Energy-Efficient Dynamic Offloading and Resource Scheduling in Mobile Cloud Computing

Songtao Guo[*], Bin Xiao[†], Yuanyuan Yang[‡] and Yang Yang[*]

[*]College of Electronic and Information Engineering, Southwest University, Chongqing, 400715 China
[†]Department of Computing, The Hong Kong Polytechnic University, Kowloon, Hong Kong
[‡]Department of Electrical & Computer Engineering, Stony Brook University, Stony Brook NY 11794, USA

***Abstract*—Mobile cloud computing (MCC) as an emerging and prospective computing paradigm, can significantly enhance computation capability and save energy of smart mobile devices (SMDs) by offloading computation-intensive tasks from resource-constrained SMDs onto the resource-rich cloud. However, how to achieve energy-efficient computation offloading under the hard constraint for application completion time remains a challenge issue. 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 the energy-efficiency cost (EEC) minimization problem while satisfying the task-dependency requirements and the completion time deadline constraint. To solve the optimization problem, we then propose a distributed eDors algorithm consisting of three subalgorithms of computation offloading selection, clock frequency control and transmission power allocation. More importantly, we find that the 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, our experimental results in a real testbed demonstrate that the eDors algorithm can effectively reduce the EEC by optimally adjusting the CPU clock frequency of SMDs based on the dynamic voltage and frequency scaling (DVFS) technique in local computing, and adapting the transmission power for the wireless channel conditions in cloud computing.**

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

I. Introduction

Recently, smart mobile devices (SMD), e.g., smartphones and tablet-PCs, are gaining enormous popularity due to their portability and compactness. As expected, SMDs are taken as the dominant future computing devices for supporting computation-intensive applications, such as interactive gaming, image/video processing, e-commerce, and online social network services [1], [2]. Such complex applications necessitate higher computing power, memory and battery lifetime on SMDs [3]. Due to the physical size constraint, however, mobile devices are in general resource-constrained. In particular, the limited energy supply from the battery has been one of the most challenging design issues for SMDs [5]–[7].

With the development of wireless communication technology such as 3G, Wi-Fi and 4G, mobile cloud computing (MCC) is envisioned as a promising approach to address such a challenge. The objective of MCC is to extends powerful computing capability of the resource-rich clouds to the resources constrained SMDs so as to augment computing potentials of SMDs. To achieve this objective, MCC needs to migrate resource-intensive computations from SMDs to the cloud via wireless access, referred to as *computation offloading*. A mobile application in MCC needs to be partitioned into a sequence of tasks that can be executed on the mobile device, called *local computing/execution*, or be executed on the cloud, named *cloud computing/execution*. Clearly, MCC can accommodate SMDs to execute complex applications which are impractical to run solely on SMDs due to insufficient SMD resources, such as perception applications [4]. As another advantage, MCC can improve the performance of mobile devices by selectively offloading tasks of an application onto the cloud. Moreover, MCC is beneficial to save energy in mobile devices and prolong operation time.

Although the MCC based on computation offloading technique can significantly enhance computation capability of SMDs, it still remains challenging 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, we should consider the following questions: (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 the tasks in cloud computing?

To answer the above questions, in this paper we focus on the joint computation offloading and resource scheduling problem, in which there are three key issues to be addressed.

- What happens with computation offloading selection when both task-dependency requirements and application completion time constraint are enforced? The enforcement is necessary since there exist in general a certain precedences among the tasks and the application completion time is a hard constraint for latency-sensitive applications.
- Can the energy-efficiency cost (EEC), as defined in *Definition 2*, be minimized by optimally controlling the CPU clock frequency of the mobile device via the DVFS technique [8] in local execution?
- Can the EEC be minimized by optimally allocating the data transmission power for each computation offloading while satisfying the task-precedence requirements 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. Compared to the previous work [8]–[12], this paper has several contributions. *First*, by considering the impact of task precedence on computation completion time, we characterize the EEC by energy consumption and computation completion time in local computing and cloud computing, respectively. *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*, To solve the optimization problem, we propose a distributed eDors algorithm for the policy of computation offloading selection, clock frequency control and transmission power allocation. More importantly, we find 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. *Fourth*, we implement the proposed eDors policy on a real testbed consisting of 20 Android smartphones and a cloud sever and the experimental results show that compared to the existing policy [8], [9], [11], the eDors policy can effectively reduce energy consumption and application completion time.

To the best of our knowledge, this is the first dynamic offloading and resource scheduling work that minimizes energy consumption and application completion time under completion time deadline constraint and task-precedence requirement, by taking into account the CPU clock frequency control in local computing and the transmission power allocation in cloud computing.

II. Related Work

There has been a lot of work on the computation offloading problem in the literature and various offloading policies have been proposed. These policies can be classified into two categories: (i) performance based offloading policies [4], [10], [13]–[16], and (ii) energy based offloading policies [8], [11], [12], [17], [18].

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, the resource-intensive computations are offloaded to the cloud. Chun et al. in [13] proposed a CloneCloud that partitions applications automatically at a fine granularity. In [10], based on the CloneCloud, Yang et al. optimized the overall execution time by dynamically offloading part of Android code running on a smartphone to the cloud. In [4], Satyanarayanan et al. proposed a model that uses a concept of virtual machine that runs on trusted and resource-rich computer, or a cluster of computers named cloudlet. Yang et al. in [14], [15] studied the multi-user computation partitioning so as to optimize the partition of a data stream application such that the application has maximum throughput. However, the above work neither focuses on the energy efficiency minimization problem, nor considers the impact of task dependency on computation offloading policy.

On the other hand, energy based offloading policies aim to reduce energy consumption of mobile devices. This is achieved by reducing the computational overhead of tasks through computation offloading. As a result, computation-intensive tasks are performed in the cloud. In [18], by using Lyapunov optimization, Huang et al. presented a dynamic offloading algorithm to save energy on the mobile device. In [11], Zhang et al. provided an energy-optimal mobile cloud computing framework under stochastic wireless channel. Furthermore, they [12] proposed a collaborative task execution framework for mobile tasks. However, the above work did not consider employing the CPU clock frequency control to reduce energy consumption. Lin et al. [8] proposed a task scheduling algorithm to minimize the total energy consumption of an application. But it did not consider power allocation in the task offloading decision.

To the best of our knowledge, only a few work has addressed the computation offloading problem to minimize both energy consumption and application completion time under the setting of multiple SMDs. In [9], Chen formulated the decentralized computation offloading problem among SMDs as a decentralized game. Compared to our work, however, they did not consider the task precedence and resource scheduling in computation offloading policy.

III. System and Computation Model

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

A. System Model

We assume that there are N smart mobile devices (SMDs) 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 partitioned into a sequence of M tasks, denoted by a set of $\mathcal{M} = \{1, 2, \ldots, M\}$. SMDs offload the computation to the computational 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, the SMDs connect to the access points through Wi-Fi.

In this paper, we employ the dynamic partitioning scheme in [15] 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 as well with our scheme in this paper. We utilize a directed acyclic task graph $G = (V, E)$ to describe the relationship among these tasks, as shown in Fig. 2. 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.

Since the communication and computation models play a key role in mobile cloud computing, we next introduce the communication and computation models in detail.

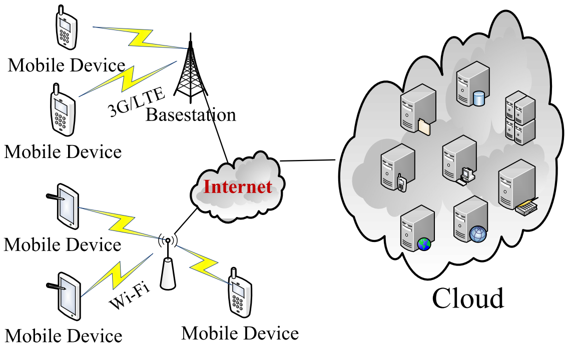


Fig. 1. Illustration of mobile cloud computing with multiple mobile devices.

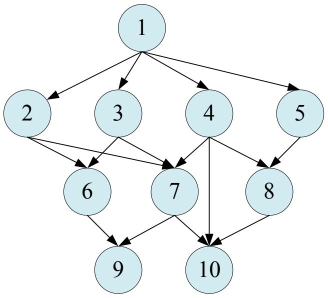


Fig. 2. A simple example of the relationships between tasks.

B. Communication Model

We first introduce the communication model for wireless access. Unless otherwise specified, the wireless access point can be either a WiFi access point, or a basestation in cellular networks. The channel from SMD n to access point s follows quasi-static block fading. 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 the tasks of all the mobile devices by $\mathcal{A} = \{a_{m,n} | m \in \mathcal{M}, n \in \mathcal{N}\}$. For a given decision profile $\mathcal{A}$, we can compute the uplink data rate for computation offloading of task m of mobile device n as

$$R_{m,n}(\mathcal{A}) = W \log_2 \left(1 + \frac{P^T_{m,n} H_{m,n}}{\sigma^2_{m,n} + \sum_{i \neq m, j \neq n, a_{i,j}=1} P^T_{i,j} H_{i,j}}\right) \quad (1)$$

where $P^T_{m,n}$ is the transmission power of SMD n offloading task m to wireless access point s. $H_{m,n}$ denotes the channel gain from SMD n to access point s when transmitting task m due to the path loss and shadowing attenuation, $\sigma^2_{m,n}$ denotes the thermal noise power associated with link (n, s), and W is channel bandwidth.

We can observe from (1) that if many mobile devices offload the computation via wireless access simultaneously, they may lead to severe interference and low data rates.

C. Computation Model

We let $D_{m,n}$ denote the size of computation input data (e.g., the program codes and input parameters) related to computation task m of SMD n. $L_{m,n}$ denotes the 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 the local execution, the wireless task transmission (i.e., the task has been completely offloaded to cloud), the cloud execution and the wireless reception of task k of SMD n, respectively. Next, we discuss the computation overhead in terms of energy consumption and application completion time for local and cloud computing.

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 a SMD. The computation execution time of task m of SMD n by local computing is then given by

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

and the energy consumption of the mobile device is given by

$$E^l_{m,n} = \kappa L_{m,n} f^2_{m,n}, \quad (3)$$

where 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 [19]. It is clear that we can adjust the clock frequency of the CPU chip to achieve the optimum computation time and energy consumption on a mobile device by employing DVFS technique.

Before a task m begins to be executed, all its immediate predecessors must have already been executed. Next, we give the definition of *ready time* of a task [8].

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*

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

*where **pred(m)** denotes the set of immediate predecessors of task m.*

Similar to the existing work [9], [18], 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. Based on this assumption, (4) can be rewritten as

$$RT^l_{m,n} \geq (1 - a_{k,n}) FT^l_{k,n} + a_{k,n} FT^c_{k,n}, k \in \textbf{pred(m)}, \quad (5)$$

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

Clearly, the completion time of the 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.,

$$FT^l_{m,n} = T^l_{m,n} + RT^l_{m,n}, \quad (6)$$

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

Definition 2 (Energy-Efficiency Cost (EEC))**.** *Energy-Efficiency Cost (EEC) 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*

$$Z^l_{m,n} = \gamma^E_{m,n} E^l_{m,n} + \gamma^T_{m,n} FT^l_{m,n}, \tag{7}$$

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.

To meet user-specific demands, we allow different SMDs to choose different wights in the decision making. For example, a device with low battery energy would like to choose a larger $\gamma^E_{m,n}$ in the decision making 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 γ^T_n to reduce the delay.

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

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

$$T^{c,trs}_{m,n}(\mathcal{A}) = D_{m,n}/R_{m,n}(\mathcal{A}), \tag{8}$$

and

$$E^{c,trs}_{m,n}(\mathcal{A}) = P^T_{m,n} T^{c,tr}_{m,n}(\mathcal{A}). \tag{9}$$

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

$$T^{c,exe}_{m,n} = L_{m,n} f_c^{-1}, \tag{10}$$

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.

In this paper, we do not take into account the 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 the offloaded tasks.

Similar to Section III-C1, in the case of considering the dependency among tasks, the ready time of task m on cloud, denoted by $RT^c_{m,n}$, given by

$$RT^c_{m,n} = \max\{FT^t_{m,n}, \max_{k \in \textbf{pred(m)}} FT^c_{k,n}\}. \tag{11}$$

We can observe that if an immediate predecessor task k of task m is executed locally, then $FT^c_{k,n} = 0$. Therefore, $\max_{k \in \textbf{pred(m)}} FT^c_{k,n}$ in (11) 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.,

$$RT^c_{m,n} \geq FT^t_{m,n}, RT^c_{m,n} \geq \max_{k \in \textbf{pred(m)}} FT^c_{k,n}. \tag{12}$$

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

$$FT^c_{m,n} = T^{c,exe}_{m,n} + RT^c_{m,n}, \tag{13}$$

As a result, from (8)-(10), we can give the EEC of computing task m of SMD n on cloud by

$$Z^c_{m,n} = \gamma^T_{m,n}(FT^c_{m,n}) + \gamma^E_{m,n} E^{c,trs}_{m,n}(\mathcal{A}). \tag{14}$$

We can observe from (14) that the low data transmission rate $R_{m,n}$ of mobile device n would result in high energy consumption in the wireless access and long transmission time for offloading the input data to cloud.

IV. Problem Formulation

In this section, we will formulate the eDors problem.

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

$$Z_n = \sum_{m=1}^{M} Z_{m,n} = \sum_{m=1}^{M} (1 - a_{m,n}) Z^l_{m,n} + a_{m,n} Z^c_{m,n} \tag{15}$$

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 the energy efficiency cost is minimized. Therefore, according to constraints (5) and (12), the eDors problem for all SMDs can be formulated as a constrained minimization problem as follows

$$\textbf{OPT-1} \quad \min_{\mathcal{A},\mathcal{F},\mathcal{P}} \sum_{n=1}^{N} Z_n \tag{16}$$

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

$$\begin{aligned}
C1 \;&:\; \sum_{m=1}^{M} (1 - a_{m,n}) FT^l_{m,n} + a_{m,n}(FT^c_{m,n}) \leq T_{n,\max},\\
C2 \;&:\; (1 - a_{k,n}) FT^l_{k,n} + a_{k,n} FT^c_{k,n} \leq RT^l_{m,n}, k \in \textbf{pred(m)},\\
C3 \;&:\; FT^t_{m,n} \leq RT^c_{m,n},\\
C4 \;&:\; \max_{k \in \textbf{pred(m)}} FT^c_{k,n} \leq RT^c_{m,n}, C5 : a_{m,n} \in \{0, 1\},
\end{aligned}$$

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. 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 the 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 (16) 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 [20]–[24], 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: We should first prove that the objective function $Z_{m,n}$ in (16) 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 $C1$-$C5$. Due to space limit, we omit the detailed proof. ■

Theorem 1 reveals that the optimization problem **OPT-1** in (16) 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 (16) derived from its corresponding dual problem.

V. 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.

A. Dual Problem Formulation

The resource allocation policy is derived via solving the dual problem of (16). For this purpose, we first give the Lagrangian function of the primal problem (16) by $L(w, \mu, \mathcal{A}, \mathcal{F}, \mathcal{P})$. 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 $\mu = [\mu_{m,n}, m = 1, ..., M, n = 1, \ldots, N]^T$ 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.

The dual problem for the primal problem (16) is given by

$$\max_{w,\mu} \min_{\mathcal{A},\mathcal{F},\mathcal{P}} L(w, \mu, \mathcal{A}, \mathcal{F}, \mathcal{P}) \tag{17}$$

The dual problem in (17) is decomposed into a hierarchy of two levels. Level 1, the inner minimization in (17), consists of N subproblems with identical structure that can be solved in a distributed manner. Level 2, the outer maximization in (17), is the master problem.

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

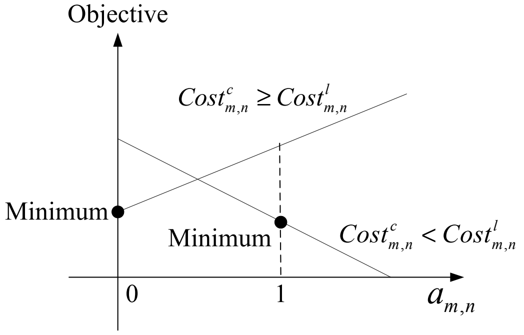


Fig. 3. Illustration of the objective function with the minimum value.

B. 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 requirements being 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

$$\min_{a_{m,n}} Cost^l_{m,n} + a_{m,n}(Cost^c_{m,n} - Cost^l_{m,n}) \tag{18}$$

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

It is clear that the objective function in (18) 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 (18) 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

$$a_{m,n} = \begin{cases} 1, \text{ if } Cost^c_{m,n} < Cost^l_{m,n}, \\ 0, \text{ otherwise.} \end{cases} \tag{19}$$

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.

C. 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

$$\min_{f_{m,n}} Z^l_{m,n} + w_n FT^l_{m,n} + \mu_{m,n}(FT^t_{m,n} - RT^c_{m,n}) \tag{20}$$

subject to constraints $C2$ and $C4$.

Similar to the proof Theorem 1, it is easy to verify that the optimization problem (20) is convex w.r.t $f_{m,n}$. The objective function $F(f_{m,n})$ in (20) can be rewritten as

$$\begin{aligned} F(f_{m,n}) &= \gamma^E_{m,n}\kappa L_{m,n} f^2_{m,n} + \mu_{m,n}(FT^t_{m,n} - RT^c_{m,n}) \quad (21) \\ &\quad + (\gamma^T_{m,n} + w_n)(L_{m,n} f^{-1}_{m,n} + RT^l_{m,n}). \end{aligned}$$

It is known that $RT^l_{m,n}$, $FT^t_{m,n}$ and $RT^l_{m,n}$ are independent of $f_{m,n}$. Using standard convex optimization techniques and the KKT conditions [25], the clock frequency control policy is given by

$$f_{m,n} = \sqrt[3]{\frac{\gamma^T_{m,n} + w_n}{2\kappa\gamma^E_{m,n}}}. \quad (22)$$

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. 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 $FT^l_{k,n}$, $k \in$ **pred(m)**.

D. 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

$$\min_{P^T_{m,n}} \sum_{n=1}^{N}\sum_{m=1}^{M} Z^c_{m,n} + w_n FT^c_{m,n} + \mu_{m,n}(FT^t_{m,n} - RT^c_{m,n}). \quad (23)$$

subject to constraints $C2$ and $C4$.

We let $Y(P^T_{m,n})$ denote the objective function in (23) and then according to Eqs. (8)-(14), we can give $Y(P^T_{m,n})$ of two forms based on two different values of $RT^c_{m,n}$.

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

$$\begin{aligned} Y(P^T_{m,n}) &= \sum_{n=1}^{N}\sum_{m=1}^{M} [(\gamma^T_{m,n} + w_n)(T^{c,exe}_{m,n} + FT^t_{m,n}) \quad (24) \\ &\quad + \gamma^E_{m,n} E^{c,trs}_{m,n}(\mathcal{A})], \end{aligned}$$

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 requirements, which is a constant for task m.

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

$$\begin{aligned} Y(P^T_{m,n}) &= \sum_{n=1}^{N}\sum_{m=1}^{M} [(\gamma^T_{m,n} + w_n)(T^{c,exe}_{m,n} + RT^c_{m,n}) \quad (25) \\ &\quad + \gamma^E_{m,n} E^{c,trs}_{m,n}(\mathcal{A}) + \mu_{m,n}(FT^t_{m,n} - RT^c_{m,n})]. \end{aligned}$$

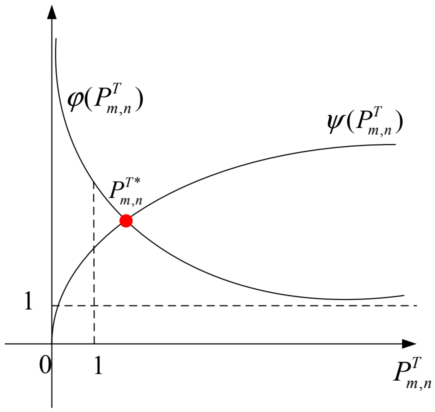


Fig. 4. The intersection point is the solution of Eq. (26).

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 (24) is convex w.r.t $P^T_{m,n}$. Using the KKT conditions [25], the transmission power allocation policy is given by

$$\frac{\alpha_{m,n}}{\varpi_{m,n} + P^T_{m,n}H_{m,n}} + 1 = \ln\left(1 + \frac{P^T_{m,n}H_{m,n}}{\varpi_{m,n}}\right) \quad (26)$$

where $\alpha_{m,n} = (\gamma^T_{m,n} + w_n)H_{m,n}/\gamma^E_{m,n} - \varpi_{m,n}$ and $\varpi_{m,n} = \sigma^2_{m,n} + \sum_{i\neq m, j\neq n, a_{i,j}=1} P^T_{i,j}H_{i,j}$ 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. (26), 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 $\psi(P^T_{m,n}) = \ln\left(1 + \frac{P^T_{m,n}H_{m,n}}{\varpi_{m,n}}\right)$. However, Eq. (26) 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

$$P^T_{m,n}(t+1) = P^T_{m,n}(t) - \frac{\varphi(P^T_{m,n}(t)) - \psi(P^T_{m,n}(t))}{\varphi'(P^T_{m,n}(t) - \psi'(P^T_{m,n}(t))} \quad (27)$$

where$\varphi'(\cdot)$ and $\psi'(\cdot)$ denote the first-order derivative w.r.t. $P^T_{m,n}(t)$. It is shown in [26] 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 $\varphi'(P^T_{m,n}(t) \neq \psi'(P^T_{m,n}(t))$.

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

$$P^T_{m,n}(t+1) = P^T_{m,n}(t) - \frac{\phi(P^T_{m,n}(t)) - \psi(P^T_{m,n}(t))}{\phi'(P^T_{m,n}(t) - \psi'(P^T_{m,n}(t))} \quad (28)$$

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 have the underlying observations from (26) and (28) that 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.

E. Lagrangian Multiplier Update

The Level 2 master problem in (17) can be solved by using the subgradient method. We can update the set of Lagrange multipliers for a given set of $\mathcal{A}, \mathcal{F}, \mathcal{P}$ by

$$\begin{aligned} w_n(k+1) = [w_n(k) + \vartheta(k)(T_{n,\max} \\ - \sum_{m=1}^{M}(1-a_{m,n})FT^{l}_{m,n} + a_{m,n}FT^{c}_{m,n}]^{+}, \end{aligned} \tag{29}$$

$$\mu_{m,n}(k+1) = [\mu_{m,n}(k) + \vartheta(k)(RT^{c}_{m,n} - FT^{t}_{m,n})]^{+}, \tag{30}$$

where index $k > 0$ is the iteration index and $\vartheta(k)$ is positive iteration step size. Then, the updated Lagrange multipliers in (29)-(30) can be used for updating the resource allocation policy in (19), (22), (27) and (28).

The proposed algorithm is described in Algorithm 1. It is not difficult to obtain that the time complexity of the algorithm for SMD n is $O(M * Iter_{\max} * Iter_{power})$, where $Iter_{\max}$ denotes the maximum number of iterations of algorithm for a task, and $Iter_{power}$ indicates the number of iterations for transmission power convergence by Newton iteration method for an iteration of algorithm.

VI. Performance Evaluation

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

A. Experiment Profile

We first consider the mobile cloud computing scenario that $N = 20$ smartphones are randomly scattered over a $100m \times 100m$ 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 [15] to partition a face recognition application in [2] into 100 tasks. Then we implement and test our proposed dynamic offloading policy on a real testbed, where 20 smartphones are SAMSUNG Galaxy S5 with quad-core 2.5 GHz of maximum CPU clock frequency and 2 GB of RAM, and the cloud server consists of 2 IBM X3850X6 servers, each of which has 4 quad-core 3.4 GHz Xeon CPUs and 128 GB of RAM running Ubuntu 14.0. The data size of the computation tasks and the total number of the CPU cycles (i.e., computing load) follow the Gaussian distribution $CN(\mu_1, \sigma_1^2)$ 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. 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.

Algorithm 1 Iterative eDors algorithm for SMD n

```
Require:
  $\mathcal{M}$:a sequence of $M$ tasks of SMD $n$;
  pred(m): the set of immediate predecessors of task $m$;
  $Iter_{\max}$: maximum number of iterations;
  $\epsilon$: 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
  while $t \leq Iter_{\max}$ and $|\mu_{m,n}(t+1) - \mu_{m,n}(t)| > \epsilon$ do
    /* Computation offloading selection */
    Compute $R_{m,n}$, $T^{l}_{m,n}$, $E^{l}_{m,n}$ by (1)-(3), respectively;
    if pred(m) $== \varnothing$ 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 $RT^{trs}_{m,n} = max_{k \in \mathbf{pred(m)}} \{FT^{l}_{k}\}$;
    end if
    Compute $FT^{l}_{m,n}$ by (6) and $Z^{l}_{m,n}$ by (7);
    Calculate $Cost^{l}_{m,n} = Z^{l}_{m,n} + w_n FT^{l}_{m,n}$;
    Compute $T^{c,trs}_{m,n}$, $E^{c,trs}_{m,n}$, $T^{c,exe}_{m,n}$ by (8)-(10), respectively;
    Compute $FT^{t}_{m,n} = T^{c,trs}_{m,n} + RT^{trs}_{m,n}$;
    if pred(m) $== \varnothing$ then
      $RT^{c}_{m,n} = T^{c,trs}_{m,n}$;
    else
      Compute $RT^{c}_{m,n}$ by (11);
    end if
    Compute $FT^{c}_{m,n}$, $Z^{c}_{m,n}$ by (13)-(14), respectively;
    Compute $Cost^{c}_{m,n} = Z^{c}_{m,n} + w_n FT^{c}_{m,n}$;
    if $Cost^{c}_{m,n} < Cost^{l}_{m,n}$ then
      $a_{m,n} = 1$
    else
      $a_{m,n} = 0$
    end if
    if $a_{m,n} == 0$ then
      /* Clock frequency control */
      Compute the clock frequency $f_{m,n}$ by (22);
      $P^{T}_{m,n}(t+1) = P^{T}_{m,n}(t)$;
    else
      /* Transmission power allocation */
      $f_{m,n}(t+1) = f_{m,n}(t)$;
      if $FT^{t}_{m,n} > \max_{k \in \mathbf{pred(m)}} FT^{c}_{k,n}$ then
        Compute $P^{T}_{m,n}$ by (Eq.22) using Newton iteration
          method;
      else
        Compute $P^{T}_{m,n}$ by (Eq.24) using Newton iteration
          method;
      end if
    end if
    /* Lagrangian multiplier update */
    Update Lagrangian multipliers $w_n(t+1)$, $\mu_{m,n}(t+1)$ by
      (29)-(30), respectively;
    $t = t + 1$;
  end while
end for
```

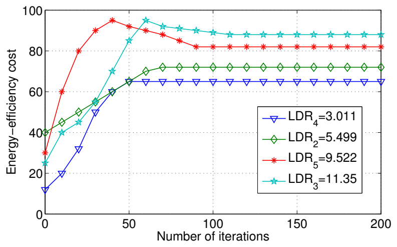
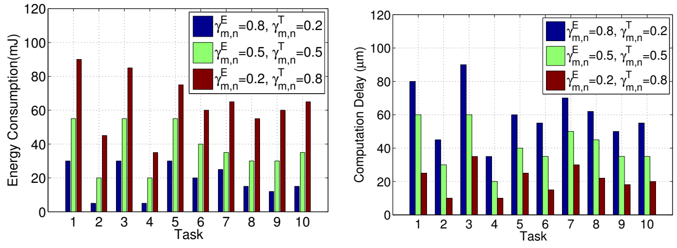


Fig. 5. Dynamics of EECs of tasks with different LDRs.



(a) (b)

Fig. 6. Comparison of energy consumption and computation delay for different $\gamma^E_{m,n}$ and $\gamma^T_{m,n}$. (a) Energy consumption. (b) Computation delay.

B. Convergence and Impact of Task Complexity

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

We plot the convergence of tasks 2-5 by the proposed eDors algorithm and the dynamics of their EECs in Fig. 5. The reason of choosing tasks 2-5 is that they have the same immediate predecessors. We can observe that (i) the proposed eDors algorithm can achieve converge within 100 iterations for all tasks; (ii) the task with larger LDR has the lower speed of convergence, that is, the complexity of task can increase the convergence time. (iii) the task with larger LDR has higher energy efficiency cost. Therefore, in the computation partitioning, the proper LDR setting of tasks plays an imperative role in improving energy efficiency and reducing execution delay.

C. Impact of Weights $\gamma^E_{m,n}$ and $\gamma^T_{m,n}$

In this subsection, we examine the impact of weights, $\gamma^E_{m,n}$ and $\gamma^T_{m,n}$ on the energy consumption and computation delay of tasks with different number of predecessors. Fig. 6 depicts the comparison of energy consumption and computation delay for different settings of $\gamma^E_{m,n}$ and $\gamma^T_{m,n}$. We can observe that for a given task, the energy consumption increase as the $\gamma^E_{m,n}$ decreases, however, the changes of the computation delay are opposite. This is reasonable since a large $\gamma^E_{m,n}$ will lead to the increase of $\alpha_{m,n}$ and $\varphi(P^T_{m,n})$. which in turn causes the decrease of transmission power in cloud execution.

More interestingly, we can find from Fig. 6 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. 6.

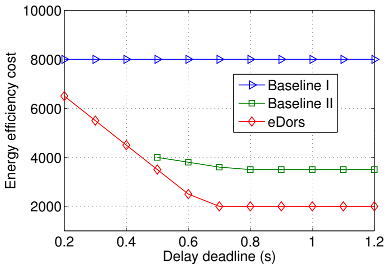


Fig. 7. Comparison of energy efficiency cost on execution strategy.

D. Comparison of Energy Efficiency Cost on Execution Strategy

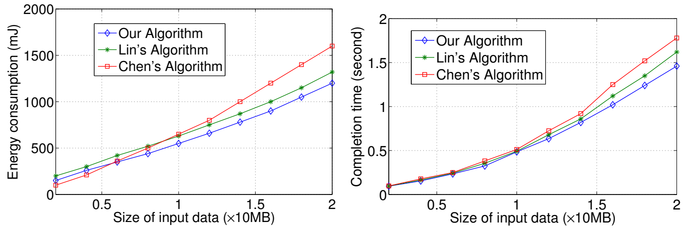
In this subsection, we compare the proposed eDors task execution with two other execution strategies, i.e., local execution and cloud execution [11], under the hard completion time deadline constraint. We implement two baseline algorithms. Baseline I algorithm implements local execution. Baseline II is used to demonstrate cloud execution. Fig. 7 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. 7. 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 when Baseline II becomes applicable. In the case of low completion time deadline, Baseline II for the cloud execution cannot be used. As the completion time deadline becomes longer, Baseline II starts to become active and then its energy efficiency cost decreases slightly. However, the energy efficiency cost of the eDors algorithm is almost two times lower than that of Baseline II algorithm. This is justified since the eDors algorithm adopts the optimal policies of clock frequency control and transmission power allocation.

E. Comparison of Energy Consumption and Completion Time

In this subsection, we compare the energy consumption and application completion time of the eDors algorithm, the task scheduling algorithm in [8] called Lin's Algorithm, and the offloading game algorithm in [9] named Chen's Algorithm for different input data sizes. Fig. 8 depicts the energy consumption and completion time for the three algorithms.

We can observe that when the size of input data is small, Chen's Algorithm has the least energy consumption. However, as the size of input data becomes large, the energy consumption increases rapidly, which is because that Chen's Algorithm does not have the adaptive and control mechanism of energy consumption. Therefore, 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 has always less energy consumption compared with Lin's algorithm, which is justified since our eDors 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



(a) (b)

Fig. 8. Comparison of energy consumption and application completion time for different algorithms. (a) Energy consumption. (b) Application completion time.

control mechanism to reduce energy consumption in cloud computing. In addition, we can also observe from Fig. 8(b) that the application completion time by our eDors algorithm increases slowly when the size of input data is large.

VII. Conclusions

In this paper, we study the problem of energy-efficient dynamic offloading and resource scheduling (eDors) in mobile cloud computing. To the best of our knowledge, this work is the first work of integrating dynamic offloading with resource scheduling so as to achieve the minimization of joint energy consumption and application completion time under completion time deadline constraint and task-precedence requirement. 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 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.

For the future work, we are going to consider the impact of the mobility of SMDs on computation offloading policy. In this case, the mobility patterns might play an important role in the problem formulation. In addition, we will study how to integrate the dynamic runtime computation partitioning with the proposed eDors policy so as to further reduce the EEC.

VIII. Acknowledgement

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 (61170248, 61373179, 61373178, 61402381), Science and Technology Leading Talent Promotion Project of Chongqing (cstc2013kjrcljrccj40001), and Fundamental Research Funds for the Central Universities (XDJK2013A018, XDJK2013C094, 2362014XK12).

References

[1] J. Cohen,"Embedded Speech Recognition Applications in Mobile Phones: Status, Trends, and Challenges," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 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. IEEE Symp. Comput. Commun.*, 2012, pp. 59-66.

[3] M. Shiraz, A. Gani, R. H. Khokhar and R. Buyya, "A Review on Distributed Application Processing Frameworks in Smart Mobile Devices for Mobile Cloud Computing," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 3, pp. 1294-1313, Third Quater 2013.

[4] M. Satyanarayanan, P. Bahl, R. Caceres and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14-23, Oct.-Dec. 2009.

[5] C. Ma and Y. Yang, "A Battery-aware Scheme for Routing in Wireless Ad Hoc Networks," *IEEE Trans. Vehicular Technology*, vol. 60, no. 8, 2011, pp. 3919-3932.

[6] C. Ma, Z. Zhang and Y. Yang, "Battery-aware Scheduling in Wireless Mesh Networks," *Mobile Networks and Applications*, vol. 13, no. 1-2, pp. 228-241, 2008.

[7] C. Ma and Y. Yang, "Battery-aware Routing for Streaming Data Transmissions in Wireless Sensor Networks," *Mobile Networks and Applications*, vol. 11, no. 5, pp. 757-767, 2006.

[8] 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. 2015.

[9] X. Chen, "Decentralized Computation Offloading Game for Mobile Cloud Computing", *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974-983, Apr. 2015.

[10] 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. Mob. Comput.*, vol. 13, no. 11, pp. 2648-2660, Nov. 2014.

[11] W. Zhang et al., "Energy-Optimal Mobile Cloud Computing under Stochastic Wireless Channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569-4581, Sep. 2013.

[12] W. Zhang, Y. Wen and D. O. Wu, "Collaborative Task Execution in Mobile Cloud Computing Under a Stochastic Wireless Channel," *IEEE Trans. Wireless Commun.*, vol. 14, no. 1, pp. 81-93, Sep. 2015.

[13] B. Chun, S. Ihm, P. Maniatis, M. Naik and A. Patti, "Clonecloud: Elastic Execution between Mobile Device and Cloud," in *Proc. ACM Sixth Conf. Comput. Syst.*, April 2011, pp. 301-314.

[14] L. Yang, J. Cao, Y. Yuan, T. Li, A. Han and A. 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, Mar. 2013.

[15] 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.

[16] 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.

[17] 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 Eighth Int. Conf. Mobile Syst.*, Jun. 2010, pp. 49-62.

[18] 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.

[19] A. P. Miettinen and J. K. Nurminen, "Energy Efficiency of Mobile Clients in Cloud Computing," in *Proc. 2010 USENIX Conference on Hot Topics in Cloud Computing*, Boston, USA, June 2010, pp. 1-7.

[20] L. Liu, R. Zhang and K.-C. Chua, "Wireless Information Transfer with Opportunistic Energy Harvesting," *IEEE Trans. Wireless Commun.*, vol. 12, no. 1, pp. 288-300, Jan. 2013.

[21] 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, Sept. 2013.

[22] S. Guo, C. Wang and Y. Yang, "Joint Mobile Data Gathering and Energy Provisioning in Wireless Rechargeable Sensor networks," *IEEE Trans. Mobile Computing*, vol. 13, no. 12, pp. 2836-2852, 2014.

[23] S. Guo and Y. Yang, "A Distributed Optimal Framework for Mobile Data Gathering with Concurrent Data Uploading in Wireless Sensor Networks," *IEEE INFOCOM*, 2012.

[24] C. Wang, S. Guo and Y. Yang, "Energy-Efficient Mobile Data Collection in Energy-Harvesting Wireless Sensor Networks," *2014 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 55-62, Hsinchu, Taiwan, 2014.

[25] S. Boyd and L. Vandenbergh, Convex Optimization. Cambridge Univ. Press, 2004.

[26] H., Michiel, Newton method, Encyclopedia of Mathematics, Springer, 2001, ISBN 978-1-55608-010-4.