Energy-Efficient Dynamic Computation Offloading and Cooperative Task Scheduling in Mobile Cloud Computing

Songtao Guo, *Member, IEEE*, Jiadi Liu, Yuanyuan Yang, *Fellow, IEEE*, Bin Xiao, *Senior Member, IEEE*, and Zhetao Li, *Member, IEEE*

Abstract—Mobile cloud computing (MCC) as an emerging and prospective computing paradigm, can significantly enhance computation capability and save energy for smart mobile devices (SMDs) by offloading computation-intensive tasks from resource-constrained SMDs onto resource-rich cloud. However, how to achieve energy-efficient computation offloading under hard constraint for application completion time remains a challenge. To address such a challenge, in this paper, we provide an energy-efficient dynamic offloading and resource scheduling (eDors) policy to reduce energy consumption and shorten application completion time. We first formulate the eDors problem into an energy-efficiency cost (EEC) minimization problem while satisfying task-dependency requirement and completion time deadline constraint. We then propose a distributed eDors algorithm consisting of three subalgorithms of computation offloading selection, clock frequency control, and transmission power allocation. Next, we show that computation offloading selection depends on not only the computing workload of a task, but also the maximum completion time of its immediate predecessors and the clock frequency and transmission power of the mobile device. Finally, we provide experimental results in a real testbed and demonstrate that the eDors algorithm can effectively reduce EEC by optimally adjusting CPU clock frequency of SMDs in local computing, and adapting the transmission power for wireless channel conditions in cloud computing.

Index Terms—Mobile cloud computing, energy-efficiency cost, computation offloading, resource allocation

1 Introduction

Recently, smart mobile devices (SMDs), e.g., smartphones and tablet-PCs, are gaining enormous popularity due to their portability and compactness. It is expected that SMDs will be taken as the dominant future computing devices for supporting many computation-intensive applications, such as interactive gaming, image/video processing, and online social network services [1], [2]. Such complex applications necessitate higher computing power, memory and battery lifetime on SMDs. However, due to physical size constraint, mobile devices are generally resource-constrained. In particular, the limited energy supply from battery has been one of the most challenging issues for SMDs [3], [4], [5].

On the other hand, with the development of wireless communication technology, such as Wi-Fi, 4G or even 5G, mobile cloud computing (MCC) is envisioned as a promising approach to addressing such a challenge. The objective of MCC is to extend the powerful computing capability of resource-rich clouds to resources constrained SMDs so as to augment the computing potentials of SMDs. However, it remains difficult to develop a reliable MCC system. A key challenge is how to achieve an energy-efficient computation offloading. In order to realize the prospective benefits of MCC in energy saving and performance improvement for mobile devices, the following problems should be addressed: (i) Which tasks of an application should be offloaded onto the cloud? (ii) How much CPU clock frequency should be assigned for each task in local computing? (iii) How much transmission power should be employed for offloading tasks in cloud computing?

In general, these three problems need to be solved optimally to achieve maximum energy saving and minimum application completion time, because the optimality of offloading decision is affected by local clock frequency and transmission power. However, the optimization does not mean that local clock frequency and transmission power reach optimum at the same time since a task will only be executed in either a local device or remote cloud. Based on the above considerations, in this paper we focus on the computation offloading and resource scheduling problem, in which the following three key issues will be addressed.

- How computation offloading is selected when both task-dependency requirement and application completion time constraint are enforced? The enforcement is necessary since there exist certain precedences among tasks and the application completion time is a hard constraint for latency-sensitive applications.

- *S. Guo and J. Liu are with the Chongqing Key Lab of Nonlinear Circuits and Intelligent Information Processing, College of Electronic and Information Engineering, Southwest University, Chongqing 400715, P. R. China. E-mail: stguo@swu.edu.cn, jandyrei@foxmail.com.*
- *Y. Yang is with the Department of Electrical & Computer Engineering, Stony Brook University, Stony Brook, NY 11794. E-mail: yuanyuan.yang@stonybrook.edu.*
- *B. Xiao is with the Department of Computing, The Hong Kong Polytechnic University, Kowloon, Hong Kong. E-mail: csbxiao@comp.polyu.edu.hk.*
- *Z. Li is with the College of Information Engineering, Xiangtan University, Hunan411105, P. R. China. E-mail: liztchina@hotmail.com.*

Manuscript received 31 Oct. 2016; revised 16 Sept. 2017; accepted 17 Apr. 2018. Date of publication 30 Apr. 2018; date of current version 7 Jan. 2019. (Corresponding author: Yuanyuan Yang.)
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TMC.2018.2831230

- Can energy-efficiency cost (EEC), which is defined in *Definition 2* in Section 3.3.1, be minimized by optimally controlling the CPU clock frequency of the mobile device by dynamic voltage and frequency scaling (DVFS) technique [6] in local execution?
- Can EEC be minimized by optimally allocating data transmission power for each computation offloading while satisfying task-precedence requirement in cloud execution?

The objective of this paper is to provide an optimal energy-efficient dynamic offloading and resource scheduling (eDors) policy, by minimizing the EEC paid by the mobile device for completing an application. In our eDors policy, the optimal offloading decision to be made is based on the comparison between the local execution cost at optimal clock frequency and the cloud execution cost at optimal transmission power. In particular, the optimal clock frequency and transmission power are calculated independently in local computing and cloud computing, respectively.

Compared to the previous work [6], [7], [8], [9], [10], our work makes several contributions. *First*, by considering the impact of task dependencies on computation completion time, we characterize EEC by energy consumption and computation completion time in local computing and cloud computing, respectively. In particular, to reduce energy consumption, we regard the task offloading rate as the elastic one by adapting transmission power for wireless channel state and completion time deadline. *Second*, we formulate the eDors problem into an EEC minimization problem under the constraints of application completion time and task-precedence requirement. In the formulation, the maximum completion time (i.e., completion time deadline) requirement of an application is enforced to satisfy different types of applications such as delay-sensitive applications and delay-tolerate applications. *Third*, we propose a distributed eDors algorithm to achieve optimal computation offloading selection, clock frequency control and transmission power allocation. More importantly, we show that the computation offloading selection decision depends on not only the computing workload of a task, but also the maximum completion time of its immediate predecessors as well as the clock frequency and transmission power of the mobile device.

We implement the proposed eDors algorithm on an Android based real testbed consisting of 5 smartphones and a cloud sever. Our experimental results show that eDors algorithm is practical in the real world scenario, and compared to existing scheduling policies [6], [7], [9], [11], [12], [13], the eDors algorithm can effectively reduce energy consumption and application completion time. To the best of our knowledge, this is the first work that minimizes energy consumption and application completion time under completion time deadline constraint and task-precedence requirement.

The rest of this paper is organized as follows. In Section 2, we review some related work. In Section 3, we present a model for EEC in mobile execution and cloud execution, respectively. In Sections 4 and 5, we formulate the eDors problem and propose the distributed eDors algorithm, respectively. In Section 6, we implement and evaluate the eDors algorithm on a real testbed. Finally, we conclude the paper in Section 8.

2 Related Work

There have been various offloading policies proposed in the literature. These policies can be classified into two categories: (i) performance based offloading policies [8], [11], [14], [15], [16], [17], and (ii) energy based offloading policies [6], [9], [10], [18], [19], [20], [21].

2.1 Performance Based Offloading

The objective of performance based offloading policies is to enhance the performance of mobile devices in terms of execution/completion time and throughput by utilizing cloud resources. Therefore, resource-intensive computations are offloaded to the cloud. In [8], Yang et al. optimized the overall execution time by dynamically offloading part of Android code running on a smartphone to the cloud. In [14], Satyanarayanan et al. proposed a model that uses a concept of virtual machine that runs on a trusted and resource-rich computer, or a cluster of computers named cloudlet. Zhang et al. [15] partitioned a single elastic application into multiple components called weblets. Yang et al. [16], [17] studied the multi-user computation partitioning so as to optimize the partition of a data stream application such that the application has maximum throughput. In [11], to maximize the benefit of mobile cloud service providers, Kaewpuang et al. proposed a framework for resource allocation to mobile applications, and revenue management and cooperation formation among service providers. However, the above works neither focus on the energy efficiency minimization problem, nor consider the impact of task dependency and application completion time on computation offloading policy.

2.2 Energy Based Offloading

Energy based offloading policies aim to reduce energy consumption of mobile devices. As a result, computation-intensive tasks need to be performed in the cloud. In [18], [19], the authors proposed a straightforward offloading decision strategy to minimize energy consumption according to the computation-to-communication ratio and the networking environment. However, their solution does not guarantee that the execution time requirement is satisfied. In [21], Huang et al. considered the application completion deadline and presented a dynamic offloading algorithm to save energy on mobile devices. To achieve joint optimization of offloading decision and clock frequency control, in [9], Zhang et al. proposed an energy-optimal mobile cloud computing framework under stochastic wireless channel. Furthermore, in [10], they proposed a collaborative task execution framework for mobile tasks that minimizes the energy consumption on mobile devices while meeting a probabilistic completion deadline. In order to minimize energy consumption, in [12], Barbarossa et al. proposed a joint dynamic allocation of transmission power and offload scheduling that guarantees the stability of the queue of instructions to be executed. However, the above work did not consider the impact of task dependency on offloading policy.

Furthermore, by considering component dependency, in [13], Mahmoodi et al. proposed joint component scheduling and computation offloading (JSCO) for multicomponent applications. By dynamically controlling voltage and frequency in a mobile device, Lin et al. [6] proposed a task scheduling algorithm to minimize the total energy consumption of an application. However, these existing works did not consider the optimal allocation of transmission and minimization of application completion time in the task offloading decision. Compared to these works, in this paper, by considering task dependency, we propose a computation offloading policy based on optimal clock frequency and

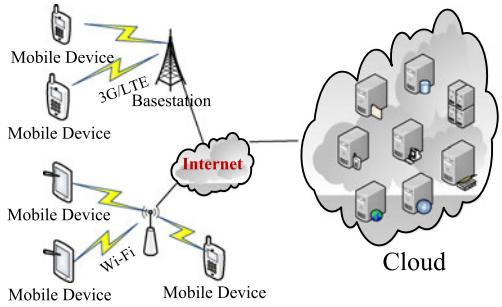


Fig. 1. Illustration of mobile cloud computing with multiple mobile devices.

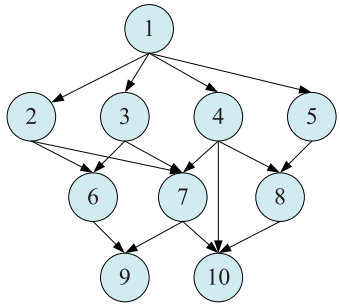


Fig. 2. A simple example of relationships among tasks.

transmission power to minimize both energy consumption and application completion time. Although Chen in [7] studied the joint minimization of processing time and energy consumption and formulated the decentralized computation offloading problem among SMDs as a decentralized game, Chen did not consider the task precedence and resource scheduling in computation offloading policy.

3 System and Computation Model

This section outlines the system model of MCC and formulates EECs in local computing and cloud computing.

3.1 System Model

We assume that there are N smart mobile devices located in a region, denoted by a set of $\mathcal{N} = \{1, 2, \ldots, N\}$, each of which has a computationally intensive mobile application to be completed. A mobile application in MCC is dynamically partitioned into a sequence of M tasks, denoted by a set of $\mathcal{M} = \{1, 2, \ldots, M\}$ that can be executed on the mobile device called *local computing* or on the cloud named *cloud computing*. SMDs offload the intensive computation to the cloud in two ways, i.e., through a mobile network (telecom network) or through access points as shown in Fig. 1. In the mobile network case, the mobile devices such as cellular smartphones are connected to a mobile network through a Base Station (BS) via 3G or LTE. In the access point case, SMDs connect to the access points through Wi-Fi. In our scheme, we employ the dynamic partitioning scheme in [17] to achieve the partition of a mobile application. The reason is that this scheme considers the partitioning of multiple users' computations together with the scheduling of offloaded computations on the cloud resources. Certainly, other partitioning schemes can work with our scheme as well.

In practice, for large scale mobile cloud applications with multiple users, due to the competition for cloud resources among users, the offloaded computations may be executed with certain scheduling delay on the cloud. Therefore, we introduce the completion time and ready time in local or cloud execution to reflect the scheduling delay on the cloud and the competition for cloud resources among multiple users. In particular, we employ the dynamic task scheduling strategy to achieve computation offloading rather than a predetermined compiler to generate scheduling order for running the application. The dynamic decision of task scheduling strategy for a task depends on task dependency, computing workload, input data size, circuit energy consumption, channel condition, etc. The study on cloud resource scheduling is out of the scope of our work.

We utilize a directed acyclic graph $G = (V, E)$ to describe the dependency relationships among these tasks, which are categorized into four types in [13]: sequential dependency, parallel dependency, random Layer-by-Layer dependency and random Fan-in/Fan-out dependency. Clearly, the parallel and sequential dependency graphs show the lowest and highest dependencies between tasks, respectively. Each node $i \in V$ in G represents a task and a directed edge $e(i, j)$ indicates the precedence constraint between tasks i and j such that task j cannot start execution until its precedent task i completes. Each task can be executed either at the mobile side or at the cloud side. Fig. 2 gives an example of the random Layer-by-Layer dependencies among 10 tasks, where the immediate predecessors of task 6 have tasks 2 and 3 and its successor/descendant is task 9.

3.2 Communication Model

In this subsection, we introduce the communication model for wireless access.

We assume that all wireless links are symmetrical, i.e., the reciprocal channel from SMD n to wireless access point s is the same as the channel from s to n. The channel from SMD n to access point s follows quasi-static block fading. In other words, the channel state remains unchanged during the offloading period of a computational intensive task of the mobile application. We let $a_{m,n} \in \{0, 1\}$ denote the computation offloading decision of task m of mobile device n. Specifically, $a_{m,n} = 1$ means that SMD n chooses to offload the computation of task m to the cloud via wireless access while $a_{m,n} = 0$ implies that SMD n decides to execute task m locally on its own device. Furthermore, we denote the decision profile of all tasks of all mobile devices by $\mathcal{A} = \{a_{m,n} | m \in \mathcal{M}, n \in \mathcal{N}\}$.

We let $P^T_{m,n}$ denote the transmission power of SMD n offloading task m to wireless access point s and $H_{m,n}$ indicate the channel power gain from SMD n to access point s when transmitting task m due to path loss and shadowing attenuation. The receiver noise at each SMD is modeled as a circularly symmetric complex Gaussian (CSCG) random variable with zero mean and variance σ^2. For a given decision profile $\mathcal{A}$, the achievable rate in bps/Hz for computation offloading of task m of mobile device n can be given by

$$R_{m,n} = W \log_2 \left(1 + \frac{P^T_{m,n} H_{m,n}}{\Gamma \sigma^2}\right), \tag{1}$$

where W is channel bandwidth and $\Gamma \geq 1$ denotes the coding gap, which is a function of bit-error-rate (BER), i.e., $\Gamma = \log(\phi_1 BER)/(-\phi_2)$, where ϕ_1 and ϕ_2 are constants depending on the actual modulation [22]. Γ models the difference between capacity-achieving codes and the actual code applied [23]. The BER is determined by the coding schemes and the medium access control (MAC) protocol. We can observe from Eq. (1) that the achievable rate is a

more realistic wireless transmission rate model that incorporates the actual modulation, coding, and MAC protocols.

3.3 Computation Model

In this subsection, we introduce the computation model in local and cloud computing.

We let $D_{m,n}$ denote the size of input data (e.g., the program codes and input parameters) related to task m of SMD n. $L_{m,n}$ denotes computing workload, i.e., the total number of CPU cycles, required for accomplishing task m of SMD n. Let $FT^l_{k,n}, FT^t_{k,n}, FT^c_{k,n}$ and $FT^r_{k,n}$ denote the completion time of local execution, the wireless task transmission from SMD n to cloud (i.e., the task has been completely offloaded to cloud), the cloud execution and the wireless reception of the computation result of task k from cloud to SMD n, respectively. Next, we discuss the energy consumption and application completion time for local and cloud computing.

3.3.1 Local Computing

Let $f_{m,n}$ denote the computation capability (i.e., the clock frequency of the CPU chip) of SMD n on task m. Here we allow different mobile devices to have different computation capability, and different tasks to be executed at different clock frequency for an SMD. The computation execution time of task m of SMD n by local computing is then given by

$$T^l_{m,n} = L_{m,n} f^{-1}_{m,n}, \tag{2}$$

and the corresponding energy consumption is given by

$$E^l_{m,n} = \kappa L_{m,n} f^2_{m,n} + E^{circ}_{m,n}, \tag{3}$$

where $E^{circ}_{m,n}$ is the constant circuit energy consumption in SMD n independent of $f_{m,n}$ for executing task m which includes the power dissipation in the digital-to-analog (/analog-to-digital) converter, digital/analog filters, mixer, and frequency synthesizer. κ is the effective switched capacitance depending on the chip architecture. We set $\kappa = 10^{-11}$ so that energy consumption is consistent with the measurements in [24]. It is clear that we can adjust the clock frequency of CPU chip to achieve optimal computation time and energy consumption on a mobile device by employing DVFS technique. The DVFS technique enables the CPU core to operate at multiple different frequency levels.

Now, we consider the impact of dependencies among tasks on the proposed scheduling algorithm, which is reflected by computation completion time of a task. The dependencies among tasks mean that before a task begins to be executed, all its immediate predecessors must have already been executed. Clearly, more complicated dependencies among tasks will increase the complexity of task scheduling. Next, we give the definition of *ready time* of a task [6].

Definition 1 (Ready Time). *The ready time of a task is defined as the earliest time when all immediate predecessors of the task have completed execution. Thus the ready time of task m of SMD n in local computing, denoted by $RT^l_{m,n}$ is given by*

$$RT^l_{m,n} = \max_{k \in \mathbf{pred}(\mathbf{m})} \max\{FT^l_{k,n}, FT^r_{k,n}\}, \tag{4}$$

where $\mathbf{pred}(\mathbf{m})$ denotes the set of immediate predecessors of task m.

We can observe from Eq. (4) that if an immediate precedent task k of task m is executed locally, then $FT^r_{k,n} = 0$, and thus $\max\{FT^l_{k,n}, FT^r_{k,n}\} = FT^l_{k,n}$. In this case, we have

$$RT^l_{m,n} \geq FT^l_{k,n}, \tag{5}$$

Otherwise, if task $k \in \mathbf{pred}(\mathbf{m})$ is offloaded onto cloud, $\max\{FT^l_{k,n}, FT^r_{k,n}\} = FT^r_{k,n}$. In this case, we have

$$RT^l_{m,n} \geq FT^r_{k,n}, \tag{6}$$

which means that task m can begin to be executed on SMD n after the mobile device has completely received the output data (results) of task k through wireless receiving channel.

By combining Eqs. (5) with (6), we have

$$RT^l_{m,n} \geq (1 - a_{k,n}) FT^l_{k,n} + a_{k,n} FT^r_{k,n}, k \in \mathbf{pred}(\mathbf{m}). \tag{7}$$

Similar to the existing work [7], [21], we ignore the time and energy consumption that the cloud returns the computation outcome back to the mobile device, due to the fact that for many applications (e.g., image processing), the size of the outcome in general is much smaller than that of input data. Accordingly, Eq. (4) can be rewritten as

$$RT^l_{m,n} \geq (1 - a_{k,n}) FT^l_{k,n} + a_{k,n} FT^c_{k,n}, k \in \mathbf{pred}(\mathbf{m}), \tag{8}$$

which implies that if not considering the time of receiving the outcome of task k, task m can start execution only after task k has completed execution.

Clearly, the completion time of local execution of task m on SMD n is the sum of the local computation execution time and the ready time in local computing, i.e.,

$$FT^l_{m,n} = T^l_{m,n} + RT^l_{m,n}, \tag{9}$$

According to Eqs. (2) and (3), we can then compute the energy-efficiency cost by the following definition.

Definition 2 (Energy-Efficiency Cost (EEC)). *Energy-Efficiency Cost is defined as the weighted sum of energy consumption and computation completion time of executing a task. Thus the EEC of task m of SMD n in local computing is given by*

$$Z^l_{m,n} = \gamma^E_{m,n} E^l_{m,n} + \gamma^T_{m,n} FT^l_{m,n}, \tag{10}$$

where $0 \leq \gamma^E_{m,n} \leq 1$ and $0 \leq \gamma^T_{m,n} \leq 1$ denote the weights of energy consumption and computation completion time for SMD n making decision on task m, respectively.

It is worth noting that the reason that the ECC is defined as a linear combination of energy consumption and computation completion time, is that both metrics can coincidentally reflect the energy-efficiency cost of executing a task, i.e., both higher energy consumption and longer computation completion time lead to higher energy-efficiency cost. To meet user-specific demands, we allow different SMDs to choose different weights in decision making. For example, a device with low battery energy would like to choose a larger $\gamma^E_{m,n}$ to save more energy. When a mobile device is running some delay-sensitive applications (e.g., online movies), it may prefer to set a larger $\gamma^T_{m,n}$ to reduce the delay.

3.3.2 Cloud Computing

For cloud computing, an SMD n will offload its computation task m to a cloud. Then the cloud will execute the task m and return the results to the SMD n. Clearly, the execution of task m in the cloud includes three phases in sequence: (i) transmitting phase, (ii) cloud computing phase, and (iii) receiving phase. In the transmitting phase, the mobile device transmits

the specification and input data of task m to the cloud through wireless uplink channel. In the cloud computing phase, task m is executed in the cloud. In the receiving phase, the mobile device receives the output of task m from the cloud through wireless downlink channel.

According to the communication model in Section 3.2, we can compute the transmission time and energy consumption of SMD n offloading task m, respectively, by

$$T^{c,trs}_{m,n}(\mathcal{A}) = D_{m,n}/R_{m,n}(\mathcal{A}), \tag{11}$$

and

$$E^{c,trs}_{m,n}(\mathcal{A}) = E^{cont}_{m,n} + P^{T}_{m,n}T^{c,trs}_{m,n}(\mathcal{A}) + E^{tail}_{m,n}, \tag{12}$$

where $E^{cont}_{m,n}$ denotes the energy consumption that SMD n contends access channel for uploading task m, which is determined by channel contention time of MAC protocol and the admission control of the cloud. The second term indicates the energy consumption for data transmission. $E^{tail}_{m,n}$ is the tail energy that SMD n will continue to hold the channel for a while even after the data transmission is done. Such a tail phenomenon is commonly observed in 3G/4G networks [25].

Furthermore, we can obtain the computation execution time of task m of SMD n on the cloud by

$$T^{c,exe}_{m,n} = L_{m,n}f_c^{-1}, \tag{13}$$

where f_c indicates the clock frequency of the processing unit on the cloud. We assume that f_c is fixed and does not change during the computation. A mobile device can know the exact clock frequency of cloud by request-to-send (RTS)/clear-to-send (CTS) message exchange when the mobile device makes connection with the cloud. To elaborate, at the beginning of the connection, a mobile device first sends a RTS message to the cloud to contend for the shared wireless channel. The RTS message includes the device ID, the modulation and coding type, the MAC protocol, etc. Once receiving the RTS message, the cloud replies a CTS message with its clock frequency and channel gain to the mobile device, which shows that they are connected successfully and are ready to communicate with each other. After the RTS/CTS exchange, therefore, the mobile device knows the information of clock frequency of the cloud as well as channel gain between them. Clearly, this way to obtain the information of cloud clock frequency is feasible and practical and does not bring extra communication overhead since such information is piggybacked by the RTS/CTS message when making connection.

In this paper, we do not take into account of energy consumption by the execution of task m of SMD n on the cloud, which is justified since the cloud is in general powered by alternating current and has enough energy to execute offloaded tasks.

Similar to Section 3.3.1, in the case of considering the dependency among tasks, the ready time of task m on cloud, denoted by $RT^{c}_{m,n}$, is given by

$$RT^{c}_{m,n} = \max\left\{FT^{t}_{m,n}, \max_{k\in\mathbf{pred(m)}} FT^{c}_{k,n}\right\}. \tag{14}$$

We can observe that if an immediate predecessor task k of task m is executed locally, then $FT^{c}_{k,n} = 0$. Therefore, $\max_{k\in\mathbf{pred(m)}} FT^{c}_{k,n}$ in Eq. (14) is the time when all the immediate predecessors of task m that are offloaded to the cloud have finished execution on the cloud. In addition, we can find that the cloud can start executing task m only after the task has been completely offloaded to cloud or all the immediate predecessors of task m have been completely executed on the cloud, i.e.,

$$RT^{c}_{m,n} \geq FT^{t}_{m,n}, RT^{c}_{m,n} \geq \max_{k\in\mathbf{pred(m)}} FT^{c}_{k,n}. \tag{15}$$

In particular, if we ignore the time of receiving the outcome of task m, the completion time of task m of SMD n in cloud execution is the sum of the execution time of task m of SMD n and its ready time on the cloud, i.e.,

$$FT^{c}_{m,n} = T^{c,exe}_{m,n} + RT^{c}_{m,n}, \tag{16}$$

As a result, from Eqs. (11), (12), (13), we can give the EEC of computing task m of SMD n on cloud by

$$Z^{c}_{m,n} = \gamma^{T}_{m,n}(FT^{c}_{m,n}) + \gamma^{E}_{m,n}E^{c,trs}_{m,n}(\mathcal{A}). \tag{17}$$

We can observe from Eq. (17) that the low data transmission rate $R_{m,n}$ of SMD n would result in high energy consumption in wireless access and long transmission time for offloading the input data to cloud. In this case, it would be more beneficial for the mobile device to compute the task locally on the mobile device.

4 Problem Formulation

In this section, we will formulate the eDors problem, i.e., determine which tasks should be offloaded onto cloud and how much CPU clock frequency in local computing and transmission power in cloud computing should be used such that the EEC of completing an application is minimized.

For a given task sequence set $\mathcal{M}$ of an application of SMD n, the EEC for this application can be computed by

$$\begin{aligned} Z_n &= \sum_{m=1}^{M} Z_{m,n} = \sum_{m=1}^{M}[(1-a_{m,n})Z^{l}_{m,n} + a_{m,n}Z^{c}_{m,n}] \\ &= \sum_{m=1}^{M}[(1-a_{m,n})(\gamma^{E}_{m,n}E^{l}_{m,n}(f_{m,n}) + \gamma^{T}_{m,n}FT^{l}_{m,n}(f_{m,n})) \\ &\quad + a_{m,n}(\gamma^{T}_{m,n}FT^{c}_{m,n}(P^{T}_{m,n}) + \gamma^{E}_{m,n}E^{c,trs}_{m,n}(P^{T}_{m,n}))], \end{aligned} \tag{18}$$

where $E^{l}_{m,n}$ and $FT^{l}_{m,n}$ are the functions of clock frequency $f_{m,n}$ as shown in Eqs. (3) and (9), respectively. In the meanwhile, $FT^{c}_{m,n}$ and $E^{c,trs}_{m,n}$ are the functions of transmission power $P^{T}_{m,n}$ as given in Eqs. (16) and (12), respectively. $E^{c,trs}_{m,n}$ denotes the energy consumption of wireless transmission.

We aim to provide the optimal computation offloading selection policy $\mathcal{A}^*$, clock frequency control policy $\mathcal{F}^*$ and transmission power allocation policy $\mathcal{P}^*$ such that energy efficiency cost is minimized. Therefore, according to constraints Eqs. (8) and (15), the eDors problem for all SMDs can be formulated as a constrained minimization problem

$$\mathbf{OPT-1}\ \min_{\mathcal{A},\mathcal{F},\mathcal{P}} \sum_{n=1}^{N} Z_n \tag{19}$$

subject to $\forall m \in \mathcal{M}, \forall n \in \mathcal{N}$,

$$
\begin{aligned}
&C1: \sum_{m=1}^{M}[(1-a_{m,n})FT^{l}_{m,n} + a_{m,n}(FT^{c}_{m,n})] \leq T_{n,\max},\\
&C2: (1-a_{k,n})FT^{l}_{k,n} + a_{k,n}FT^{c}_{k,n} \leq RT^{l}_{m,n}, \forall k \in \mathbf{pred}(\mathbf{m}),\\
&C3: FT^{t}_{m,n} \leq RT^{c}_{m,n},\\
&C4: \max_{k \in \mathbf{pred}(\mathbf{m})} FT^{c}_{k,n} \leq RT^{c}_{m,n},\\
&C5: a_{m,n} \in \{0,1\},
\end{aligned}
$$

where $\mathcal{A} = \{a_{m,n} | m \in \mathcal{M}, n \in \mathcal{N}\}, \mathcal{F} = \{f_{m,n} | m \in \mathcal{M}, n \in \mathcal{N}\}, \mathcal{P} = \{P^{T}_{m,n} | m \in \mathcal{M}, n \in \mathcal{N}\}$. Constraint $C1$ is completion time constraint which specifies that the total completion time of all the tasks of an application of SMD n is bounded by the required maximum completion time (i.e., completion time deadline), $T_{n,\max}$. The setting of $T_{n,\max}$ is determined by the latency requirement for an application, that is, the delay-tolerant application can have high maximum completion time; otherwise, for the delay-sensitive application, it is preferred to set low maximum completion time. Local task-precedence requirement $C2$ ensures that task m can start execution only after all its immediate predecessors have completely finished execution. Constraints $C3$ and $C4$ are cloud task-precedence requirement constraints which indicate that task m can begin to be executed on cloud only after the task has been completely offloaded to cloud, as shown in constraint $C3$, or all the immediate predecessors of task m have been completely executed on the cloud, as shown in constraint $C4$. Computation offloading selection constraint $C5$ specifies that task m of SMD n is locally executed on its own device or is offloaded onto the cloud to complete execution.

The key challenge in solving the optimization problem *OPT-1* in (19) is that the integer constraint $a_{m,n} \in \{0,1\}$ makes problem *OPT-1* become a mixed integer programming problem, which is in general non-convex and NP-hard. Thus, similar to the relaxation method in [26], [27], [28], [29], [30], we first relax the binary computation offloading decision variable $a_{m,n}$ to a real number between 0 and 1, i.e., $0 \leq a_{m,n} \leq 1$.

Next, we explore the convexity of the optimization problem *OPT-1* with the relaxed optimization variable $a_{m,n}$.

Theorem 1. *The optimization problem* OPT-1 *with constraints* $C1$-$C5$ *is convex with respect to (w.r.t) the optimization variables* $\{a_{m,n}\}$, $\{f_{m,n}\}$ *and* $\{P^{T}_{m,n}\}$.

Proof. See the proof at the end of the paper. □

Theorem 1 reveals that the optimization problem *OPT-1* in Eq. (19) has a zero duality gap and satisfies the Slater's constraint qualification. The zero-duality-gap result provides an avenue to obtain the optimal solution of the primal problem in Eq. (19) derived from its corresponding dual problem.

5 Distributed Algorithm for eDors Problem

In this section, we solve the problem *OPT-1* with the relaxed constraint $C5$ to provide a distributed eDors algorithm for the policy of computation offloading selection, clock frequency control and transmission power allocation.

5.1 Dual Problem Formulation

The resource allocation policy is derived via solving the dual problem of Eq. (19). For this purpose, we first give the Lagrangian function of the primal problem Eq. (19) by

$$
\begin{aligned}
&L(w,\mu,\mathcal{A},\mathcal{F},\mathcal{P})\\
&= \sum_{n=1}^{N}\sum_{m=1}^{M}[(1-a_{m,n})Z^{l}_{m,n} + a_{m,n}Z^{c}_{m,n}]\\
&\quad + \sum_{n=1}^{N} w_n \sum_{m=1}^{M}[(1-a_{m,n})FT^{l}_{m,n} + a_{m,n}FT^{c}_{m,n}]\\
&\quad - \sum_{n=1}^{N} w_n T^{c}_{n,\max} + \sum_{n=1}^{N}\sum_{m=1}^{M}\mu_{m,n}(FT^{t}_{m,n} - RT^{c}_{m,n}).
\end{aligned} \tag{20}
$$

Lagrangian multiplier $w = [w_n, n = 1, \ldots, N]^T$ is for the completion time constraint $C1$, where w_n denotes the prices of total completion time of an application of SMD n no more than the required maximum completion time. Lagrangian multiplier $\mu = [\mu_{m,n}, m = 1, 2, \ldots, M, n = 1, 2, \ldots, N]^T$ corresponds to cloud task-precedence requirement constraint $C3$, where $\mu_{m,n}$ represents the price for task m to be executed on the cloud only after being completely offloaded to cloud. In fact, these multipliers are penalty factors on the objective function to make it evolve toward its optimum under the corresponding constraints. This is why these multipliers can be defined as the prices that the algorithm has to pay for satisfying the constraints. The expressions of the prices can be given by solving the Lagrangian-dual problem, which can be found in Section 5.5. On the other hand, constraints $C2$ and $C4$ are only related to the immediate predecessors of task m and determine the earliest time of task m starting execution on the local device or the cloud. Therefore, they would be captured by the Karush-Kuhn-Tucker (KKT) conditions when deriving the resource allocation solution later.

The dual problem for the primal problem Eq. (19) is given by

$$
\max_{w,\mu} \min_{\mathcal{A},\mathcal{F},\mathcal{P}} L(w,\mu,\mathcal{A},\mathcal{F},\mathcal{P}) \tag{21}
$$

The dual problem in Eq. (21) is decomposed into a hierarchy of two levels b y using Layering as Optimization Decomposition (LOD) approach [31]. Level 1, the inner minimization in Eq. (21), consists of N subproblems with identical structure that can be solved in a distributed manner. Level 2, the outer maximization in Eq. (21), is the master problem. In the following, we will prove that such decomposition can yield an optimal solution.

Theorem 2. *There exists an optimal solution* $(f^{*}_{m,n}, P^{T*}_{m,n}, w^{*}_{n}, \mu^{*}_{m,n})$ *of the distributed eDors algorithm by LOD approach.*

Proof. It is shown in [31] that the decomposed subproblems by applying LOD approach have an optimal solution if and only if there is no duality gap between the primal problem and its corresponding dual problem. This means that the necessary conditions for applying LOD are convexity and separability. It is known from Theorem 1 that the optimization problem *OPT-1* in Eq. (19) is convex and there is a zero duality gap between Eqs. (19) and (21). In addition, we can observe from Eqs. (19) and (21) that the optimization variables $f^{*}_{m,n}$ and $P^{T*}_{m,n}$ are independent of each other, which implies that the dual problem Eq. (21) is separable. Therefore, the eDors algorithm proposed by LOD approach can yield an optimal solution $(f^{*}_{m,n}, P^{T*}_{m,n}, w^{*}_{n}, \mu^{*}_{m,n})$. □

Based on zero-duality-gap result in Theorem 1, we know that the solution of *OPT-1* problem in Eq. (19) can be derived from its dual problem in Eq. (21). We use an iterative

approach to solving the dual problem Eq. (21) as follows: in each iteration, given dual variables w and μ, we first solve N subproblems to obtain the primal variables $\mathcal{A}, \mathcal{F}, \mathcal{P}$ by applying the KKT conditions; Then by using the primal variables, we update the dual variables via the subgradient method.

In the following, we give the distributed subalgorithms of computation offloading selection, clock frequency control and transmission power allocation.

5.2 Computation Offloading Selection

Computation offloading selection subalgorithm aims to determine which tasks of an application are offloaded onto the cloud such that the energy efficiency cost of completing the application is minimized while the task-precedence requirement is preserved. Let $Cost^l_{m,n} = Z^l_{m,n} + w_n FT^l_{m,n}$ denote the computation cost on the local device and $Cost^c_{m,n} = Z^c_{m,n} + w_n FT^c_{m,n}$ denote the computation cost on the cloud. The optimal computation offloading selection policy can be obtained by solving the following minimization problem

$$\min_{a_{m,n}} [Cost^l_{m,n} + a_{m,n}(Cost^c_{m,n} - Cost^l_{m,n})] \tag{22}$$

subject to constraints $C2$, $C4$ and $C5$.

It is clear that the objective function in Eq. (22) is a linear function on variable $a_{m,n}$. We have the following observations. If $Cost^c_{m,n} \geq Cost^l_{m,n}$, the objective function in Eq. (22) achieves minimum when $a_{m,n} \in [0,1]$ reaches minimum; on the contrary, if $Cost^c_{m,n} < Cost^l_{m,n}$, then the objective function has the minimum value when $a_{m,n}$ reaches maximum, as shown in Fig. 3.

Therefore, we have the computation offloading selection policy as follows

$$a_{m,n} = \begin{cases} 1, & \text{if } Cost^c_{m,n} < Cost^l_{m,n}, \\ 0, & \text{otherwise.} \end{cases} \tag{23}$$

This indicates that when the computation cost on the cloud is less than that on the local device, it is beneficial that task m is offloaded onto the cloud to compute. We can observe from the definitions of $Cost^c_{m,n}$ and $Cost^l_{m,n}$ that the computation offloading selection policy of SMD n for task m depends on not only the specification of SMD n such as clock frequency and transmission power, but also the maximum completion time of its immediate predecessors and the offloading data rate. Therefore, in order to determine the optimal computation offloading selection, an important task we need to do is to get the completion time of every immediate predecessor $FT^c_{k,n}$ and $FT^l_{k,n}$ computes the ready time of task m on the local device or the cloud, $RT^l_{m,n}$ and $RT^c_{m,n}$, and calculate the offloading data rate $R_{m,n}$ based on the current communication channel conditions.

5.3 Clock Frequency Control

The goal of the clock frequency control policy is to optimally set the clock frequency of the mobile device such that the energy efficiency cost of the application execution is minimized. Clearly, the policy works in the case when a task is executed on the local device, i.e., $a_{m,n} = 0$. The policy of clock frequency control can be derived by solving the following optimization problem

$$\min_{f_{m,n}} [Z^l_{m,n} + w_n FT^l_{m,n} + \mu_{m,n}(FT^t_{m,n} - RT^c_{m,n})] \tag{24}$$

subject to constraints $C2$ and $C4$.

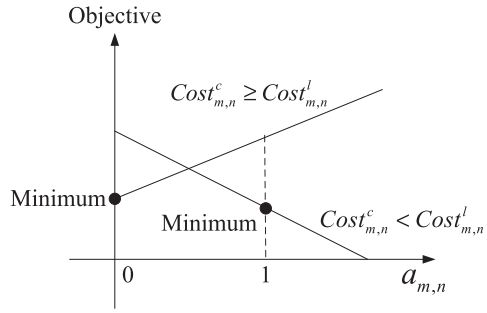


Fig. 3. Illustration of the objective function with the minimum value.

Similar to the proof of Theorem 1, it is easy to verify that the optimization problem Eq. (24) is convex w.r.t $f_{m,n}$. The objective function $F(f_{m,n})$ in Eq. (24) can be rewritten as

$$\begin{aligned} F(f_{m,n}) = {} & \gamma^E_{m,n} E^l_{m,n} + \mu_{m,n}(FT^t_{m,n} - RT^c_{m,n}) \\ & + (\gamma^T_{m,n} + w_n)(L_{m,n} f^{-1}_{m,n} + RT^l_{m,n}). \end{aligned} \tag{25}$$

It is known that $RT^l_{m,n}$, $FT^t_{m,n}$ and $RT^c_{m,n}$ are independent of $f_{m,n}$. Using standard convex optimization techniques and the KKT conditions [32], the clock frequency control policy is given by

$$f_{m,n} = \sqrt[3]{\frac{\gamma^T_{m,n} + w_n}{2\kappa \gamma^E_{m,n}}}. \tag{26}$$

We can observe that the clock frequency of SMD n executing task m depends on the weight of computation completion time, $\gamma^T_{m,n}$, the weight of energy consumption, $\gamma^E_{m,n}$, and the price for the required application completion time deadline, w_n. The analytical results in Eq. (26) reveal engineering insights on the optimal clock frequency configuration for the mobile device execution, as elaborated in the following propositions.

Proposition 1. *The optimal clock frequency increases monotonically with weight $\gamma^T_{m,n}$ and price w_n. This implies that the low computation completion time requires the CPU to accelerate clock frequency to meet the application completion deadline.*

Proposition 2. *The optimal clock frequency is inversely proportional to the weight of energy consumption, $\gamma^E_{m,n}$. This indicates that saving energy consumption can be achieved by reducing clock frequency.*

Moreover, we can find that for given $\gamma^T_{m,n}$ and $\gamma^E_{m,n}$, the price w_n plays an important role in the adjustment of the optimal clock frequency configuration of SMD n, which is affected by the completion time of its immediate predecessors $FT^c_{k,n}$ and $FT^l_{k,n}$, $k \in \mathbf{pred(m)}$.

5.4 Transmission Power Allocation

The transmission power allocation policy aims to optimally allocate the transmission power for each task of the mobile device such that the EEC of cloud computing the task is minimized. Clearly, this policy is valid in the case that the task needs to be offloaded onto the cloud, i.e., $a_{m,n} = 1$. In this case, the transmission power allocation policy can be obtained by solving the following minimization problem

$$\min_{P^T_{m,n}} \sum_{n=1}^{N} \sum_{m=1}^{M} [Z^c_{m,n} + w_n FT^c_{m,n} + \mu_{m,n}(FT^t_{m,n} - RT^c_{m,n})] \tag{27}$$

subject to constraints $C2$ and $C4$.

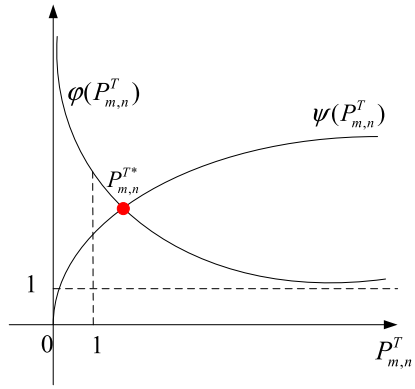


Fig. 4. The intersection point is the solution of Eq. (30).

We let $Y(P^T_{m,n})$ denote the objective function in Eq. (27) and then according to Eqs. (11), (12), (13), (14), (15), (16), (17), we can give $Y(P^T_{m,n})$ of two forms based on two different values of $RT^c_{m,n}$.

Case I. $FT^t_{m,n} > \max_{k\in\mathbf{pred}(\mathbf{m})} FT^c_{k,n}$, i.e., $RT^c_{m,n} = FT^t_{m,n}$. $Y(P^T_{m,n})$ can be rewritten as

$$Y(P^T_{m,n}) = \sum_{n=1}^{N}\sum_{m=1}^{M}[(\gamma^T_{m,n}+w_n)(T^{c,exe}_{m,n}+FT^t_{m,n}) + \gamma^E_{m,n}E^{c,trs}_{m,n}(\mathcal{A})], \quad (28)$$

where $FT^t_{m,n} = T^{c,trs}_{m,n}(\mathcal{A}) + RT^{trs}_{m,n}$. $RT^{trs}_{m,n}$ denotes the ready time of task m being transmitted on the wireless sending channel in order to preserve the task-precedence requirement, which is a constant for task m.

Case II. $FT^t_{m,n} \leq \max_{k\in\mathbf{pred}(\mathbf{m})} FT^c_{k,n}$, i.e., $RT^c_{m,n} = \max_{k\in\mathbf{pred}(\mathbf{m})} FT^c_{k,n}$, which is a function independent of $P^T_{m,n}$. $Y(P^T_{m,n})$ can be expressed by

$$Y(P^T_{m,n}) = \sum_{n=1}^{N}\sum_{m=1}^{M}[(\gamma^T_{m,n}+w_n)(T^{c,exe}_{m,n}+RT^c_{m,n}) + \gamma^E_{m,n}E^{c,trs}_{m,n}(\mathcal{A}) + \mu_{m,n}(FT^t_{m,n}-RT^c_{m,n})]. \quad (29)$$

Next, we first give the transmission power allocation policy for Case I.

It is easy to verify from the proof of Theorem 1 that $Y(P^T_{m,n})$ in Eq. (28) is convex w.r.t $P^T_{m,n}$. Using the KKT conditions [32], the transmission power allocation policy is given by

$$\frac{\alpha_{m,n}}{\varpi_{m,n}+P^T_{m,n}H_{m,n}} + 1 = \ln\left(1+\frac{P^T_{m,n}H_{m,n}}{\varpi_{m,n}}\right), \quad (30)$$

where $\alpha_{m,n} = (\gamma^T_{m,n}+w_n)H_{m,n}/\gamma^E_{m,n} - \varpi_{m,n}$ and $\varpi_{m,n} = \Gamma\sigma^2$ denotes the power of the thermal noise and the interferences at access point s receiving task m of SMD n from transmitters other than SMD n.

It is not difficult to observe that the optimal transmission power is the solution of Eq. (30), i.e., the intersection point between two equations, $\varphi(P^T_{m,n})$ and $\psi(P^T_{m,n})$, as shown in Fig. 4, which can be given by $\varphi(P^T_{m,n}) = \frac{\alpha_{m,n}}{\varpi_{m,n}+P^T_{m,n}H_{m,n}} + 1$, and $\psi(P^T_{m,n}) = \ln(1+\frac{P^T_{m,n}H_{m,n}}{\varpi_{m,n}})$. However, Eq. (30) is a transcendental equation, and in general does not have a closed-form solution for $P^T_{m,n}$. Thus we can only obtain its approximate solution by Newton iteration method, i.e., the transmission power $P^T_{m,n}$ is updated iteratively by

$$P^T_{m,n}(t+1) = P^T_{m,n}(t) - \frac{\varphi(P^T_{m,n}(t)) - \psi(P^T_{m,n}(t))}{\varphi'(P^T_{m,n}(t)) - \psi'(P^T_{m,n}(t))}, \quad (31)$$

where $\varphi'(\cdot)$ and $\psi'(\cdot)$ denote the first-order derivative w.r.t. $P^T_{m,n}(t)$.

It is shown in [33] that the method usually converges, provided that the initial value $P^T_{m,n}(0)$ is close enough to the zero $P^{T*}_{m,n}$, and that $\varphi'(P^T_{m,n}(t)) \neq \psi'(P^T_{m,n}(t))$. Furthermore, for the equation with a simple root, the convergence is at least quadratic in a neighborhood of the zero.

It is worth noting that in the process of solving Eq. (29) by the KKT conditions, to make the solution simple and feasible, we only take into account the interference of other mobile devices to access point receiving the tasks of SMD n, while neglecting the interference of SMD n to the access point receiving the tasks of other SMDs.

Now, we consider the power allocation policy in Case II. Similar to the derivation process of the transmission power in Case I, based on the KKT conditions for Eq. (29), the transmission power allocation policy is given by

$$\frac{\beta_{m,n}}{\varpi_{m,n}+P^T_{m,n}H_{m,n}} + 1 = \ln\left(1+\frac{P^T_{m,n}H_{m,n}}{\varpi_{m,n}}\right), \quad (32)$$

where $\beta_{m,n} = \mu_{m,n}H_{m,n}/\gamma^E_{m,n} - \varpi_{m,n}$.

Similar to Case I, the transmission power $P^T_{m,n}$ for Case II can be iteratively updated by

$$P^T_{m,n}(t+1) = P^T_{m,n}(t) - \frac{\phi(P^T_{m,n}(t)) - \psi(P^T_{m,n}(t))}{\phi'(P^T_{m,n}(t)) - \psi'(P^T_{m,n}(t))}, \quad (33)$$

where $\phi(P^T_{m,n}(t) = \frac{\beta_{m,n}}{\varpi_{m,n}+P^T_{m,n}H_{m,n}}$ and $\beta_{m,n} = \mu_{m,n}H_{m,n}/\gamma^E_{m,n} - \varpi_{m,n}$.

We can draw following observations from Eqs. (30) and (33) that (i) the optimal transmission power is closely related to the weight of computation completion time, $\gamma^T_{m,n}$, the weight of energy consumption, $\gamma^E_{m,n}$, the wireless channel conditions and the interferences from other SMDs; (ii) for Case I, the optimal transmission power depends on the price for satisfying the required completion time deadline while for Case II, the power is determined by the price for achieving cloud computing.

5.5 Lagrangian Multiplier Update

The Level 2 master problem in Eq. (21) can be solved by using the subgradient method. We can update the set of Lagrange multipliers for a given set of $\mathcal{F}, \mathcal{P}$ by for local computing

$$w_n(k+1) = \left[w_n(k) + \vartheta(k)\left(T_{n,\max} - \sum_{m=1}^{M} FT^l_{m,n}\right)\right]^+ \quad (34)$$

for cloud computing

$$w_n(k+1) = \left[w_n(k) + \vartheta(k)\left(T_{n,\max} - \sum_{m=1}^{M} FT^c_{m,n}\right)\right]^+ \quad (35)$$

$$\mu_{m,n}(k+1) = \left[\mu_{m,n}(k) + \vartheta(k)\left(RT^c_{m,n} - \sum_{m=1}^{M} FT^t_{m,n}\right)\right]^+, \quad (36)$$

where index $k > 0$ is the iteration index and $\vartheta(k)$ is positive iteration step size. Then, the updated Lagrange multipliers in Eqs. (34), (35), (36) can be used for updating the resource allocation policy in Eqs. (23), (26), (31) and (33). As the primal

problem is jointly convex with respect to the optimization variables in Theorem 1, it is guaranteed that the primal optimal solution can be obtained by solving the problems in Level 1 and Level 2 iteratively, provided that the chosen step sizes, $\vartheta(k)$, are sufficiently small.

The proposed algorithm is described in Algorithm 1. The underlying observations behind the algorithm are that (i) the task sequence of of an application and the immediate predecessors of a task play an imperative role in its offloading selection decision; (ii) in the clock frequency control policy, transmission power has no change. Similarly, in the transmission power allocation policy, clock frequency $f_{m,n}$ keeps unchanged. The convergence of the proposed iterative algorithm is guaranteed when the iteration step size $\vartheta(k)$ satisfies the following theorem regardless of initial Lagrangian multipliers.

Theorem 3. *The update iterations of $f_{m,n}$ and $P^T_{m,n}$ by Eqs. (26) and (31) will converge to the optimal solution $(f^*_{m,n}, P^{T*}_{m,n})$ provided that a sequence of diminishing step sizes satisfy*

$$\vartheta(k) \to 0, \sum_{k=1}^{\infty} \vartheta(k) \to \infty \text{ and } \sum_{k=1}^{\infty} \vartheta(k)^2 < \infty.$$

Proof. As shown in Theorem 1, the optimization problem *OPT-1* is convex w.r.t the optimization variables $\{f_{m,n}\}$ and $\{P^T_{m,n}\}$, which implies that the iterations in Eqs. (26) and (31) converge to the optimal dual solution $(w^*_n, \mu^*_{m,n})$ with the above diminishing step size. The detailed proof can be found in [32]. □

5.6 Scheduling Overhead and System Complexity

As shown in Algorithm 1, task dependency constraints play the imperative role in offloading decision because we have to compute $RT^l_{m,n}, RT^{trs}_{m,n}$ in local execution and $RT^c_{m,n}$ in cloud execution based on task dependencies. Therefore, one of task scheduling overheads is additional memory usage to save the temporary results of the ready time of all the immediate predecessors of a task. Another scheduling overhead is that in transmission power allocation, the eDors algorithm has to take the extra computation resource including CPU and memory to perform the additional K inner-loop iterations to achieve convergence of transmission power at each outer loop iteration.

It is not difficult to observe from Algorithm 1 that for a task m, the time complexity of clock frequency control subalgorithm is $T(M) = 2M + 9$ since the computation of $RT^l_{m,n}, RT^{trs}_{m,n}$ for task m needs to wait for the completion of its M predecessors in the worst case while all other computations each takes one time of iteration, respectively. Similarly, for a task m, the time complexity of transmission power allocation subalgorithm is $T(M) = 2M + K + 11$. Therefore, the time complexity of the eDors algorithm is $T(M) = M(4M + K + 20) = O(M^2)$ for an application. It is worth noting that we do not consider the overhead in the cloud because we assume that (i) the RAM of the cloud server executing the offloaded tasks is high enough, and (ii) wireless channel parameters keep fixed during offloading. In practice, the number of iterations of algorithm convergence also depends on diminishing step sizes.

6 Performance Evaluation

In this section, we evaluate the performance of the proposed algorithm.

Algorithm 1. Iterative eDors algorithm for SMD n

Require:
$\mathcal{M}$:a sequence of M tasks of SMD n;
$\mathbf{pred(m)}$: the set of immediate predecessors of task m;
ϵ, ξ: an infinitesimal number;
Ensure:
$\{\mathcal{A}, \mathcal{F}, \mathcal{P}\}$: optimal resource allocation policy;
Initialize: $D_{m,n}$, $L_{m,n}$, $\gamma^E_{m,n}$, $\gamma^T_{m,n}$, $\vartheta(t)$, w_n and $\mu_{m,n}$, $\{f_{m,n}\}$, $\{P^T_{m,n}\}$ and iteration index $t \leftarrow 1$;
for $m = 1$ to M **do**
  /* *Clock frequency control* */
  $t = 1$;
  repeat
    Compute $T^l_{m,n}$, $E^l_{m,n}$ by (2)-(3), respectively;
    if pred(m) == ∅ then
      $RT^l_{m,n} = 0$, $RT^{trs}_{m,n} = 0$
    else
      Compute $RT^l_{m,n} = \max_{k \in \mathbf{pred(m)}} \max\{FT^l_{k,n}, FT^c_{k,n}\}$;
      Compute $RT^{trs}_{m,n} = \max_{k \in \mathbf{pred(m)}} \{FT^l_{k,n}\}$;
    end if
    Compute $FT^l_{m,n}$ by (9) and $Z^l_{m,n}$ by (10);
    Calculate $Cost^l_{m,n} = Z^l_{m,n} + w_n FT^l_{m,n}$;
    Compute the clock frequency $f_{m,n}$ by (26);
    Update Lagrangian multipliers $w_n(t+1)$ by (34);
    $t = t + 1$;
  until $|w_n(t+1) - w_n(t)| < \epsilon$
  /* *Transmission power allocation* */
  $t = 1, k = 1$;
  repeat
    Compute $R_{m,n}$ by (1);
    Compute $T^{c,trs}_{m,n}$, $E^{c,trs}_{m,n}$, $T^{c,exe}_{m,n}$ by (11)-(13);
    Compute $FT^t_{m,n} = T^{c,trs}_{m,n} + RT^{trs}_{m,n}$;
    if pred(m) == ∅ then
      $RT^c_{m,n} = T^{c,trs}_{m,n}$;
    else
      Compute $RT^c_{m,n}$ by (14);
    end if
    Compute $FT^c_{m,n}$, $Z^c_{m,n}$ by (16)-(17), respectively;
    Compute $Cost^c_{m,n} = Z^c_{m,n} + w_n FT^c_{m,n}$;
    if $FT^t_{m,n} > \max_{k \in \mathbf{pred(m)}} FT^c_{k,n}$ **then**
      repeat
        Compute $\varphi(P^T_{m,n}(k)), \psi(P^T_{m,n}(k)), \varphi'(P^T_{m,n}(k), \psi'(P^T_{m,n}(k))$
        Compute $P^T_{m,n}(k+1)$ by (31);
        $k = k + 1$
      until $|P^T_{m,n}(k+1) - P^T_{m,n}(k)| < \xi$
    else
      repeat
        Compute $\phi(P^T_{m,n}(k)), \psi(P^T_{m,n}(k)), \phi'(P^T_{m,n}(k), \psi'(P^T_{m,n}(k))$
        Compute $P^T_{m,n}(k+1)$ by (33);
        $k = k + 1$
      until $|P^T_{m,n}(k+1) - P^T_{m,n}(k)| < \xi$
    end if
    Update Lagrangian multipliers $w_n(t+1)$, $\mu_{m,n}(t+1)$ by (35) and (36), respectively;
    $t = t + 1$;
  until $|\mu_{m,n}(t+1) - \mu_{m,n}(t)| < \epsilon$
  /* *Computation offloading selection* */
  if $Cost^c_{m,n} < Cost^l_{m,n}$ **then**
    $a_{m,n} = 1$
  else
    $a_{m,n} = 0$
  end if
end for

Fig. 5. Experimental testbed setting.

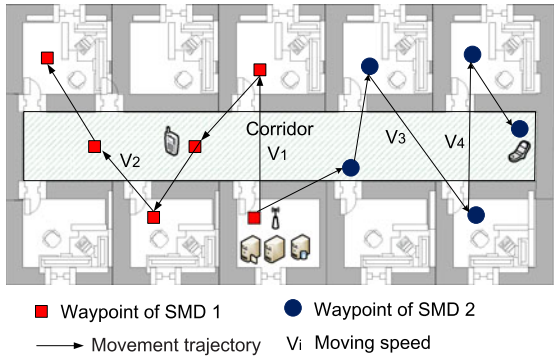


Fig. 6. Movement trajectory of mobile users.

6.1 Experiment Profile

We first consider the mobile cloud computing scenario that $N = 5$ smartphones are randomly scattered over a $50\ \mathrm{m} \times 10\ \mathrm{m}$ region and the wireless access base-station is located in the center of the region. For the wireless access, we set the channel bandwidth $W = 5$ MHz, and the thermal noise power $\sigma^2_{m,n} = 50$ dBm. We set the channel gain $H_{m,n} = d^{\zeta}_{n,s}$ from SMD n to access point s, where $d_{n,s}$ is the distance between mobile device n and wireless access point s, and $\zeta = 4$ is the path loss factor. We set the initial decision weights $\gamma^E_{m,n} = \gamma^T_{m,n} = 0.5$.

To evaluate the effectiveness of dynamic offloading policy, we employ the computation partitioning scheme in [17] to partition a face recognition application in [2] into 20 tasks. The face recognition application is separated into two phases: face detection and face recognition. Similar to [2], we develop our face detection and face recognition programs by Microsoft Visual Studio 2010 C++, and Open CV libraries [34]. We utilize the Haar Features and Haar Classifiers described in [35] to perform face detection and employ the widely-accepted Eigenfaces approach [36] to execute face recognition, i.e., determine the likelihood of each detected face matching one of the template images in a database. Then we implement and test our proposed dynamic offloading policy on a real testbed as shown in Fig. 5, where five smartphones Android-based and the cloud server consists of H3C FlexServer R390 and Dell Power Edge M910. The bandwidth that the SMDs connect to the access points through Wi-Fi is 150Mbps. The data size of the computation tasks and the total number of the CPU cycles (i.e., computing load) follow the Gaussian distribution $CN(\mu_1, \sigma^2_1)$ and $CN(\mu_2, \sigma^2_2)$, where the mean $\mu_1 = 200$KB, $\mu_2 = 1000$ Mega cycles, and the standard deviation $\sigma_1 = 50$ and $\sigma_2 = 100$. For the sake of illustration, we take 10 of the partitioned tasks as observation objects, the dependency of which are as shown in Fig. 2. The data size and the total number of the CPU cycles of 10 tasks of an example smartphone are given by [81.04 178.95 96.12 262.69 124.35 205.17 212.40 176.45 233.97 245.61] KB and [896 984 1091 791 1184 973 1073 1021 860 964] Mega cycles, respectively. We use the computing Load-input Data Ratio (LDR) to characterize the complexity of a task, i.e., $LDR_{m,n} = L_{m,n}/D_{m,n}$. Without loss of generality, we can let the LDRs of 10 tasks be [11.056 5.499 11.35 3.011 9.522 4.742 5.052 5.786 3.676 3.925] Mega cycles/KB input data and the diminishing step size $\vartheta(k) = \frac{1+\beta}{k+\beta}$, where β is a fixed positive integer.

In our real experiment, we build the radio frequency (RF) environment by the WiFi connection between the Android smartphones and the Access Points (APs) near the cloud server located in the middle room. We choose the wireless router of TP-LINK TL-WDR5620 as the APs because it has stronger ability to go through the walls so that the WiFi signals of the APs in the middle room can cover the whole laboratory. Furthermore, to achieve different radio frequency environments, we set some waypoints with different distances and wall shadows to the cloud sever so as to obtain the corresponding RF parameters such as channel gains and SINRs at these waypoints. Therefore, the fixed RF parameters in the environment can be guaranteed because these parameters are measured at the predetermined waypoints. In the experiment, we make Android smartphones travel along the predefined waypoints at the given speeds as shown in Fig. 6, while stopping at each waypoint for a predefined amount of time.

In addition, the changing of CPU frequencies of Android smartphones required by our algorithms in the experiment is based on the dynamic voltage and frequency scaling technique [6]. The core idea of the technique is that according to the different computing needs of applications running on the chip, both operating frequency and voltage on the CPU chip are adjusted dynamically (on the same chip, the higher the frequency, the higher the required voltage), to save energy. It is usually possible to control the voltages supplied to the CPU, RAM, PCI, and PCI Express (or AGP) port through the BIOS. At present, many chips support the DVFS technique, such as Intel's SpeedStep and ARM's IEM (Intelligent Energy Manager) and AVS (Adaptive Voltage Scaling).

In particular, we will evaluate the convergence of the proposed eDors algorithm by numerical analysis in Section 6.2 and demonstrate the impact of weights, $\gamma^E_{m,n}$ and $\gamma^T_{m,n}$ on the energy consumption and computation delay by Monte Carlo simulation in Section 6.3. Furthermore, we compare energy efficiency cost, energy consumption and application completion time of several rated algorithms in Sections 6.4 to 6.6 by experiment on the real testbed. It is clear that analytical results guarantee the convergence of simulation results and experimental results, while simulation results provide the criterion for weight settings of energy consumption and computation delay. At the same time, the experimental results imply the correctness and practicability of analytical results and simulation results.

6.2 Convergence and Impact of Task Complexity

In this subsection, we evaluate the convergence of the proposed eDors algorithm and the impact of task complexity

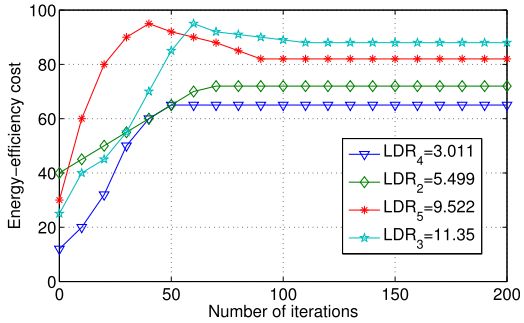


Fig. 7. Simulation dynamics of EECs of tasks with different LDRs.

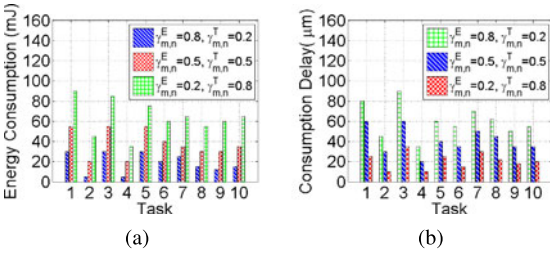


Fig. 8. Simulation result comparison of energy consumption and computation delay for different $\gamma_{m,n}^{E}$ and $\gamma_{m,n}^{T}$. (a) Energy consumption. (b) Computation delay.

on EECs by numerical analysis on Lenovo Thinkpad T430s, where we choose tasks 2-5 as observation objects, which is because they have the same immediate predecessors, i.e., task 1. For the convenience of comparison, we assume that these tasks are locally executed.

We plot the convergence of tasks 2-5 by the proposed eDors algorithm and the dynamics of their EECs in Fig. 7. We can observe that (i) the proposed eDors algorithm can achieve converge about 60-80 iterations for all tasks. An iteration takes 0.3-0.5 ms, therefore, our eDors algorithm takes at most 40 ms to finish the most complex task. In practice, the exact running time for a task is determined by the computation capacity of the mobile device and cloud sever, the complexity of the task, the immediate predecessors of the task, and the ready time of the task. Accordingly, the execution time of an iteration or a task varies with communication condition and computation environment; (ii) the task with larger LDR has the lower speed of convergence, that is, the complexity of task can increase the convergence time. This is because the task with larger LDR would have higher probability to be offloaded onto the cloud; (iii) the task with larger LDR has higher energy efficiency cost, which implies that large LDR will increase the difficulty for a task to achieve energy efficiency. Therefore, in the computation partitioning, the proper LDR setting of tasks plays an imperative role in improving energy efficiency and reducing execution delay.

6.3 Impact of Weights $\gamma_{m,n}^{E}$ and $\gamma_{m,n}^{T}$

In this subsection, we evaluate the impact of weights, $\gamma_{m,n}^{E}$ and $\gamma_{m,n}^{T}$ on the energy consumption and computation delay of tasks with different number of predecessors by Monte Carlo simulation, where we assume that wireless offloading channel follows block fading. In the simulation, the processing time of each cloud server is a fixed value, chosen from a uniform distribution between 1 and 10 ms to complete each task. The communication latency between the SMDs to cloud server is also a fixed value, chosen from a uniform

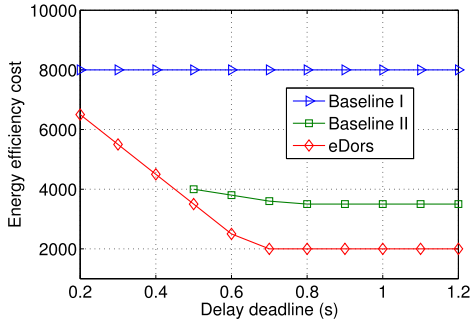


Fig. 9. Experimental result comparison of energy efficiency cost on execution strategy.

distribution between 10 ms and 100 ms to send a packet to the cloud server or from the cloud server back to the SMDs. Other parameter settings are shown in Section 6.1.

Fig. 8 depicts the comparison results of energy consumption and computation delay for different settings of $\gamma_{m,n}^{E}$ and $\gamma_{m,n}^{T}$. We can observe that for a given task, the energy consumption increase as the $\gamma_{m,n}^{E}$ decreases, however, the changes of the computation delay are opposite. This is reasonable since a large $\gamma_{m,n}^{E}$ will lead to the increase of $\alpha_{m,n}$ and $\varphi(P_{m,n}^{T})$, which in turn causes the decrease of transmission power in cloud execution. Moreover, a large $\gamma_{m,n}^{E}$ will result in the increase of clock frequency $f_{m,n}$ by Eq. (26) in local execution. Therefore, increasing $\gamma_{m,n}^{E}$ can reduce energy consumption in both cloud execution and local execution, and furthermore, lower computation delay. Notice that the computation delay here is simply the execution time and does not accumulate the completion time of predecessors. More interestingly, we can find from Fig. 8 that for the tasks with the same number of immediate predecessors, the LDRs of the predecessors have more impacts on computation delay than energy consumption. Also, we can observe that more immediate predecessors can bring slightly more energy consumption and computation delay, such as tasks 9 and 10 shown in Fig. 8.

6.4 Comparison of Energy Efficiency Cost on Execution Strategy

In this subsection, we compare the proposed eDors policy with other two execution strategies, i.e., local execution and cloud execution [9], under the hard completion time deadline constraint. We implement two baseline algorithms, Baseline I and Baseline II, on Android smartphones and H3C and Dell cloud servers to achieve face recognition application. Baseline I demonstrates that all the tasks are executed locally on the mobile device; and Baseline II implies that all the tasks are offloaded to the cloud for execution. Fig. 9 plots the comparison of energy efficiency costs of the proposed eDors algorithm and Baseline algorithms I and II for the same application profile.

We can draw several observations from Fig. 9. First, compared to Baseline I, the eDors algorithm can reduce the energy efficiency cost significantly. Almost 4 times of energy efficiency cost can be reduced by the eDors algorithm when it stays stable. This is because that our eDors algorithm can optimally select tasks to be offloaded on the could to execute according the computation cost on the cloud and the local device. Second, compared to Baseline II, the eDors algorithm has also lower energy efficiency cost

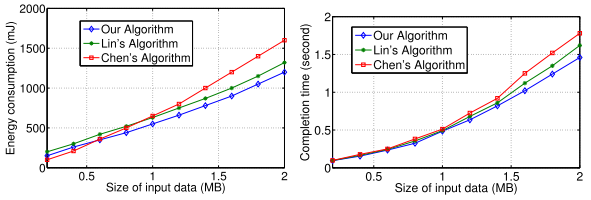


Fig. 10. Comparison of energy consumption and application completion time for different algorithms. (a) Energy consumption. (b) Application completion time.

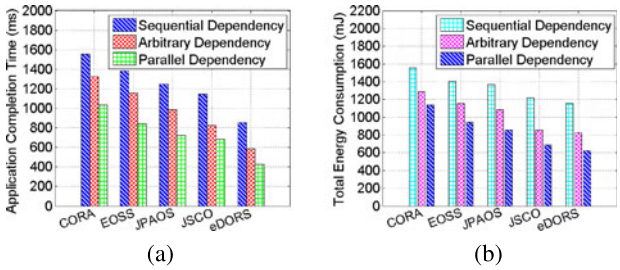


Fig. 11. Comparison of energy efficiency cost. (a) Application completion time. (b) Total energy consumption.

when Baseline II becomes applicable. In the case of low completion time deadline, Baseline II for the cloud execution cannot be used. This is because that it takes a long time for the remote execution to transmit the input data, which violates the delay constraint $C1$ of the application. Third, the energy efficiency cost by the eDors algorithm decreases rapidly as the completion time deadline increases, then becomes stable when the completion time deadline $T_{n,\max}$ is 0.7s, while it always keeps unchanged at a high cost by Baseline I. This is because that at the low completion time deadline, most of tasks are executed on the local device to satisfy the complete time constraint $C1$.

6.5 Comparison of Energy Consumption and Completion Time

In this subsection, we first compare the energy consumption and application completion time of the eDors algorithm, the task scheduling algorithm in [6] called Lin's Algorithm, and the offloading game algorithm in [7] named Chen's Algorithm for different input data sizes. These algorithms are implemented on the real testbed with the same parameter settings and channel conditions to achieve face recognition application. Fig. 10 depicts the energy consumption and completion time for the three algorithms.

We can observe that for large-scale input data, our algorithm and Lin's algorithm can reduce energy consumption significantly by adaptively adjusting clock frequency and transmission power. However, our algorithm always has less energy consumption compared to Lin's algorithm, which is justified since our algorithm not only applies the dynamic voltage and frequency scaling technique to control CPU clock frequency in local computing, but also takes advantage of transmission power control mechanism to reduce energy consumption in cloud computing; In addition, we can also observe from Fig. 10b that the application completion time by three offloading algorithms increases with the data size for computation offloading, due to the fact that a larger data size requires more time for computation offloading via wireless communication. Moreover, all the algorithms have almost the same application completion time when the data size of an application is less than 1 MB. However, we see that the application completion time by our eDors algorithm increases slowly when the size of input data is large.

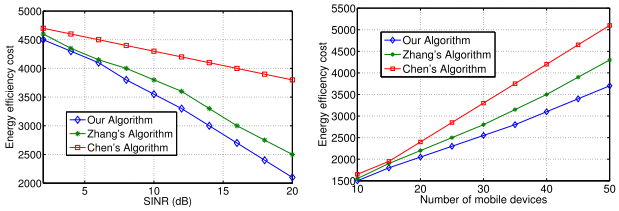


Fig. 12. Comparison of energy efficiency cost for different SINRs and number of mobile devices. (a) SINRs. (b) Number of mobile devices.

Furthermore, we compare our proposed eDors algorithm with (1) cooperative resource allocation (CORA) in MCC in [11], (2) energy-optimal scheduling strategy (EOSS) in [9], (3) joint dynamic transmission power allocation and offload scheduling (JPAOS) in [12], (4) joint scheduling and computation offloading in [13]. We analyze the performance of the five schemes w.r.t the critical parameters of completion time and energy consumption on the real 20-task application with three kinds of task dependencies, i.e., arbitrary dependencies, fully parallel dependencies and fully sequential dependencies.

In Fig. 11a, we compare the application completion time of our eDors algorithm with CORA, EOSS, JPAOS and JSCO. Similar to [13], this comparison is normalized to the scheme with local execution of all the tasks. It is observed that eDors can shorten 45.0, 38.2, 31.3, and 25.4 percent application completion time in comparison to the schemes using CORA, EOSS, JPAOS and JSCO, respectively, in the case of sequential dependency. This is mainly because (i) the goal of these schemes is to maximize the benefit of the mobile cloud service providers or minimize the energy consumption rather than minimize the application completion time; (ii) CORA, EOSS and JPAOS do not consider the impact of task dependency on scheduling and offloading strategy.

Fig. 11b depicts total energy consumption of the 5 schemes on the real 20-task application. Similarly, this comparison is also normalized to the scheme with local execution of all the tasks. We can observe that eDors can save 26.2, 17.7, 15.3, and 5.1 percent total energy consumption in comparison to the schemes using CORA, EOSS, JPAOS and JSCO, respectively, in the worst case of sequential dependency. This is because CORA, EOSS and JSCO do not consider the optimal allocation of transmission power in cloud computing to reduce energy consumption. Although JPAOS achieved joint radio resources allocation and offload scheduling, it does not take into account the impact of clock frequency control and task dependency on energy consumption.

6.6 Impact of Channel Gain and Number of Mobile Devices

In this subsection, we compare the energy efficiency cost for different SINRs and number of mobile devices by our eDors algorithm, Chen's Algorithm in [7] and the collaborative task execution [10] named Zhang's Algorithm. For the sake of fairness, we implement these algorithms on the real testbed with the same parameter settings and environment conditions.

Fig. 12 plots the energy efficiency cost for different SINRs and number of mobile devices. We can observe from Fig. 12a that the energy efficiency costs of all algorithms

decrease as the SINR becomes larger, i.e., the wireless channel conditions get improved. This indicates that all three algorithms consider the impact of the changes of offloading data rate caused by wireless channel conditions on energy consumption and task execution time. However, it is not difficult to see that our algorithm has the most energy efficiency cost reduction compared to other two algorithms.

Furthermore, we can find that as the number of mobile devices increases, the energy efficiency cost of all three algorithms increases significantly. It is clear that too many mobile devices offloading computation task simultaneously will make the communication resource much scarce and cause severe interference and low data rate. However, compared to the Chen's and Zhang's algorithms, our algorithm has the slowest increases of energy efficiency cost. This is because that our algorithm takes into account the impact of other devices' interference on offloading selection and transmission power allocation of a mobile device so that the energy efficiency cost can be minimized.

7 Proof of Theorem 1

We first prove that the objective function $Z_{m,n} = (1 - a_{m,n}) Z^l_{m,n} + a_{m,n} Z^c_{m,n}$ in Eq. (19) is jointly convex w.r.t. the optimization variables $a_{m,n}$, $f_{m,n}$ and $P^T_{m,n}$. Then we show the convexity of constraints $C1 - C5$.

By Eqs. (9) and (10) in Section 3.3.1 and Eqs. (16) and (17) in Section 3.3.2, the objective function can be transformed as

$$\begin{aligned} Z_{m,n} &= (1 - a_{m,n})(\gamma^E_{m,n} E^l_{m,n} + \gamma^T_{m,n} FT^l_{m,n}) \\ &\quad + a_{m,n}(\gamma^T_{m,n}(FT^c_{m,n}) + \gamma^E_{m,n} E^{c,trs}_{m,n}(\mathcal{A})) \\ &= (1 - a_{m,n})[\gamma^E_{m,n} E^l_{m,n} + \gamma^T_{m,n}(T^l_{m,n} + RT^l_{m,n})] \\ &\quad + a_{m,n}(\gamma^T_{m,n}(T^{c,exe}_{m,n} + RT^c_{m,n}) + \gamma^E_{m,n} E^{c,trs}_{m,n}(\mathcal{A})) \end{aligned}$$

Since $RT^l_{m,n} = \max\limits_{k \in \mathbf{pred}(\mathbf{m})} \max\{FT^l_{k,n}, FT^r_{k,n}\}$, which is only related to the immediate predecessors of task m and does not change with the behavior of task m, it can be regarded as a constant for task m.

In addition, $RT^c_{m,n} = \max\{FT^t_{m,n}, \max\limits_{k \in \mathbf{pred}(\mathbf{m})} FT^c_{k,n}\}$ means that (i) task m starts execution on the cloud after it is completely offloaded onto the cloud or (ii) all immediate predecessors of task m have finished execution. Clearly, for the second case, $RT^c_{m,n}$ does not change with the behavior (such as transmission and execution) of task m. Thus, in this proof, we only consider the first case, and we have

$$RT^c_{m,n} = FT^t_{m,n} = T^{c,trs}_{m,n}(\mathcal{A}) + RT^{trs}_{m,n}, \tag{37}$$

where $RT^{trs}_{m,n}$ denotes the ready time when task m can be transmitted on the wireless sending channel in order to preserve the task-precedence requirement, which is a constant for the offloading of task m.

Therefore, $Z_{m,n}$ can be rewritten as

$$\begin{aligned} Z_{m,n} &= [\gamma^E_{m,n} E^l_{m,n} + \gamma^T_{m,n}(T^l_{m,n} + RT^l_{m,n})] \\ &\quad + a_{m,n}\gamma^T_{m,n}(T^{c,trs}_{m,n}(\mathcal{A}) - T^l_{m,n} + C_{m,n}) \\ &\quad + a_{m,n}\gamma^E_{m,n}(E^{c,trs}_{m,n}(\mathcal{A}) - E^l_{m,n}), \end{aligned} \tag{38}$$

where $C_{m,n} = T^{c,exe}_{m,n} + RT^{trs}_{m,n} - RT^l_{m,n}$ is constant for the behavior of task m.

We can observe from Eqs. (2) and (3) that $T^l_{m,n}$ and $E^l_{m,n}$ are convex w.r.t $f_{m,n}$. It is known that $R_{m,n}(\mathcal{A})$ is concave w.r.t $P^T_{m,n}$ [32], then we have $T^{c,trs}_{m,n}(\mathcal{A})$ is convex function with $P^T_{m,n}$. Furthermore, it is not difficult to verify that $E^{c,trs}_{m,n}(\mathcal{A})$ is also a convex function of $P^T_{m,n}$.

For notational simplicity, we define a vector $\mathbf{x}_{m,n} = [a_{m,n}, f_{m,n}, P^T_{m,n}]$ and use $\mathbf{H}(Z_{m,n}(\mathbf{x}_{m,n}))$ to denote the Hessian matrix of function $Z_{m,n}(\mathbf{x}_{m,n})$. Then we can verify that $\mathbf{H}(Z_{m,n}(\mathbf{x}_{m,n}))$ is a positive semi-definite matrix. In other words, the objective function $Z_{m,n}(\mathbf{x}_{m,n})$ is jointly convex w.r.t. $a_{m,n}$, $f_{m,n}$ and $P^T_{m,n}$.

Now, we verify the convexity of constraints $C1 - C5$. According to Eqs. (9) and (16), constraint $C1$ can be rewritten as

$$\begin{aligned} &\sum_{m=1}^{M}(T^l_{m,n} + RT^l_{m,n}) + a_{m,n}(T^{c,trs}_{m,n}(\mathcal{A}) - T^l_{m,n} + C_{m,n}) \\ &\quad \le T^c_{n,\max}. \end{aligned} \tag{39}$$

Based on the convexity of $T^l_{m,n}$ and $T^{c,trs}_{m,n}(\mathcal{A})$, we can verify that the left term of the inequality Eq. (39) is convex, which means that constraint $C1$ is convex w.r.t $a_{m,n}$, $f_{m,n}$ and $P^T_{m,n}$.

Constraint $C3$ can be rewritten as

$$C3 : T^{c,trs}_{m,n}(\mathcal{A}) + RT^{trs}_{m,n} \le RT^c_{m,n}. \tag{40}$$

Since $T^{c,trs}_{m,n}(\mathcal{A})$ is convex w.r.t $P^T_{m,n}$, and $RT^{trs}_{m,n}$ and $RT^c_{m,n}$ are fixed for the behavior of task m, we can obtain that constraint $C3$ is convex w.r.t $P^T_{m,n}$.

As a result, the optimization problem *OPT-1* is a convex optimization problem w.r.t. the optimization variables $\{a_{m,n}\}$, $\{f_{m,n}\}$ and $\{P^T_{m,n}\}$. That completes the proof.

8 Conclusions

In this paper, we study the problem of energy-efficient dynamic offloading and resource scheduling in mobile cloud computing. We propose a novel distributed eDors algorithm which is composed of the subalgorithms of computation offloading selection, clock frequency control and transmission power allocation. We find that computation offloading selection policy is determined by not only the computing workload of a task, but also the maximum complete time of its immediate predecessors and the clock frequency and transmission power of the mobile device. Furthermore, we observe that the clock frequency of an SMD executing a task depends on the balance between the weight of computation completion time and that of energy consumption, and the price for the required application completion time deadline. Moreover, we derive the optimal transmission power by Newton iteration method. In practice, the transmission power update can be implemented by employing the DVFS technique. Finally, we implement the eDors algorithm in a real testbed and experimental results demonstrate that compared to the existing offloading policies, the eDors algorithm can effectively reduce the energy consumption and application completion time, by taking advantage of the CPU clock frequency control in local computing and the transmission power allocation in cloud computing.

Acknowledgments

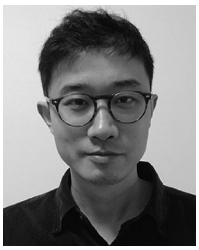
This research work was supported in part by the grant from the US National Science Foundation under grant

number CSR 1513719, and the National Natural Science Foundation of China (61772432, 61772433, 61772446), Fundamental Research Funds for the Central Universities (XDJK2015C010, XDJK2015C019, XDJK2015D023, XDJK2016A011, XDJK2016D047, XDJK201710635069), and Natural Science Key Foundation of Chongqing (cstc2015jcyjBX0094), and by Tianjin Key Laboratory of Advanced Networking (TANK), School of Computer Science and Technology, Tianjin University, Tianjin China, 300350.

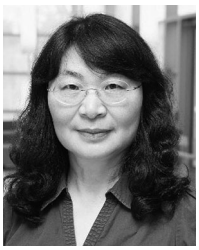
REFERENCES

[1] J. Cohen, "Embedded speech recognition applications in mobile phones: Status, trends, and challenges," in *Proc. IEEE Int. Conf. Acoust.*, 2008, pp. 5352–5355.
[2] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *Proc. Int. Symp. Comput. Commun.*, 2012, pp. 59–66.
[3] C. Ma and Y. Yang, "A battery-aware scheme for routing in wireless ad hoc networks," *IEEE Trans. Veh. Technol.*, vol. 60, no. 8, pp. 3919–3932, Oct. 2011.
[4] C. Ma, Z. Zhang, and Y. Yang, "Battery-aware scheduling in wireless mesh networks," *Mobile Netw. Appl.*, vol. 13, no. 1/2, pp. 228–241, 2008.
[5] C. Ma and Y. Yang, "Battery-aware routing for streaming data transmissions in wireless sensor networks," *Mobile Netw. Appl.*, vol. 11, no. 5, pp. 757–767, 2006.
[6] X. Lin, Y. Wang, Q. Xie, and M. Pedram, "Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment," *IEEE Trans. Serv. Comput.*, vol. 8, no. 2, pp. 175–186, Mar./Apr. 2015.
[7] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2015.
[8] S. Yang, D. Kwon, H. Yi, Y. Cho, Y. Kwon, and Y. Paek, "Techniques to minimize state transfer costs for dynamic execution offloading in mobile cloud computing," *IEEE Trans. Mobile Comput.*, vol. 13, no. 11, pp. 2648–2660, Nov. 2014.
[9] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.
[10] Y. Zhang, D. Niyato, and P. Wang, "Offloading in mobile cloudlet systems with intermittent connectivity," *IEEE Trans. Mobile Comput.*, vol. 14, no. 12, pp. 2516–2529, Dec. 2015.
[11] R. Kaewpuang, D. Niyato, P. Wang, and E. Hossain, "A framework for cooperative resource management in mobile cloud computing," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 12, pp. 2685–2700, Dec. 2013.
[12] S. Barbarossa, S. Sardellitti, and P. D. Lorenzo, "Computation offloading for mobile cloud computing based on wide cross-layer optimization," in *Proc. Future Netw. Mobile Summit*, Jul. 2013, pp. 1–10.
[13] S. E. Mahmoodi, R. N. Uma, and K. P. Subbalakshmi, "Optimal joint scheduling and cloud offloading for mobile applications," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2016.2560808.
[14] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct. 2009.
[15] X. Zhang, S. Jeong, A. Kunjithapatham, and S. Gibbs, "Towards an elastic application model for augmenting computing capabilities of mobile platforms," in *Proc. Int. Conf. Mobilware*, 2010, pp. 161–174.
[16] L. Yang, J. Cao, S. Tang, T. Li, and A. T. S. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 4, pp. 23–32, 2013.
[17] L. Yang, J. Cao, H. Cheng, and Y. Ji, "Multi-user computation partitioning for latency sensitive mobile cloud applications," *IEEE Trans. Comput.*, vol. 64, no. 8, pp. 2253–2266, Aug. 2015.
[18] Y. C. Lee and A. Y. Zomaya, "Minimizing energy consumption for precedence-constrained applications using dynamic voltage scaling," in *Proc. IEEE/ACM Int. Symp. Cluster Comput. Grid*, 2009, pp. 92–99.
[19] K. Kumar and Y. H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Comput.*, vol. 43, no. 4, pp. 51–56, Apr. 2010.
[20] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: Making smartphones last longer with code offload," in *Proc. ACM 8th Int. Conf. Mobile Syst.*, Jun. 2010, pp. 49–62.
[21] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Trans. Wireless Commun.*, vol. 11, no. 6, pp. 1991–1995, Jun. 2012.
[22] A. Goldsmith, *Wireless Communications*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[23] J. M. Cioffi, *Lecture Notes for Advanced Digital Dommunitions*. Stanford, CA, USA: Stanford Unv. Press, 2005. [Online]. Available: http://www.stanford.edu/class/ee379c/
[24] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. USENIX Conf. Hot Top. Cloud Comput.*, Jun. 2010, pp. 1–7.
[25] W. Hu and G. Cao, "Quality-aware traffic offloading in wireless networks," in *Proc. ACM Mobihoc*, 2014, pp. 277–286.
[26] L. Liu, R. Zhang, and K. C. Chua, "Wireless information and power transfer: A dynamic power splitting approach," *IEEE Trans. Commun.*, vol. 61, no. 9, pp. 3990–4001, Sep. 2013.
[27] M. Zhao and Y. Yang, "Optimization-based distributed algorithms for mobile data gathering in wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 11, no. 10, pp. 1464–1477, Oct. 2012.
[28] S. Guo, C. Wang, and Y. Yang, "Joint mobile data gathering and energy provisioning in wireless rechargeable sensor networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 12, pp. 2836–2852, Dec. 2014.
[29] S. Guo and Y. Yang, "A distributed optimal framework for mobile data gathering with concurrent data uploading in wireless sensor networks," in *Proc. IEEE INFOCOM*, 2012, vol. 131, no. 5, pp. 1305–1313.
[30] C. Wang, S. Guo, and Y. Yang, "Energy-efficient mobile data collection in energy-harvesting wireless sensor networks," in *Proc. IEEE Int. Conf. Parallel Distrib. Syst.*, 2014, pp. 55–62.
[31] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle, "Layering as optimization decomposition: A mathematical theory of network architectures," *Proc. IEEE*, vol. 95, no. 1, pp. 255–312, Jan. 2007.
[32] S. Boyd and L. Vandenbergh, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[33] *M. H., Newton Method, Encyclopedia of Mathematics*. New York, NY, USA: Springer, 2001, iSBN 978–1-55608-010-4.
[34] G. Bradski and A. Kaehler, *OpenCV Computer Vision with OpenCV Library*. Newton, MA, USA: OReilly, 2008.
[35] P. Viola and M. J. Jones, "Robust real time face detection," in *Proc. 2nd Int. Workshop Statist. Comput. Theories Vis.*, Jul. 2001, no. 2, pp. 1–25.
[36] M. Turk and A. Pentland, "Eigenfaces for recognition," *J. Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

Songtao Guo received the BS, MS, and PhD degrees in computer software and theory from Chongqing University, Chongqing, China, in 1999, 2003, and 2008, respectively. He was a professor from 2011 to 2012 with Chongqing University. At present, he is a full professor with Southwest University, China. He was a senior research associate with the City University of Hong Kong from 2010 to 2011, and a visiting scholar with Stony Brook University, New York, from May 2011 to May 2012. His research interests include wireless sensor networks, wireless ad hoc networks, and parallel and distributed computing. He has published more than 70 scientific papers in leading refereed journals and conferences. He has received many research grants as a principal investigator from the National Science Foundation of China and Chongqing and the Postdoctoral Science Foundation of China. He is a member of the IEEE.

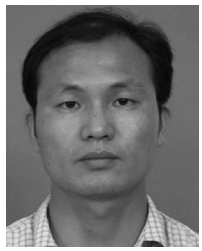


Jiadi Liu received the BS degree in computer science and technology from Jiangnan University, in 2009 and the MS degree in computer application from Jiangnan University, Jiangsu, Wuxi, China, in 2013. He is currently working toward the PhD degree in the College of Electrical Information and Engineering, Southwest University, Chongqing, China. His current research interests include game theory, mobile cloud computing, machine learning, convex optimization theory, and its applications.



Yuanyuan Yang received the BEng and MS degrees in computer science and engineering from Tsinghua University and the MSE and PhD degrees in computer science from Johns Hopkins University. She is a SUNY distinguished professor of computer engineering and computer science and an associate dean for academic affairs with the College of Engineering and Applied Sciences, Stony Brook University, New York. Her research interests include wireless networks, data center networks and cloud computing. She has published more than 400 papers in major journals and refereed conference proceedings and received seven US patents in these areas. She is currently an associate editor-in-chief for the *IEEE Transactions on Cloud Computing* and an associate editor for the *ACM Computing Surveys*. She has served as an associate editor-in-chief and associate editor for the *IEEE Transactions on Computers* and associate editor for the *IEEE Transactions on Parallel and Distributed Systems*. She has also served as a general chair, program chair, or vice chair for several major conferences and a program committee member for numerous conferences. She is a fellow of the IEEE.

Bin Xiao received the BSc and MSc degrees in electronics engineering from Fudan University, China, in 1997 and 2000, respectively, and the PhD degree in computer science from the University of Texas at Dallas, in 2003. Currently, he is an associate professor with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. His research interests include wireless sensor networks and RFID systems. He is an associate editor for the *International Journal of Parallel, Emergent, and Distributed Systems*. He is a recipient of the Best Paper Award of IEEE/IFIP EUC 2011. He is a senior member of the IEEE.



Zhetao Li received the BEng degree in electrical information engineering from Xiangtan University, in 2002, the MEng degree in pattern recognition and intelligent system from Beihang University, in 2005, and the PhD degree in computer application technology from Hunan University, in 2010. He is a professor with the College of Information Engineering, Xiangtan University. From December 2013 to December 2014, he was a post-doc in wireless networks with Stony Brook University. From December 2014 to December 2015, he was an invited professor with Ajou University. His research interests include mobile computing, wireless communication, and signal processing. He has published about 30 papers and submitted about 40 patents on wireless networks and signal processing in the past five years. He was a lead guest editor of two special issues of *PPAN*. He is a member of the IEEE.