connection. And we assume that the application has already been partitioned into several tasks using previous technologies [25]. The parameters used in the paper are listed in Table 1. In the following, we will introduce the task offloading process and computing model.

3.1. Task Offloading Scheme

In this paper, we assume that a client can access multiple edge clouds and one central cloud over different channels. Let M denote the total number of channels. And all channels are employed in a first-come first-serve (FCFS) manner. In our scheme, after a new service request arrives, there

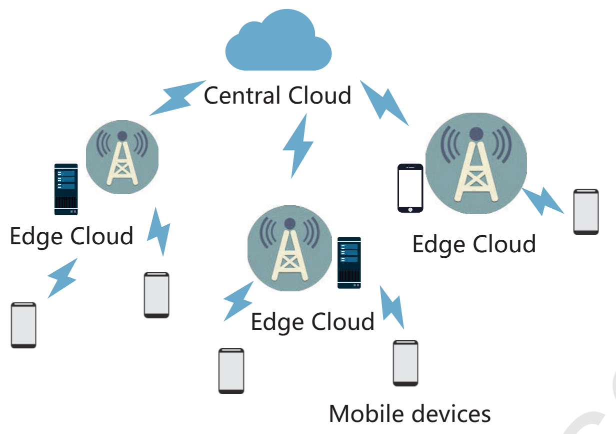

Figure 1: Illustration of MEC.

Table 1: Partial system parameters

Notation	Description
J	number of tasks
M	number of channels
f_i	clock frequency of task i
p_i	transmission power of task i
h_i	channel gain of task i
Cy_i	total CPU cycles of task i
d_i	required data size of task i
T_i	completion time of task i
E_i	energy consumption of task i
$Cost_i$	total cost of task i
α_i	offloading selection factor
β_i	cloud selection factor
r_i	uplink data rate
W	channel bandwidth
N	noise power and interference
$T_{i,q}^m$	queueing delay at channel m of task i
$\delta_{i,T}$	the weight of time of task i
$\delta_{i,E}$	the weight of energy of task i
κ	effective switched capacitance
λ^m	deliver rates on channel m

will be some information exchanges between edge clouds and local device. The edge clouds will get some task information of mobile device, i.e., input data size, the type of task and the maximum latency the mobile device can tolerate, and then send back the their information to mobile device. Based on the information about the edge clouds, the mobile device will analyze which edge cloud is optimal when the tasks are offloaded. Once the optimal edge cloud is selected, we will compare local task execution with the best edge cloud execution to determine whether the task is offload to edge cloud or not. If there is no edge cloud to handle the task and it is beneficial (cost-saving) to offload the task to the central cloud, the local device will connect to the central cloud. It is worth noting that we monitor and allocate the channel to the central cloud by the CSMA/CA protocol for 802.11 wireless LANs [26]. When the channel condition cannot guarantee user needs (i.e. there is a task waiting within the channel), we optimize the task offloading rate to minimize the queuing time. Fig. 2 illustrates the processes of our task offloading scheme.

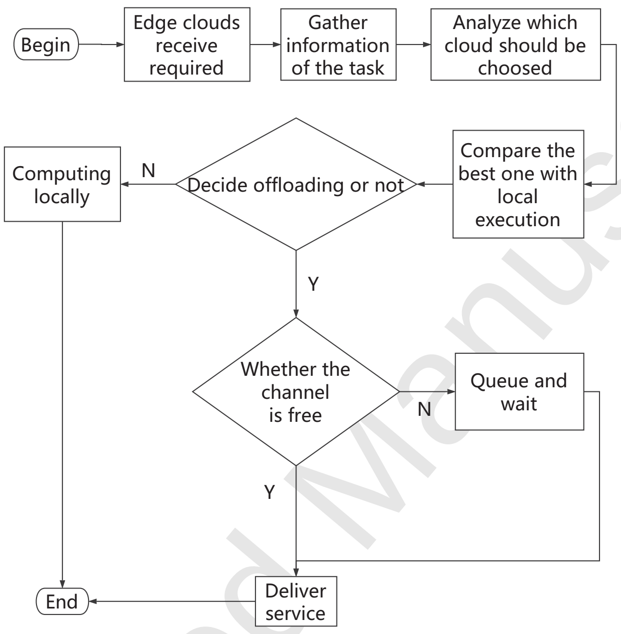
Begin
Edge clouds receive required
Gather information of the task
Analyze which cloud should be choosed
Compare the best one with local execution
Decide offloading or not
N
Computing locally
Y
Whether the channel is free
N
Queue and wait
Y
Deliver service
End

Figure 2: The flowchart of our task offloading scheme.

3.2. Local Computing Model

Intuitively, the tasks with smaller size may be executed on local device. Let clock frequency $f_{i,l}$ denote the computation capability of local computing, i.e., processing data size per second. It is known that smart devices can change their clock frequency to deal with different tasks to save energy by dynamic voltage and frequency scaling (DVFS) technique [27]. The execution time of task i by local computing can be expressed as

$$T_{i,l} = \frac{Cy_{i,l}}{f_{i,l}} \tag{1}$$

and the energy consumption is given by

$$E_{i,l} = \kappa Cy_{i,l} f_{i,l}^2 \tag{2}$$

where $Cy_{i,l}$ is the total CPU cycles of computing task i on local device, and κ denotes the effective switched capacitance depending on the chip architecture. In this paper, we set $\kappa = 10^{-11}$ so that the energy consumption is consistent with the measurements in [28]. The expression of local computing cost is derived as follows:

$$Cost_{i,l} = \delta_{i,T} T_{i,l} + \delta_{i,E} E_{i,l} \tag{3}$$

where $0 \leq \delta_{i,T} \leq 1$ and $0 \leq \delta_{i,E} \leq 1$ represent the wights of completion time and energy consumption for task i. In general, we assume that $\delta_{i,T} + \delta_{i,E} = 1$. Clearly, we can control the clock frequency of mobile device to minimize local computing cost.

3.3. Cloud Computing Model

A task may be computed on the edge cloud or central cloud via offloading, which is called cloud computing. Moreover, if we offload the task onto the central cloud, the data offloaded may contain some virtual machine (VM) information to match our task. For this reason the data size of the same task transferred to edge cloud or central cloud may be different. We use $d_{i,e}$ and $d_{i,t}$ to denote the data size on edge cloud and central cloud, respectively. The data size for cloud computing can be expressed by

$$d_{i,c} = \beta_i d_{i,e} + (1 - \beta_i) d_{i,t} \tag{4}$$

where β_i can be regarded as *Cloud selection factor*, which is used to choose edge clouds or central cloud. According to Shannon formula, furthermore, we can compute the data rate for offloading task i as

$$r_i = W log_2(1 + \frac{p_i h_i}{N + \sum_{j \in \mathcal{J} \setminus \{i\} : \alpha_i = \alpha_j} p_j h_j}) \tag{5}$$

where p_i is the transmission power of mobile device offloading task i, and h_i denotes the channel gain between mobile device and cloud. N denotes the thermal noise power and the remaining part of the denominator represents mutual interference power between users. W represents channel bandwidth. As aforementioned, $f_{i,e}$ and $f_{i,t}$ denote the computation capability of edge cloud and central cloud, respectively. Thus the completion time and transmission energy consumption at edge cloud can be computed by

$$T_{i,e} = \frac{d_{i,e}}{r_i} + \frac{C y_{i,e}}{f_{i,e}} + T_{i,q}^m, \tag{6}$$

$$E_{i,e} = (\frac{d_{i,e}}{r_i} + T_{i,q}^m) p_i + \kappa C y_{i,e} f_{i,e}^2, \tag{7}$$

respectively. It is known that edge cloud is also resource-restricted. Thus it is necessary to consider the computational energy consumption of edge cloud, which can be calculated by (2). The completion time and transmission energy consumption at central cloud can be given by

$$T_{i,t} = \frac{d_{i,t}}{r_i} + \frac{C y_{i,t}}{f_{i,t}} + T_{i,q}^m, \tag{8}$$

$$E_{i,t} = (\frac{d_{i,t}}{r_i} + T_{i,q}^m) p_i, \tag{9}$$

respectively, where $T_{i,q}^m$ is the queue waiting time of task i on channel m. It is worth noting that in this paper we do not consider the computation energy consumption on cloud. In the following, we first give the definition of energy-efficiency cost at edge cloud and central cloud by

$$Cost_{i,e} = \delta_{i,T} T_{i,e} + \delta_{i,E} E_{i,e}, \tag{10}$$

$$Cost_{i,t} = \delta_{i,T} T_{i,t} + \delta_{i,E} E_{i,t}, \tag{11}$$

respectively.

Thus the unified energy-efficiency cost in cloud computing can be given by

$$Cost_{i,c} = \beta_i Cost_{i,e} + (1 - \beta_i) Cost_{i,t} \tag{12}$$

3.4. Problem Formulation

The total energy-efficiency cost at local execution and cloud execution can be given by

$$Cost_i = \alpha_i Cost_{i,c} + (1-\alpha_i)Cost_{i,l} \tag{13}$$

where α_i denotes *Offloading selection factor* making a decision on local computing or cloud computing. For an application with the set of tasks $\mathcal{J} = \{1, 2, ..., J\}$, we aim to provide an optimal strategy profile $\alpha^* = \{\alpha_1^*, \alpha_2^*, ..., \alpha_J^*\}$ and $\beta^* = \{\beta_1^*, \beta_2^*, ..., \beta_J^*\}$, the optimal frequency control on mobile device $f_{i,l}^*$, the optimal transmission power p_i^* and the optimal queue delay $T_{i,q}^*$ to achieve the minimum total energy-efficiency cost. Thus our total energy-efficiency cost minimization problem can be formulated as:

$$min \sum_{i=1}^{J} Cost_i \tag{14}$$

such that the following constraints should be satisfied: $\forall i \in \{1, 2, ..., J\}$
(1) Time constraint:

$$\sum_{i=1}^{J} \alpha_i[\beta_i T_{i,e} + (1-\beta_i)T_{i,t}] + (1-\alpha_i)T_{i,l} < T_{max} \tag{15}$$

(2) Energy constraint:

$$\sum_{i=1}^{J} \alpha_i[\beta_i E_{i,e} + (1-\beta_i)E_{i,t}] + (1-\alpha_i)E_{i,l} < E_{max} \tag{16}$$

(3) Selection constraint:

$$\alpha_i \in \{0,1\}, \beta_i \in \{0,1\} \tag{17}$$

(4) Processing capability constraint:

$$f_{i,e} < f_{i,t} \tag{18}$$

The time constraint (15) implies that the total completion time of all tasks is lower than the required maximum application completion time T_{max}. The maximum completion time is determined by the maximum delay that the mobile user can tolerate. The Energy constraint (16) reflects that the total energy consumption of all tasks is lower than the maximum application energy limitation E_{max}. The selection constraint (17) specifies that task i is executed on the local or the cloud. The processing capability constraint (18) follows that the processing capability of edge cloud is limited, compared with central cloud.

The critical restriction in solving the optimization problem in (14) is that the selection constraint in (17) contains two integer variables, $\alpha_i \in \{0,1\}, \beta_i \in \{0,1\}$, which makes the optimization problem become NP-hard and non-convex because it is a mixed integer programming problem. In order to transform the non-convex problem to the convex one, we relax the selection factors to a real number between zero and one, i.e., $0 \leq \alpha_i \leq 1, 0 \leq \beta_i \leq 1$ similar to the method in [30].

Theorem 1. *The optimization problem (14) with constraints (15)-(18) is convex with respect to (w.r.t) the optimization variables α_i, β_i, $f_{i,l}$, p_i and λ_m.*

Proof. Please refer to Appendix A. ■

Theorem 1 implies that the problem (14) has a zero duality gap and satisfies the Slaters constraint qualification. Therefore, we will solve the optimization problem to give the optimal solutions in the next section.

4. Distributed Algorithm

In this section, we will propose a distributed algorithm consisting of clock frequency configuration, transmission power allocation, channel rate scheduling and offloading strategy selection. "Distributed algorithm" means that the offloading decision algorithm runs independently on local devices, that is to say, the local devices can decide whether and where to offload the task independently rather than being determined by centralized controller such as edge cloud and central cloud.

4.1. Processing Capability Optimization

The completion time and energy consumption are associated with CPU clock frequency $f_{i,l}$, if we execute the task on local mobile, *i.e.*, $\alpha_i = 0$. Processing capability optimization aims to optimize CPU clock frequency so as to minimize the total cost when we choose local computing. The cost minimization problem in (14) can be rewritten as:

$$\min_{f_{i,l}} \sum_{i=1}^{J} \delta_{i,T} \frac{Cy_{i,l}}{f_{i,l}} + \delta_{i,E} C y_{i,l} \kappa f_{i,l}^2 \tag{19}$$

$$s.t. \quad \sum_{i=1}^{J} f_{i,l} < J \cdot F \tag{20}$$

where F is the minimum capability of edge clouds. We can find the optimization problem is convex, thus we can employ the convex method [31] to solve this problem. The the dual function of optimization problem (19) can be given by

$$L(f_{i,l}, \theta) = \sum_{i=1}^{J} (\delta_{i,T} \frac{Cy_{i,l}}{f_{i,l}} + \delta_{i,E} C y_{i,l} \kappa f_{i,l}^2) + \theta(\sum_{i=1}^{J} f_{i,l} - JF) \tag{21}$$

According to KKT conditions [31], we can get the following equation,

$$\frac{\partial L}{\partial f_{i,l}} = -\delta_{i,T} \frac{Cy_{i,l}}{f_{i,l}^2} + 2\delta_{i,E} C y_{i,l} \kappa f_{i,l} + \theta = 0 \tag{22}$$

Furthermore, Eq. (22) can be rewritten as

$$2\delta_{i,E} C y_{i,l} \kappa f_{i,l}^3 + \theta f_{i,l}^2 - \delta_{i,T} C y_{i,l} = 0 \tag{23}$$

Then we have

$$f_{i,l}^3 + \frac{\theta}{2\delta_{i.E} C y_{i,l} \kappa} f_{i,l}^2 - \frac{\delta_{i,T}}{2\delta_{i,E} \kappa} = 0 \tag{24}$$

We let $w_1 = \frac{\theta}{2\delta_{i,E} C y_{i,l} \kappa}, w_2 = 0, w_3 = -\frac{\delta_{i,T}}{2\delta_{i,E}\kappa}$, then we can get,

$$f_{i,l}^3 + w_1 f_{i,l}^2 + w_2 f_{i,l} + w_3 = 0 \tag{25}$$

According to the literature [32], we can obtain such a conclusion that if $w_3 < 0$, (25) has at least one positive root. Substituting $f_{i,l} = y - \frac{w_1}{3}$ into (25), we can eliminate the quadratic term and obtain

$$y^3 - \frac{1}{3} w_1^2 y + \frac{2}{27} w_1^3 + r_3 = 0 \tag{26}$$

which can be rewritten as

$$y^3 + ay + b = 0 \tag{27}$$

where $a = -\frac{1}{3} w_1^2, b = \frac{2}{27} w_1^3 + w_3$. Let $y = u + v$ and substitute it into (27), and we have

$$(u^3 + v^3 + b) + (u + v)(3uv + a) = 0 \tag{28}$$

Then, we can obtain u and v, which satisfy the following conditions

$$\begin{cases} u^3 + v^3 = -b \\ (uv)^3 = -\frac{a^3}{27} \end{cases} \tag{29}$$

It is obvious that u^3 and v^3 are two roots of the following equation

$$z^2 + bz - \frac{a^3}{27} \tag{30}$$

By solving (30), if the following condition holds

$$\Delta = b^2 + \frac{4a^3}{27} > 0 \tag{31}$$

we have $u = \sqrt[3]{\frac{-b+\sqrt{\Delta}}{2}}$ and $v = \sqrt[3]{\frac{-b-\sqrt{\Delta}}{2}}$. Therefore, the optimal CPU clock frequency can be given by

$$f_{i,l}^* = u + v - \frac{1}{3} w_1 \tag{32}$$

For a given $f_{i,l}$, we can update the Lagrange multiplier θ by

$$\theta(k+1) = [\theta(k) + \epsilon(k)(\sum_{i=1}^{J} f_{i,l} - JF)]^+ \tag{33}$$

where index $k > 0$ is the iteration index and $\epsilon(k)$ is positive iteration step size. Therefore, the Lagrange multiplier in (33) can be used for updating the clock frequency in (32).

4.2. Transmission Power Allocation

If a task is offloaded onto cloud, the offloading rate r_i is determined by transmission power p_i. Thus the cost minimization problem in (14) can be transformed into the following transmission power allocation problem:

$$\min_{p_i} \sum_{i=1}^{J} \delta_{i,T} \frac{d_{i,c}}{r_i} + p_i(\delta_{i,E} \frac{d_{i,c}}{r_i} + \delta_{i,T} T_{i,q}^m) \tag{34}$$

It is clear that the optimization problem (34) is convex about transmission power p_i. We let

$$L(p_i) = \sum_{i=1}^{J} \delta_{i,T} \frac{d_{i,c}}{r_i} + p_i(\delta_{i,E} \frac{d_{i,c}}{r_i} + \delta_{i,T} T_{i,q}^m) \tag{35}$$

According to KKT conditions [31], we take the derivation of $L(p_i)$ about p_i and have

$$\begin{aligned} \frac{dL}{dp_i} = \frac{d_{i,c}}{W} & \frac{\frac{\frac{h_i}{N}(p_i\delta_{i,E}+\delta_{i,T})}{(1+\frac{p_ih_i}{N})ln2} - \delta_{i,E} ln(1+\frac{p_ih_i}{N})}{[log_2(1+\frac{p_ih_i}{N})]^2} \\ & + \frac{\delta_{i,T} T_{i,q}^m [log_2(1+\frac{p_ih_i}{N})]^2}{[log_2(1+\frac{p_ih_i}{N})]^2} = 0 \end{aligned} \tag{36}$$

which can be rewritten as

$$\begin{aligned} \frac{\delta_{i,E}\frac{p_ih_i}{N} + \delta_{i,T}\frac{h_i}{N}}{1+\frac{p_ih_i}{N}} = \delta_{i,E} ln(1+\frac{p_ih_i}{N}) - \\ \frac{\delta_{i,T} T_{i,q}^m}{ln2}[ln(1+\frac{p_ih_i}{N})]^2 \end{aligned} \tag{37}$$

It is not difficult to observe that the optimal transmission power is the solution of (37). i.e., the intersection point between two equations, $g(p_i)$ and $G(p_i)$, which can be given by

$$g(p_i) = \frac{\delta_{i,E}\frac{p_ih_i}{N} + \delta_{i,T}\frac{h_i}{N}}{1+\frac{p_ih_i}{N}},$$

and

$$G(p_i) = \delta_{i,E} ln(1+\frac{p_ih_i}{N}) - \frac{\delta_{i,T} T_{i,q}^m}{ln2}[ln(1+\frac{p_ih_i}{N})]^2.$$

However, Eq. (37) is a transcendental equation and in general does not have closed-form solution about p_i. Thus we have to utilize the Newton method[33] to achieve its approximate solution, i.e., the transmission power is updated iteratively by

$$p_i(k+1) = p_i(k) - \frac{g(p_i(k)) - G(p_i(k))}{g'(p_i(k)) - G'(p_i(k))} \tag{38}$$

where $g'(p_i)$ and $G'(p_i)$ denote the first-order derivative of $g(p_i)$ and $G(p_i)$ with regard to p_i, respectively.

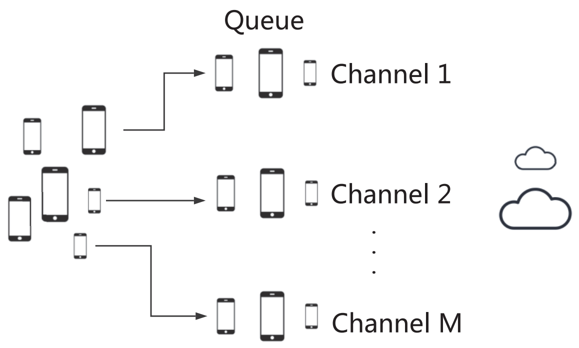

Figure 3: The queue model.

4.3. Queue Delay Optimization

As shown in Fig. 3, there are a lot of tasks with different size. We assume that there are M channels for data offloading, denoted by $\mathcal{M} = \{1, 2, ..., M\}$. The vector $\vec{\lambda} = \{\lambda^1, \lambda^2, ..., \lambda^M\}$ represents a set of service arrival rates allocated to M channels, where λ^m denotes the data offloading rate on channel m. We are dedicated to find a set of optimal data offloading rate $\vec{\lambda}^*$ to minimize the queue delay. Thus we can formulate queue delay minimization problem as

$$\min_{\lambda_i^m} \sum_{i=1}^{J} \sum_{m=1}^{M} \alpha_i T_{i,q}^m \quad m \in \{1, 2, ..., M\} \tag{39}$$

where $T_{i,q}^m$ is the queueing delay at channel m, which is given by (8). For channel m, we use a M/M/n queue model of multi-service windows with different capacity to analyze the queue delay. We can depict the state flow chart as Fig.4.

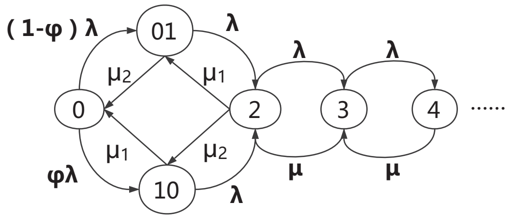

Figure 4: The state transition for each channel

The different states (i.e., the numbers in circles) in Fig.4 represent different task scenarios, i.e., number of tasks in the channel. For example, state '0' and '2' represent that there is no task and there are 2 tasks in the channel respectively. State '01' means that there is no task in the central cloud window and the edge cloud window has one task in the channel. State '10' has the opposite meaning of state '01'.

As shown in Fig.4, for facilitating the calculation, we assume that there is just one central cloud and one edge cloud. They are independent of each other and have different service rate μ_1 and μ_2. μ_1 represents central cloud's service rate and μ_2 denotes edge cloud's service rate. In fact, we can utilize the computation capability $f_{i,c}$ to reflect service rate. It is not difficult to obtain $\mu_1 > \mu_2$. We let $\mu = \mu_1 + \mu_2$, and φ denote the accessing rate of central cloud. Then the accessing rate of edge cloud is $1 - \varphi$.

We can get a set of global balance equations by using the following state balance equations: At state '0',

$$\lambda P_0 = \mu_2 P_{01} + \mu_1 P_{10} \tag{40}$$

At state '01',

$$(\lambda + \mu_2) P_{01} = \mu_1 P_2 + (1 - \varphi)\lambda P_0 \tag{41}$$

At state '10',

$$(\lambda + \mu_1) P_{10} = \mu_2 P_2 + (1 - \varphi)\lambda P_0 \tag{42}$$

At state '2',

$$(\lambda + \mu) P_2 = \mu P_3 + \lambda P_{01} + \lambda P_{10} \tag{43}$$

According to Fig. 4, we obtain

$$(\lambda + \mu) P_k = \mu P_{k+1} + \lambda P_{k-1} \qquad (k > 2) \tag{44}$$

where P_k is the probability in state k.

Theorem 2 (Queueing delay). *For a channel in a stead-state condition $\lambda - \mu < 0$, the expected queueing delay can be expressed by*

$$T_{i,q}^m = \frac{(1+\gamma)^2(\rho_i^m)^3 + q(\rho_i^m)^2}{-(1+\gamma^2)(\rho_i^m)^3 + (q-\gamma)(\rho_i^m)^2 + q(\rho_i^m) + \gamma}\frac{1}{\mu} \tag{45}$$

where $\gamma = \frac{\mu_2}{\mu_1}, \rho_i^m = \frac{\lambda_i^m}{\mu}, q = 1 - \varphi + \gamma + \gamma^2\varphi$.

Proof. Please refer to Appendix B. ■

We can observe that (45) is the form of division, therefore in this subsection, we would like to transform it into the form of subtraction by using fractional programming [34]. Let $T_{i,q}^m = \eta = \frac{a(\lambda_i^m)}{b(\lambda_i^m)}$, where $a(\lambda_i^m) = (1+\gamma)^2(\rho_i^m)^3 + q(\rho_i^m)^2$ and $b(\lambda_i^m) = \mu[-(1+\gamma^2)(\rho_i^m)^3 + (q-\gamma)(\rho_i^m)^2 + q(\rho_i^m) + \gamma]$. Therefore, the optimization problem (39) can be transformed into

$$\min_{\lambda_i^m} \sum_{i=1}^{J} \sum_{m=1}^{M} \alpha_i \eta \tag{46}$$

It is clear that (46) is a fractional programming, we formulate a new function about η

$$g(\eta) = \min_{\lambda_i^m}\{a(\lambda_i^m) - \eta b(\lambda_i^m)\} \tag{47}$$

According to [34], $\eta = \min\limits_{\lambda_i^m} \frac{a(\lambda_i^m)}{b(\lambda_i^m)}$ if and only if $g(\eta) = \min\limits_{\lambda_i^m}\{a(\lambda_i^m) - \eta b(\lambda_i^m)\} = 0$. Therefore we can find the optimal solution η^* by the Method of Bisection[35]. Once we obtain the η^*, we can get the optimal λ_i^{m*} from Eq.(47). Accordingly, we can achieve optimal queue delay.

4.4. Selection Strategy Optimization

In this subsection, according to the results of previous subsections, our designed offloading selection strategy includes two aspects of selections, i.e., first determining that the task should be offloaded onto the edge cloud or the central cloud, i.e., cloud selection, and then deciding whether this task should be offloaded or not, i.e., offloading selection, such that the total application cost is minimum while meeting user's requirements. Therefore, our first objective function with respect to selection strategy β_i can be written as $Cost_{i,c} = \beta_i Cost_{i,e} + (1-\beta_i)Cost_{i,t}$, we turn it into the following form:

$$\min_{\beta_i} Cost_{i,t} + \beta_i(Cost_{i,e} - Cost_{i,t}) \tag{48}$$

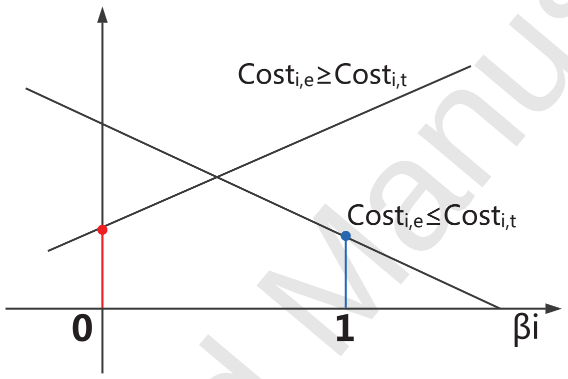

Figure 5: The optimization of β_i.

It is clear that the objective function in (48) is a linear function on β_i. As shown in Fig.5, we can easily draw the following conclusions. If $Cost_{i,e} > Cost_{i,t}$, we will get the minimum of (48) when we take $\beta_i = 0$. By contrast, if $Cost_{i,e} < Cost_{i,t}$, we will get the minimum of (48) when we take $\beta_i = 1$. Thus, we can obtain the optimal cloud selection as follows

$$\beta_i = \begin{cases} 1 & \text{if} \quad Cost_{i,e} < Cost_{i,t}, \\ 0 & \text{otherwise.} \end{cases} \tag{49}$$

similar to the computation of β_i, we can compute α_i. The objective function with respect to selection strategy α_i can be turned into the following form:

$$\min_{\alpha_i} \alpha_i Cost_{i,c} + (1-\alpha_i)Cost_{i,l}, \tag{50}$$

which can be rewritten as

$$\min_{\alpha_i} Cost_{i,l} + \alpha_i(Cost_{i,c} - Cost_{i,l}) \tag{51}$$

Clearly, if $Cost_{i,c} > Cost_{i,l}$, we will get the minimum of (51) when we take $\alpha_i = 0$. On the contrary, if $Cost_{i,c} < Cost_{i,l}$, we will get the minimum of (51) when we take $\alpha_i = 1$. That is, we can obtain the optimal offloading selection as follows

$$\alpha_i = \begin{cases} 1 & \text{if} \quad Cost_{i,c} < Cost_{i,l}, \\ 0 & \text{otherwise.} \end{cases} \tag{52}$$

The cloud selection strategy (49) indicates that when the computation cost on the edge cloud is less than that on central cloud, we can offload the task onto edge cloud. The offloading selection strategy (52) means when the computation cost on the cloud is less than that on local device, we will offload the task onto the cloud.

Note that our objective function is divided into four parts to be solved sequentially by using different methods. The proposed algorithm is described in Algorithm 1. In Algorithm 1, firstly, based on the information of computation task, mobile device and edge cloud, the local device determines whether and where the task is offloaded by calculating and comparing the energy-efficiency cost on local device, edge cloud and central cloud respectively (line 3-line 18). Then, if the task is executed locally, the energy-efficiency cost will be minimized by optimizing CPU clock frequency of mobile device (line 19-line 22). If the task is allocated onto the cloud, the energy-efficiency cost will be minimized by optimizing transmission power and queue delay (line 24-line 28). It is not difficult to find that the time complexity of the algorithm is $\mathcal{O}(J * Iter_{max} * Iter_{p_i})$, where $Iter_{max}$ denotes the maximum number of iterations, and $Iter_{p_i}$ means the number of iterations for transmission power in line 38 by Newton iteration method.

5. Performance Evaluation

In this section, we evaluate the performance for our proposed algorithm and offloading strategy with different specifications.

5.1. Simulation Setting

In our simulations, we let link bandwidth $W = 3MHz$ and the noise power be 50 dBm. We set the channel gain $h_i = D^{(-\zeta)}$, where $\zeta = 4$ is the path loss factor and D is the distance between mobile devices and cloud. We characterize the complexity of a task from two ways, i.e., the required total CPU cycles of task $Cy_{i,l}$ and the data size d_i. Since different tasks have different execution requirement, $Cy_{i,l}$ is set from 100 to 1000 Mega cycles and d_i is from 100KB to 3MB. The number of tasks varies from 10 to 40 for our simulation.

In order to measure the complexity of the offloading task, similar to [29], we utilize Load-input Data Ratio (LDR), i.e., $LDR = \frac{Cy_{i,l}}{d_i}$. If the LDR is high, then the task is complex and vice versa.

5.2. Impact of Task Complexity

In this subsection, we discuss the impact of task complexity on energy-efficiency cost. We employ LDR to reflect the complexity of tasks. Without loss of generality, we select four values of LDR from 2 to 10 randomly, i.e., let $LDR = 2.19, 4.28, 6.47, 10.00$. Fig.6 shows the changes of cost for different LDR and Fig.7 depicts the changes of the numbers of offloaded tasks for different LDR.

We can find from Fig. 6 that the cost increases rapidly with LDR and the number of tasks. Fig.7 shows that as the LDR increases, the number of tasks offloaded onto the cloud increases. According to the definition of LDR, the higher the LDR is, the more complex the task is, thus

Algorithm 1 Iterative optimization algorithm for task i.

```
Input: : the task $i$, the transmission channel $m$
    $d_i, Cy_i, \delta_{i,T}, \delta_{i,E}, \epsilon(k), f_i, p_i, \lambda_m$
    $Iter_{max}$: maximum number of iterations and iteration index $k \leftarrow 1$
    $\tau$:an infinitesimal number;
Output: : $\alpha_i^*$, $\beta_i^*$, $f_{i,l}^*$, $p_i^*$, $T_{i,q}^{m*}$: the optimal policy;
 1: for $i = 1 \quad to \quad J$ do
 2:     while $t < Iter_{max}$ and $|\theta(k+1) - \theta(k)| > \tau$ do
 3:         /*Selection strategy optimization*/
 4:         Compute $T_{i,l}, E_{i,l}, Cost_{i,l}$ by (1)-(3) respectively
 5:         Compute $r_i$ by (5)
 6:         Compute $T_{i,e}, E_{i,e}, Cost_{i,e}, T_{i,t}, E_{i,t}, Cost_{i,t},$ by (6)-(11) respectively
 7:         if $Cost_{i,e} < Cost_{i,t}$ then
 8:             $\beta_i = 1$;
 9:         else
10:             $\beta_i = 0$;
11:         end if
12:         Compute $Cost_{i,c}$ by (12)
13:         if $Cost_{i,c} < Cost_{i,l}$ then
14:             $\alpha_i = 1$;
15:         else
16:             $\alpha_i = 0$;
17:         end if
18:         if $\alpha_i == 0$ then
19:             /*Processing capability optimization*/
20:             Calculate $f_{i,l}$ by (21)-(32)
21:             Update Lagrange multiplier $\theta$ by (33)
22:             $p_i(k+1) = p_i(k)$
23:         else
24:             /*Transmission power allocation*/
25:             $f_{i,l}(k+1) = f_{i,l}(k)$
26:             Compute $p_i$ by (34)-(38) using Newton iteration method
27:             /*Queue delay optimization*/
28:             Compute $T_{i,q}^m$ using (47)
29:         end if
30:     end while
31: end for
```

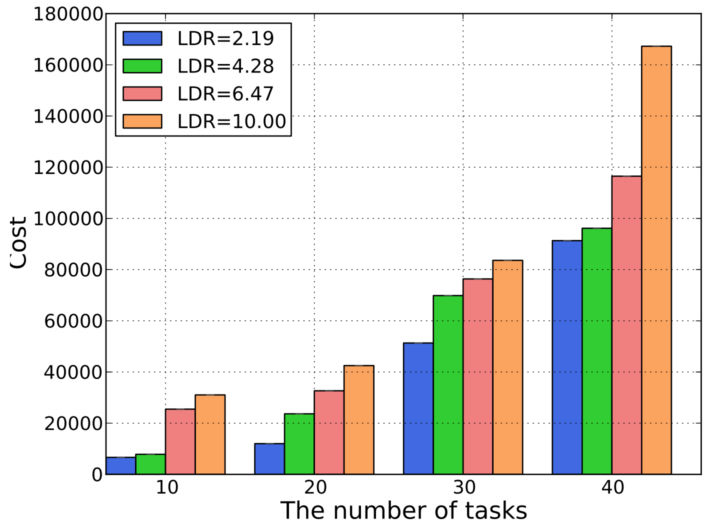

Figure 6: The impact of task complexity.

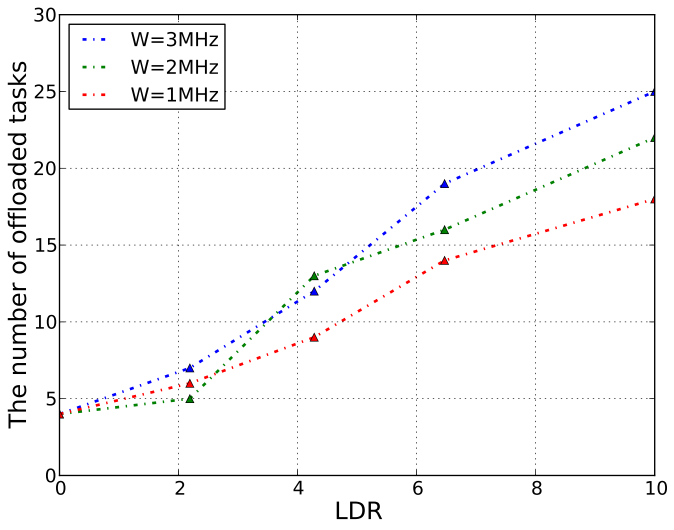

Figure 7: The impact of task complexity.

the more energy it consumes. Fig.7 verifies that if a task has high Cy_i and low d_i, then the LDR is high, which means the cost in local computing is higher than that in cloud computing. Thus, if the LDR is low, the task is more suitable for local processing. Otherwise, the task is more suitable for offloading. And Fig.7 also describes the relationship between the number of tasks offloaded onto the cloud and the channel bandwidth. Channel bandwidth indicates the condition of communication. When the LDR is fixed, the better communication conditions, the higher the number of offloaded tasks.

5.3. *Impact of Mobility*

In this subsection, we explore the impact of mobility on our algorithm by changing different channel gain h_i. The channel gain is related to the path loss factor ζ and the distance between mobile devices and cloud D. The greater the relative distance D per unit time, the smaller the channel gain when the the path loss factor is fixed. In other words, the smaller the channel gain, the faster the mobile device moves. Fig. 8 shows the evolution of transmission time for different moving speeds.

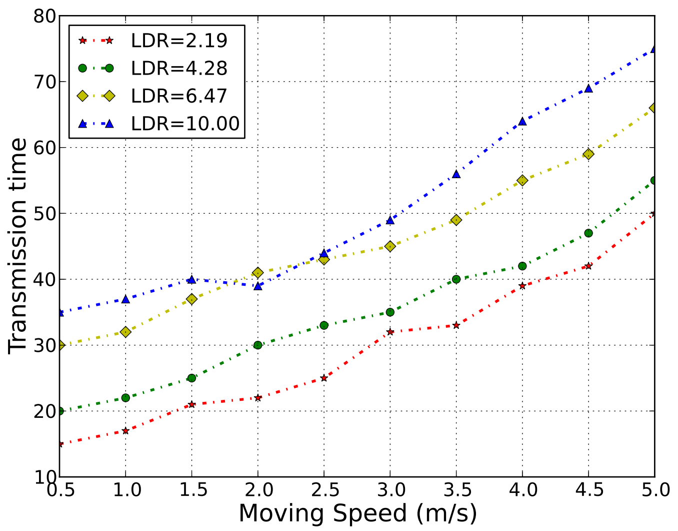

Figure 8: The impact of mobility.

It is not difficult to observe from Fig. 8 that as the relative velocity increases, the transmission time also increases for all LDRs. High-speed movement makes the communication conditions between the clients and the edge clouds become unstable. Unstable communication has resulted in a continuously decreasing data rate. Thus the transmission time increases. For different LDR, the higher LDR is, the more complex the task is. It is known that as the LDR increases, the number of tasks offloaded onto the cloud increases. This may cause some tasks to be put in the queue and thus increase the transmission time.

5.4. *Comparison of Different Strategies*

In this subsection, we compare our scheme with other schemes, i.e., all tasks are offloaded onto the central cloud (central cloud execution), all tasks are offloaded onto the edge cloud (edge cloud execution) and all tasks are executed in local device (local device execution).

Fig. 9 depicts the evolution of cost over different schemes. We can observe from Fig. 9 that the change rates of cost by our strategy in remote cloud execution for all tasks are increasing stably. The reason is that the service rate of the central cloud is stable and our scheme has a continuous regulating role for saving cost. However, in local execution for all tasks, the mobile device may be in the preheating stage such that the cost is high for 10 to 20 tasks.

On the other hand, we can also observe from Fig. 9 that as the number of tasks grows, the advantages of our strategy become more obvious. When the number of tasks reaches 30, our

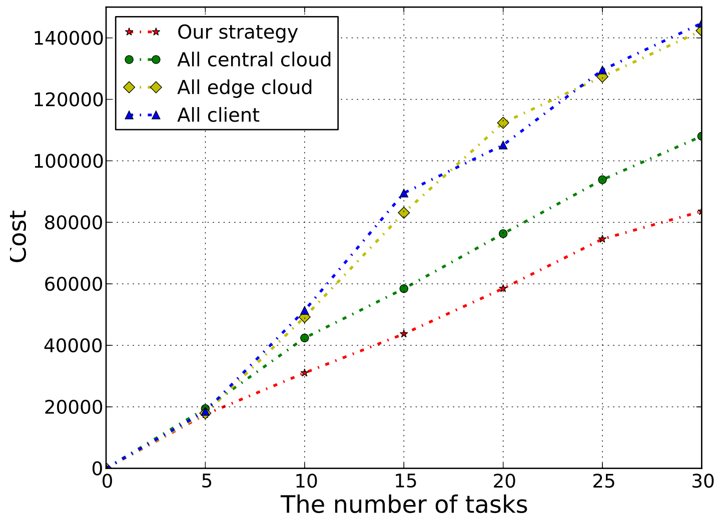

Figure 9: The comparison of different patterns. "All central cloud": All tasks are offloaded onto central cloud. "All edge cloud": All tasks are offloaded onto edge cloud. "All client": All tasks are executed on local device.

strategy can save about 40% of the cost compared to "All client". And "All central cloud" is not most energy efficient due to the channel conditions and transmission cost. In practice, offloading all tasks to the central cloud is almost impossible under the constraint of bandwidth and data rate. In addition, the edge cloud execution for all tasks brings more cost than the local device execution at some stage due to the additional offloading cost. In addition, we can find that the local device execution for all tasks has highest cost owing to the limitation of local devices in computation capability and energy.

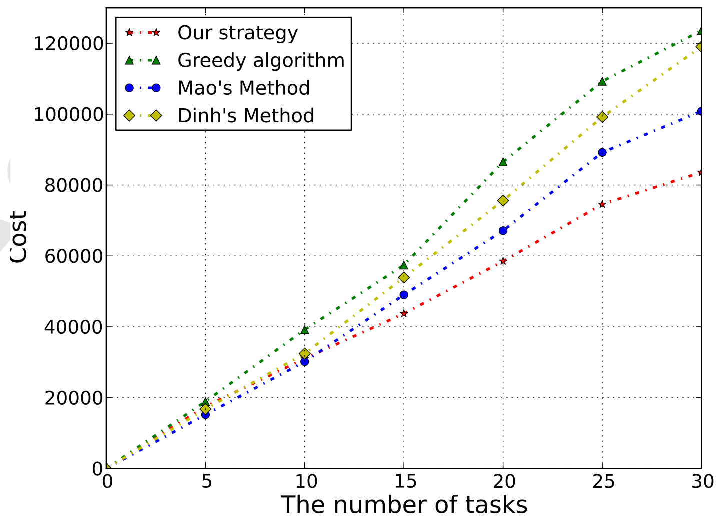

Figure 10: The comparison of different strategies.

Furthermore, we compare our strategy with other offloading schemes, i.e., Greedy algorithm

[36], Mao's Method in [10] and Dinh's Method in [13] when there are 5 to 30 tasks need to be executed. It is not difficult to observe from Fig.10 that the cost by greedy algorithm grows fastest than our scheme. Although Mao's Method and Dinh's Method are slightly better than our method when the number of tasks is small, the advantages of our strategy become more obvious with the number of tasks. This is because greedy algorithm only pursues the optimal value at the current state such that when the number of tasks is larger, its cost grows rapidly. Mao's Method in [10] ignores the central cloud in MEC system, thus lacks the supervision of the central cloud so that the tasks cannot be co-processed by the powerful central cloud in their algorithm. Dinh's Method in [13] mainly focuses on CPU frequency and data rate. However, Dinh's method lacks of scheduling of limited channel resources. Thus a large number of offloaded tasks result in channel congestion and higher communication cost. To improve performance, our designed algorithm not only considers the assistance of the central cloud but also optimizes the data delivering rate to avoid channel congestion. In numerical terms, our algorithm can save about 30% of the cost of the worst greedy algorithm and reduce about 19% compared to the Mao's method.

5.5. Trade-off Between Central Cloud and Edge Clouds

As aforementioned, the central cloud is responsible for monitoring whether our strategy works well and helps the edge cloud handle some complex tasks. However, excessive offloading requests onto the central cloud may cause the longer queues and lead to channel congestion. Although we optimize the data delivering rate to minimize the cost, the trade-off between the central cloud and edge clouds still needs to be considered.

We assume that there are 30 tasks, and also we employ the Decision Ratio (DR) to indicate the trade-off between the central cloud and edge clouds. Here, DR is the ratio of the computation capability of central cloud $f_{i,t}$ to that of edge cloud $f_{i,e}$. In this subsection, we will evaluate the impact of DR on system cost and the number of tasks offloaded onto the central cloud over different $f_{i,t}$ for a given $f_{i,e}$. The offloading decision is determined by DR due to the changes in the processing capability of the central cloud.

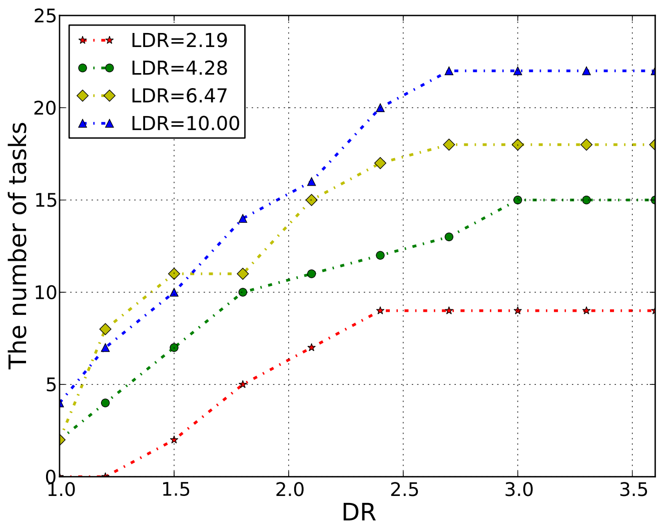

Figure 11: The numbers of offloading tasks with DR.

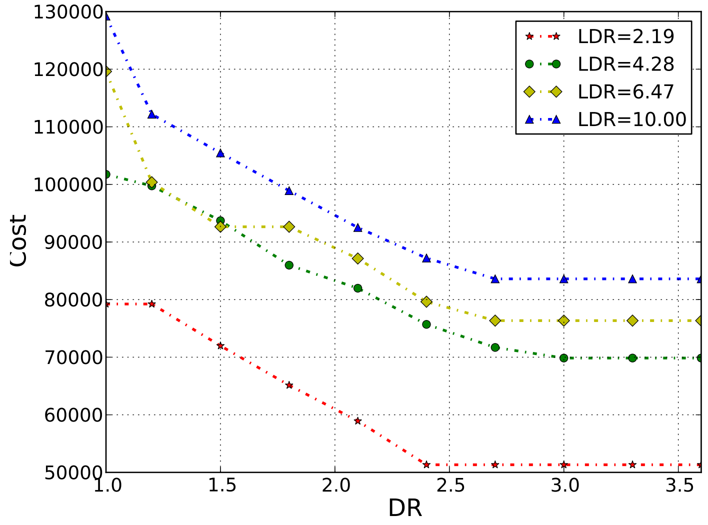

Figure 12: System cost with DR.

As shown in Fig. 11, the number of tasks offloaded onto the central cloud grows remarkably when DR is between 1.0 and 2.7. When DR reaches 2.7, the number of tasks offloaded onto the central cloud keeps unchanged. In contrast, Fig. 12 shows that the cost rapidly descends when the DR is less than 2.7 and the cost keeps stable when DR reaches 2.7. Furthermore, we can observe from Fig. 11 and Fig. 12 that due to the powerful computation ability of the center cloud, a large number of tasks are offloaded onto the central cloud so that the system cost is greatly reduced. This reflect the importance of the central cloud in the MEC system. However, when a threshold is reached, the growth of processing capability does not bring the reduction of the system cost. This is because the excessive offloading will cause congestion and much queuing after the center cloud is fully utilized. Therefore, there are always some tasks that are more beneficial to be executed at the edge cloud to reduce the offloading cost. In addition, the results in the figure also prove our previous conclusion that if the LDR is low, the task is more suitable for local processing. Otherwise, the task is more suitable for offloading.

6. Conclusions

In this paper, we propose a task offloading and resource scheduling algorithm to address the minimization problem of total energy consumption and processing time during offloading in MEC system. To the best of our knowledge, this work is the first work on dynamic task offloading and resource scheduling that minimizes total cost by taking into account both the CPU clock frequency control on local device, and the transmission power allocation and the task offloading rate on mobile edge computing with the assistance of the central cloud. We first propose an offloading selection strategy with the help of central cloud to determine where the task should be executed. Then we optimize the clock frequency for local execution and the transmission power allocation and queue delay in mobile edge computing. Our experimental results show that our task offloading and resource scheduling algorithm outperform other existing approaches in cost-reducing.

7. Acknowledgement

This work was supported by the National Natural Science Foundation of China (No. 61373179, 61373178, 61402381, 61503309, 61772432), Natural Science Key Foundation of Chongqing (cstc2015jcyjBX009 Natural Science Foundation of Chongqing (CSTC2016JCYJA0449), China Postdoctoral Science Foundation (2016M592619), Chongqing Postdoctoral Science Foundation (XM2016002), and the Fundamental Research Funds for the Central Universities (XDJK2015C010, XDJK2015D023, XDJK2016A011, XDJK2016D047, XDJK 201710635069).

A. proof theorem 1

We prove that the objective function $Cost_i$ in (14) is jointly convex with respect to (w.r.t) the optimization variables α_i, β_i, $f_{i,l}$, p_i and λ_i^m firstly.

According to the model in Section 3, the objective function can be rewritten as

$$\begin{aligned} Cost_i &= \alpha_i Cost_{i,c} + (1-\alpha_i)Cost_{i,l} \\ &= \alpha_i(\beta_i Cost_{i,e} + (1-\beta_i)Cost_{i,t}) + (1-\alpha_i)Cost_{i,l} \\ &= \alpha_i[\beta_i(\delta_{i,T}T_{i,e} + \delta_{i,E}E_{i,e}) \\ &+ (1-\beta_i)(\delta_{i,T}T_{i,t} + \delta_{i,E}E_{i,t})] \\ &+ (1-\alpha_i)(\delta_{i,T}T_{i,l} + \delta_{i,E}E_{i,l}) \end{aligned} \tag{53}$$

Since the parameters κ, the weight of time $\delta_{i,T}$, the weight of energy $\delta_{i,E}$, total CPU cycles Cy_i and the data size d_i can be regarded as constant for task i, $Cost_i$ can be transformed as

$$\begin{aligned} Cost_i &= \delta_{i,T}T_{i,l} + \delta_{i,E}E_{i,l} \\ &\quad + \alpha_i\delta_{i,T}(\beta_i T_{i,e} + T_{i,t} - \beta_i T_{i,t} - T_{i,l}) \\ &\quad + \alpha_i\delta_{i,E}(\beta_i E_{i,e} + E_{i,t} - \beta_i E_{i,t} - E_{i,l}) \end{aligned} \tag{54}$$

We can observe from (1) and (2) that $T_{i,l}$ and $E_{i,l}$ are convex w.r.t $f_{i,l}$. As shown in (2) r_i is concave w.r.t p_i and $T_{i,q}^m$ is only related with λ_i^m, thus we have $T_{i,e}$ is convex w.r.t p_i and λ_i^m, respectively. Furthermore, it is not difficult to verify that $E_{i,e}$ is also convex with p_i and λ_i^m, respectively. Similarly, we can find that $T_{i,t}$ and $E_{i,t}$ are convex with p_i and λ_i^m, respectively.

In order to unify the expression, we define a vector $\mathbf{h}_{i,m} = [\alpha_i, \beta_i, f_{i,l}, p_i, \lambda_i^m]$. Then we use $\mathbf{H}(Cost_i(h_{i,m}))$ to denote the Hessian matrix of function $Cost_i(h_{i,m})$. So we can verify that $\mathbf{H}(Cost_i(h_{i,m}))$ is a positive semi-definite matrix. In short, the objective function $Cost_i(h_{i,m})$ is jointly convex w.r.t α_i, β_i, $f_{i,l}$, p_i and λ_i^m.

Now we verify the convexity of the constraints. We can verify that the left terms of inequalities (15) and (16) are convex, and the right terms of them are constants, which mean that the constraints (15) and (16) are convex w.r.t α_i, β_i, $f_{i,l}$, p_i and λ_i^m.

As a result, the optimization problem (14) is a convex optimization problem w.r.t the optimization variables α_i, β_i, $f_{i,l}$, p_i and λ_i^m.

B. proof theorem 2

We can use the global balance equations (40)-(44) to obtain the following equations:

$$P_{01} = \frac{\rho}{2\rho+1}\frac{1+\gamma}{\gamma}(\rho+1-\varphi)P_0 \tag{55}$$

$$P_{10} = \frac{\rho}{2\rho+1}(1+\gamma)(\rho+\varphi)P_0 \tag{56}$$

$$\mu P_2 = \lambda P_{01} + \lambda P_{10} \tag{57}$$

where $\gamma = \frac{\mu_2}{\mu_1}, \rho = \frac{\lambda}{\mu}$. Then we combine the equation (56) with (57) and can get

$$P_3 = \rho P_2 \tag{58}$$

Using equation(56), (57) and (58), we calculate P_2 as

$$P_2 = \frac{\rho^2}{2\rho+1}\frac{1+\gamma}{\gamma}[(1+\gamma)\rho + (\gamma-1)\varphi + 1] \tag{59}$$

And then P_3 could be written as

$$P_3 = \frac{\rho^3}{2\rho+1}\frac{1+\gamma}{\gamma}[(1+\gamma)\rho + (\gamma-1)\varphi + 1] \tag{60}$$

Therefore, we can conclude that

$$P_n = \frac{\rho^n}{2\rho+1}\frac{1+\gamma}{\gamma}[(1+\gamma)\rho + (\gamma-1)\varphi + 1] \tag{61}$$

According to the regularity condition $P_0 + P_{01} + P_{10} + \sum_{n=2}^{+\infty} P_n = 1$, we compute P_0 by

$$P_0 = \frac{1-\rho}{1+\rho[1+(1+\gamma^2)\rho - (1-\gamma^2)\varphi]/\gamma(1+2\rho)} \tag{62}$$

The queue length L_q^m is expressed as

$$L_q^m = \sum_{k=3}^{+\infty}(k-2)p_k = \frac{(1+\gamma)^2\rho^4 + q\rho^3}{-(1+\gamma^2)\rho^3 + (q-\gamma)\rho^2 + q\rho + \gamma} \tag{63}$$

Thus the queue time is

$$T_{i,q}^m = \frac{(1+\gamma)^2(\rho_i^m)^3 + q(\rho_i^m)^2}{-(1+\gamma^2)(\rho_i^m)^3 + (q-\gamma)(\rho_i^m)^2 + q(\rho_i^m) + \gamma}\frac{1}{\mu} \tag{64}$$

where $\gamma = \frac{\mu_2}{\mu_1}, \rho_i^m = \frac{\lambda_i^m}{\mu}, q = 1 - \varphi + \gamma + \gamma^2\varphi$.

[1] P. Corcoran, S. K. Datta, Mobile-edge computing and the internet of things for consumers: Extending cloud computing and services to the edge of the network, IEEE Consumer Electronics Magazine 5 (4) (2016) 73–74.

[2] Y. Jararweh, A. Doulat, O. Alqudah, E. Ahmed, M. Al-Ayyoub, E. Benkhelifa, The future of mobile cloud computing: Integrating cloudlets and mobile edge computing, in: International Conference on Telecommunications, 2016.

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

[4] P. Mach, Z. Becvar, Mobile edge computing: A survey on architecture and computation offloading, IEEE Communications Surveys and Tutorials PP (99) (2017) 1–1.

[5] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, D. Sabella, On multi-access edge computing: A survey of the emerging 5g network edge architecture and orchestration, IEEE Communications Surveys and Tutorials PP (99) (2017) 1–1.

[6] A. U. R. Khan, M. Othman, S. A. Madani, S. U. Khan, A survey of mobile cloud computing application models, IEEE Communications Surveys and Tutorials 16 (1) (2014) 393–413.

[7] H. T. Dinh, C. Lee, D. Niyato, P. Wang, A survey of mobile cloud computing: architecture, applications, and approaches, Wireless Communications and Mobile Computing 13 (18) (2013) 15871611.

[8] Z. Kuang, S. Guo, J. Liu, Y. Yang, A quick-response framework for multi-user computation offloading in mobile cloud computing, Future Generation Computer Systems 81 (2018) 166 – 176.

[9] P. Zhao, H. Tian, C. Qin, G. Nie, Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing, IEEE Access PP (99) (2017) 1–1.

[10] Y. Mao, J. Zhang, S. H. Song, K. B. Letaief, Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems, IEEE Transactions on Wireless Communications PP (99) (2017) 1–1.

[11] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, Y. Zhang, Energy-efficient offloading for mobile edge computing in 5g heterogeneous networks, IEEE Access 4 (99) (2016) 5896–5907.

[12] C. Wang, F. R. Yu, C. Liang, Q. Chen, L. Tang, Joint computation offloading and interference management in wireless cellular networks with mobile edge computing, IEEE Transactions on Vehicular Technology PP (99) (2017) 1–1.

[13] T. Q. Dinh, J. Tang, Q. D. La, T. Q. S. Quek, Offloading in mobile edge computing: Task allocation and computational frequency scaling, IEEE Transactions on Communications PP (99) (2017) 1–1.

[14] B. Han, P. Hui, V. S. A. Kumar, M. V. Marathe, J. Shao, A. Srinivasan, Mobile data offloading through opportunistic communications and social participation, IEEE Transactions on Mobile Computing 11 (5) (2012) 821–834.

[15] Y. Yu, J. Zhang, K. B. Letaief, Joint subcarrier and cpu time allocation for mobile edge computing, in: 2016 IEEE Global Communications Conference (GLOBECOM), 2016, pp. 1–6. doi:10.1109/GLOCOM.2016.7841937.

[16] S. Sardellitti, G. Scutari, S. Barbarossa, Joint optimization of radio and computational resources for multicell mobile-edge computing, IEEE Transactions on Signal and Information Processing Over Networks 1 (2) (2014) 89–103.

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

[18] M. H. U. Rehman, C. Sun, T. Y. Wah, A. Iqbal, P. P. Jayaraman, Opportunistic computation offloading in mobile edge cloud computing environments, in: IEEE International Conference on Mobile Data Management, 2016, pp. 208–213.

[19] C. You, K. Huang, H. Chae, B. H. Kim, Energy-efficient resource allocation for mobile-edge computation offloading, IEEE Transactions on Wireless Communications 16 (3) (2017) 1397–1411. doi:10.1109/TWC.2016.2633522.

[20] U. Drolia, R. Martins, J. Tan, A. Chheda, M. Sanghavi, R. Gandhi, P. Narasimhan, The case for mobile edge-clouds, in: 2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing, 2013, pp. 209–215. doi:10.1109/UIC-ATC.2013.94.

[21] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, K. K. Leung, Dynamic service migration in mobile edge-clouds, in: 2015 IFIP Networking Conference (IFIP Networking), 2015, pp. 1–9. doi:10.1109/IFIPNetworking.2015.7145316.

[22] Y. Wang, M. Sheng, X. Wang, L. Wang, J. Li, Mobile-edge computing: Partial computation offloading using dynamic voltage scaling, IEEE Transactions on Communications 64 (10) (2016) 4268–4282. doi:10.1109/TCOMM.2016.2599530.

[23] D. Chatzopoulos, M. Ahmadi, S. Kosta, P. Hui, Flopcoin: A cryptocurrency for computation offloading, IEEE Transactions on Mobile Computing PP (99) (2017) 1–1.

[24] V. Cozzolino, A. Y. Ding, J. Ott, Fades: Fine-grained edge offloading with unikernels, in: Proceedings of the Workshop on Hot Topics in Container Networking and Networked Systems, HotConNet '17, ACM, New York, NY, USA, 2017, pp. 36–41. doi:10.1145/3094405.3094412.
URL `http://doi.acm.org/10.1145/3094405.3094412`

[25] L. Yang, J. Cao, S. Tang, T. Li, A. T. S. Chan, A framework for partitioning and execution of data stream applications in mobile cloud computing, Acm Sigmetrics Performance Evaluation Review 40 (4) (2013) 23–32.

[26] G. Bianchi, L. Fratta, M. Oliveri, Performance evaluation and enhancement of the csma/ca mac protocol for 802.11 wireless lans, in: Personal, Indoor and Mobile Radio Communications, 1996. PIMRC'96., Seventh IEEE International Symposium on, Vol. 2, 1996, pp. 392–396 vol.2. doi:10.1109/PIMRC.1996.567423.

[27] X. Lin, Y. Wang, Q. Xie, M. Pedram, Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment, IEEE Transactions on Services Computing 8 (2) (2015) 175–186. doi:10.1109/TSC.2014.2381227.

[28] A. P. Miettinen, 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] S. Guo, B. Xiao, Y. Yang, Y. Yang, Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing, in: IEEE INFOCOM 2016 - the IEEE International Conference on Computer Communications, 2016, pp. 1–9.

[30] L. Liu, R. Zhang, K. C. Chua, Wireless information transfer with opportunistic energy harvesting, IEEE Transactions on Wireless Communications 12 (1) (2013) 288–300.

[31] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge Univ. Press, 2013.

[32] S. Guo, G. Feng, X. Liao, Q. Liu, Hopf bifurcation control in a congestion control model via dynamic delayed feedback, Chaos 18 (4) (2008) 043104.

[33] P. Deuflhard, Newton methods for nonlinear problems, Springer 28 (6) (2011) 12991316.

[34] W. Dinkelbach, On nonlinear fractional programming, Management Science 13 (7) (1967) 492–498.
URL `http://www.jstor.org/stable/2627691`

[35] G. R. Wood, The bisection method in higher dimensions, Mathematical Programming 55 (1-3) (1992) 319–337.

[36] M.-R. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, R. Govindan, Odessa: Enabling interactive perception applications on mobile devices, in: Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services, MobiSys '11, ACM, New York, NY, USA, 2011, pp. 43–56. doi:10.1145/1999995.2000000.
URL `http://doi.acm.org/10.1145/1999995.2000000`

Highlights

We propose a task offloading and resource scheduling mechanism with three offloading destinations.
We propose a distributed algorithm containing strategy selection, clock frequency control and transmission power allocation
We give the optimal task offloading rate and queue delay by using convex optimization method.