2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing

# Online Algorithms for Location-Aware Task Offloading in Two-Tiered Mobile Cloud Environments

Qiufen Xia†, Weifa Liang†, Zichuan Xu†, and Bingbing Zhou‡
† The Australian National University, Canberra, ACT 0200, Australia
‡ The University of Sydney, Sydney, NSW 2006, Australia

***Abstract*—Mobile Cloud Computing (MCC) is emerging as a main ubiquitous computing platform which enables to leverage the resource limitations of mobile devices and wireless networks by offloading data-intensive computation tasks from resource-poor mobile devices to resource-rich clouds. In this paper, we consider an online location-aware offloading problem in a two-tiered mobile cloud computing environment consisting of a local cloudlet and remote clouds, with an objective to fair share the use of the cloudlet by consuming the same proportional of their mobile device energy, while keeping their individual SLA, for which we devise an efficient online algorithm. We also conduct experiments by simulations to evaluate the performance of the proposed algorithm. Experimental results demonstrate that the proposed algorithm is promising and outperforms other heuristics.**

## I. Introduction

With technological advance of wireless portable mobile devices, more and more people nowadays depend heavily on mobile devices including smartphones and tablets for various applications such as Facebook, Twitter, playing games, etc. A report by Juniper Research [10] predicts that the number of applications downloaded globally onto consumers' handsets and tablets will achieve 160 billion in 2017 from 80 billion in 2013. The rapid explosion in demand for rich mobile applications has created the need for new platforms and architectures that can cope with the scalability and QoS needs of ever-growing mobile user populations.

Mobile Cloud Computing (MCC) platforms that aim to overcome resource limitations of mobile devices and wireless networks by offloading computation and data intensive tasks from resource-poor mobile devices to resource-rich clouds, have been demonstrated to meet such demands [2], [4], [11], [20]. In cloud markets, service providers offer reliable and customized services by enabling Service Level Agreements (SLAs) to customers that dictate SLA bounds in terms of speed, bandwidth, and delay. One main concern in ensuring mobile SLA is the level of wireless connectivity offered by last hop access networks such as 3G/4G and Wi-Fi, which usually exhibits varying characteristics. On one hand, 3G/4G connections usually suffer from long delay and slow data transfers [1], [2], which result in more power consumptions of mobile devices and higher service costs of mobile users. On the other hand, Wi-Fi deployments such as 802.11 hotspots enable mobile devices connecting to local cloudlets with low communication latencies. However, the use of local only solutions with Wi-Fi networks will lead to poor system scalability with the growth of number of mobile users. Consequently, latencies and packet losses will increase too. Furthermore, the rich mobile applications often require significant compute and storage resources that most mobile devices cannot meet, due to various constraints imposed on mobile devices such as portable sizes, CPU speeds, storage and batteries. One potential solution for this is to offload compute- and storage-intensive tasks from mobile devices to clouds while keeping the specified SLAs of mobile users met. In addition, mobile users usually move around several places rather than staying at a single location for a whole day. For example, a university student may travel several venues daily in campus such as classrooms, libraries, gyms, and so on. The student needs to leave and join the campus network when he/she is at different locations. The mobility of mobile users will bring new challenges to offload their tasks to clouds. Thus, task offloading must be performed opportunistically; otherwise, offloading a time-stringent task via a heavily-loaded Wi-Fi access point may incur a much longer delay and a heavy energy consumption of its mobile device.

To improve the performance and scalability of mobile applications by dealing with location-aware online task offloading, in this paper we propose a two-tiered MCC architecture that combines the capabilities of the local cloudlet and remote public clouds, where the local cloudlet can be accessed through multiple Wi-Fi wireless access points, and the remote clouds such as Amazon EC2, Microsoft Azure and Google AppEngine can be accessed through cellular networks (3G/4G) by paying the services. Notice that the similar architecture such this one has also been proposed in [16], the main difference between ours and theirs lies in that they did not consider the user access limitation on wireless access points, which cannot be ignored in real settings. Under the proposed architecture, we focus on devising efficient online algorithms for offloading location-aware tasks of mobile users to a local cloudlet or remote clouds such that their energy consumption is fair shared, thereby prolonging the lifetimes of their mobile devices. Our objective is to ensure that each mobile device consumes the same proportion of its energy to its energy capacity while meeting its user SLA requirement.

The main contributions of this paper are as follows. We first propose an energy cost model to model the cost of executing a task in different computing facilities including mobile devices, the local cloudlet, and the remote cloud. We then devise an efficient online algorithm for fairly offloading location-aware tasks among mobile users. We finally conduct experiments by simulations to evaluate the performance of the proposed algorithm. Experimental results demonstrate that the proposed algorithm is very promising. To the best of our knowledge, this is the first time that online location-aware task offloading is considered, and an efficient online algorithm for this problem is proposed.

The remainder of this paper is organized as follows. Section II introduces related work, followed by an introduction of the system model and problem definition in Section III. The proposed algorithm is proposed in Section IV, and the performance evaluation of the proposed algorithm is conducted in Section V, and the conclusion is given in Section VI.

## II. Related work

Offloading computing-intensive applications from energy-constrained mobile devices to powerful clouds is one fundamental problem in mobile cloud computing. Considerable efforts have been taken in recent years [1], [2], [5], [7], [11], [15], [16], [17], [18], [21]. For example, Chun *et al.* [5] proposed a system CloneCloud that aims to partition applications between mobile devices and a local cloudlet in an offline manner by finding an execution point. Specifically, the partitioning mechanism of the CloneCloud runs multiple times under different execution conditions such as network characteristics, CPU speeds and various objectives such as the execution time and energy consumption, and keeps this historic data into a database. At runtime, a partition from the database is picked and offloaded to the cloud for execution. However, it is very difficult to find such an execution point for an online application subject to the SLA requirement. Gelenbe *et al.* [7] dealt with offloading jobs between a local cloud and a remote cloud by incorporating energy consumption and quality of service (QoS) criteria. They considered load balancing between the local cloud and the remote cloud. However, executing tasks on mobile devices sometimes is more energy-saving, since the energy overhead on the wireless communication with the cloud and the energy consumed by network interfaces on mobile devices cannot be neglected [1], [11]. Rahimi *et al.* [15], [16] exploited a tiered framework consisting of a local cloud and a remote public cloud, and modeled mobile applications of a mobile user as a location-time workflow. By assuming that all task locations and time are given, they formulated an optimization problem to determine when and where the tasks should be executed in mobile devices, the local cloudlet or the remote cloud while meeting multidimensional QoS (application delay, device power consumption and user cost/price) constraints, using simulated annealing. Notice that the applications and tasks they considered are offline and they did not consider the capacity limitation of wireless access points in their work. Soyata *et al.* [18] proposed a mobile-cloudlet-cloud architecture for a face recognition application, where the cloudlet receives tasks of users and partitions the tasks between itself and remote clouds aiming to minimize the response time of the application. Cuervo *et al.* [2] considered an application partitioning problem to minimize the energy consumption of mobile devices, by formulating a linear program solution.

These mentioned studies mainly focused on minimizing either the energy consumption of mobile devices [1], [2], [5], [7], [11], or the response time of applications [18]. Most of them assumed that task offloading can be performed immediately without considering the availability and capacity limitation of wireless access points, while others dealt with offline application partitioning without incorporating the mobility of mobile users. In contrast, in this paper we consider online task offloading in more realistic settings. That is, mobile users do not have the knowledge about when and where they have tasks to be offloaded in the future. Tasks from different mobile users may arrive to the system at any time, the system will determine whether each arrived task to be executed on its mobile device itself, or offload the task to a local cloudlet or to a remote cloud, while the SLA requirement of each task is met. We also incorporate the user mobility and admission capacity of each access point into the problem formulation.

To the best of our knowledge, none of existing studies has ever considered the fair use of the cloud or cloudlet resources among mobile users. As some of mobile devices such as laptops are powerful in terms of both energy supply and processing ability, while others such as smart phones are very restricted, they all compete to offload their tasks to the clouds in order to minimize their own energy consumptions. Careless task offloading will result in a biased allocation that favors only few powerful mobile devices while punishing those restricted mobile users. Consequently, the mobile cloud users may switch to other cloud providers. Unlike most existing studies that assumed all tasks from each user must be given prior to task offloading [16], we do not adopt this assumption, instead, each mobile user may not know when and where he/she will have a task in future. In this paper we aim to provide 'equal opportunities' for all mobile users to offload their tasks to the clouds, no matter whether they are powerful or weak. In other words, we aim to maximize the minimum ratio of the energy consumption to the battery capacity of each task-offloading mobile device, and we refer to this criteria as *the offloading fairness* among mobile users.

## III. Preliminaries

In this section, we first introduce the system model, notions and cost metrics. We then elaborate the service level agreement and define the problem precisely.

### A. System model

We consider a two-tiered mobile cloud computing (MCC) environment consisting of a local cloudlet with limited resources, and remote rich-resource public clouds. There is a set of mobile users that can access both clouds wirelessly, where each mobile user either executes a task on his/her mobile device or offload the task to the local cloud through a Wi-Fi access point (AP) or the remote clouds through 3G/4G communication, as illustrated by Fig. 1. Usually, Wi-Fi communication takes much less energy consumption than that of 3G/4G communication. However, if there is only limited bandwidth available at an access point, the energy consumption on Wi-Fi may outweigh the energy consumption on 3G/4G communication. Specifically, let $C_r$ and $C_l$ represent the remote cloud and local cloudlet with service rates $\mu_r$ and $\mu_l$, respectively. Clearly, $\mu_r \geq \mu_l$. The remote cloud $C_r$ owned by cloud service providers has abundant cloud resources for mobile users, and the users have to pay for their services, whereas the local cloudlet $C_l$ managed by a local organization usually has limited resources and its use usually is free of charge.

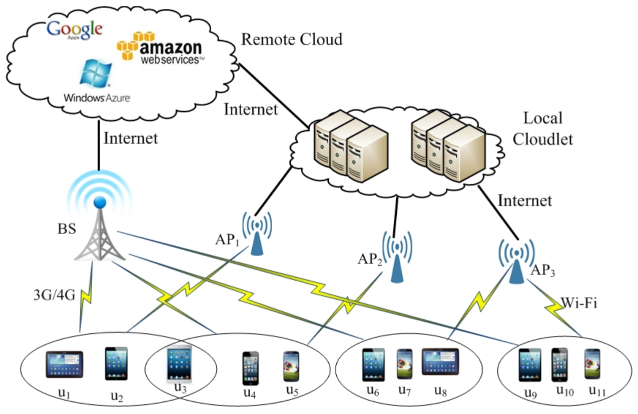


Fig. 1. The architecture of a multi-tiered cloud

Let $\mathcal{AP} = \{AP_1, \ldots, AP_H\}$ be the set of $H$ mobile access points to the local cloudlet. A mobile user can access an AP only if the AP is within the transmission range of the mobile device of the user. Denote by $B_h$ the communication bandwidth capacity of access point $AP_h$ with $1 \leq h \leq H$. As the bandwidth of each AP is fixed, the number of mobile users that can access the local cloudlet simultaneously through the AP must be bounded. In other words, when the number of users using an AP reaches its capacity, the AP will no longer accept any new requests until some of its existing users leave. Assume that the bandwidth allocated to each mobile user for accessing $AP_h$ is equal, which is $\frac{B_h}{n_a}$, where $n_a$ is the maximum number of mobile users that can access an AP at the same time. In case a mobile user is at a location where the user can access multiple APs, the system will determine which AP the user can access, depending on the signal strengths from these APs. As mentioned, each user can also access the remote cloud through local 3G/4G base stations. In this case, we assume that the 3G/4G base stations have sufficient bandwidths to accommodate user access requirements. Let $R_r$ and $R_l$ be the data transmission rates of 3G/4G cellular and Wi-Fi connections, respectively. For the sake of simplicity, we assume that the data transmission/reception rates of all devices to 3G/4G (or Wi-Fi) are identical. If not, the proposed algorithm can be easily modified for this general setting.

## B. *Location-aware tasks*

Let $U = \{\ u_i \ |1 \leq i \leq n\}$ be the set of mobile users. A user can access the local cloudlet from different AP locations, and the user may have different tasks at different locations to offload. When the user has a task to be executed, whether the task is executed in the user's mobile device, or offloaded to the local cloudlet or remote cloud is determined by the system. For example, if a task has a stringent QoS requirement, offloading the task to remote clouds or the local cloudlet may violate its SLA requirement, due to the high workload of its access point at that moment. Instead, the user may execute the task on his/her mobile device. Otherwise, the user should offload the task to the remote cloud or local cloudlet to save energy of the mobile device. We refer to this offloading affected by the locations of mobile users as a *location-aware* application offloading.

We consider a relatively long monitoring period, and time is slotted into equal time slots. The task offloading scheduling proceeds at the beginning of each time slot. Let $s_{i,t}$ be a task of user $u_i$ generated between time slots $t-1$ and $t$ with workload $A_{i,t}$, where the workload of a task can be represented by the number of instructions. Each task $s_{i,t}$ can be represented by a tuple consisting of its workload $A_{i,t}$, arrival time slot $t$, location $l_{i,t}$, and its SLA requirement, i.e., $s_{i,t} = \langle A_{i,t}, t, l_{i,t}, D(s_{i,t})\rangle$. We further assume that at any time slot, a user may or may not have a task request, and the system does not have any knowledge of future task arrivals and arrival rates. The SLA of a task $s_{i,t}$ refers to its delay requirement, i.e., the duration from offloading $s_{i,t}$ to the clouds until receiving its result is no more than a pre-defined time threshold $D(s_{i,t})$. Denoted by $D_r(s_{i,t})$ and $D_l(s_{i,t})$ the average delays occurred by offloading task $s_{i,t}$ to the remote cloud and local cloudlet. Here, we assume that the data size related to $s_{i,t}$ is represented by function $f(s_{i,t})$. For simplicity, we further assume that the duration of receiving the result of $s_{i,t}$ equals the one of uploading $s_{i,t}$. Given workload $A_{i,t}$, the size of data $f(s_{i,t})$ of $s_{i,t}$, transmission rates $R_r$ and $R_l$ assigned to $s_{i,t}$, service rates $\mu_r$ and $\mu_l$ of the remote cloud and local cloudlet, $D_r(s_{i,t})$ and $D_l(s_{i,t})$ are calculated by

$$D_r(s_{i,t}) = 2 \cdot \frac{f(s_{i,t})}{R_r} + \frac{A(s_{i,t})}{\mu_r} \qquad (1)$$

and

$$D_l(s_{i,t}) = 2 \cdot \frac{f(s_{i,t})}{R_l} + \frac{A(s_{i,t})}{\mu_l}. \qquad (2)$$

The SLA requirement of task $s_{i,t}$ thus is $D_r(s_{i,t}) \leq D(s_{i,t})$ if it is executed on the remote cloud; $D_l(s_{i,t}) \leq D(s_{i,t})$ if it is executed on the local cloudlet.

## C. Energy consumption

Let $P_{tr}$ and $P_{idle}$ be the energy consumptions of a mobile device per unit time in transmission and idle modes, respectively, and further let $P_d$ and $\mu_d$ represent the power consumption and service rate of a mobile device when executing tasks by itself. If task $s_{i,t}$ is executed on the remote cloud, the energy consumption $\Delta E_r(t)$ is represented by Eq. (3).

$$\Delta E_r(t) = 2 \cdot \frac{f(s_{i,t})}{R_r} \cdot P_{tr} + \frac{A(s_{i,t})}{\mu_r} \cdot P_{idle}, \tag{3}$$

where for simplicity we assume that the amount of energy consumed for uploading task $s_{i,t}$ equals that for receiving results of $s_{i,t}$. If task $s_{i,t}$ is executed in the local cloudlet, the energy consumption $\Delta E_l(t)$ can be defined as follows.

$$\Delta E_l(t) = 2 \cdot \frac{f(s_{i,t})}{R_l} \cdot P_{tr} + \frac{A(s_{i,t})}{\mu_l} \cdot P_{idle}. \tag{4}$$

If task $s_{i,t}$ is executed on the mobile device, the amount of energy consumed of the mobile device $\Delta E_d(t)$ is

$$\Delta E_d(t) = \frac{A(s_{i,t})}{\mu_d} \cdot P_d, \tag{5}$$

where $\mu_d$ is the service rate of the mobile device and $P_d$ is computing power consumption of the mobile device.

## D. Problem definition

Given the two-tiered mobile cloud computing architecture, a set of mobile users $U = \{u_i \mid 1 \le i \le n\}$, and a monitoring period consisting of $T$ time slots, *the location-aware mobile application offloading problem* is to find a task scheduling at each time slot for mobile users such that mobile devices of offloading tasks can fairly share the use of the cloud resources by consuming the same proportional energy, i.e., the objective is to maximize the minimum ratio of the residual energy to its battery capacity of each mobile device for task offloading, subject to the SLA constraint on each task.

Specifically, given any time slot $t$ within the monitoring period $T$, let $\mathcal{T}(t)$ be the set of arriving tasks between time slots $t-1$ and $t$. Assume that each mobile user has at most one task arrival at each time slot. Let $r_i(t)$ be *the residual energy ratio* of user $u_i$ at time slot $t$, then

$$r_i(t) = \frac{E_{residual}(u_i,t)}{E(u_i)}, \tag{6}$$

where $E_{residual}(u_i,t)$ is the amount of residual energy of mobile device $u_i$ after time slot $t$ and $E(u_i)$ is the battery capacity of $u_i$. The problem thus is to maximize the minimum residual energy ratio among mobile users, i.e., to maximize the value $r_{min} = \{r_i(t) \mid 1 \le i \le n\}$.

## IV. Algorithm

In this section, we first introduce an energy cost model of processing a task in different computing facilities. We then devise an efficient algorithm for the location-aware mobile application offloading problem.

## A. Energy cost model

We model the energy cost of executing a task in different computing facilities as follows.

The energy cost $\zeta_1(s_{i,t})$ of executing task $s_{i,t}$ on the mobile device $u_i$ is determined by not only its battery capacity but also the amount of residual energy it has, i.e., it will be determined by the residual energy ratio $r_i(t)$. This energy cost typically marginally increases with the decrease of the residual energy ratio. The rationale behind is that the less residual energy a mobile device has, the higher probability the device will exhaust its energy, and the high probability the violation of its SLA requirement. That is, a lower residual energy ratio implies a higher energy cost of $s_{i,t}$. We therefore model the *energy cost* of processing task $s_{i,t}$ in a computing facility as an exponential function of its residual energy ratio as follows.

$$\zeta_1(s_{i,t}) = \begin{cases} a^{1-\frac{E_{residual}(u_i,t-1)-\Delta E_d(t)}{E(u_i)}} & \text{if } (E_{residual}(u_i,t-1) \\ & -\Delta E_d(t)) \ge 0, \\ \infty & \text{otherwise} \end{cases} \tag{7}$$

Similarly, the energy costs $\zeta_2(s_{i,t})$ and $\zeta_3(s_{i,t})$ of executing task $s_{i,t}$ in the local cloudlet and remote cloud are defined as follows.

$$\zeta_2(s_{i,t}) = \begin{cases} a^{1-\frac{E_{residual}(u_i,t-1)-\Delta E_l(t)}{E(u_i)}} & \text{if } (E_{residual}(u_i,t-1) \\ & -\Delta E_l(t)) \ge 0, \\ \infty & \text{otherwise} \end{cases} \tag{8}$$

and

$$\zeta_3(s_{i,t}) = \begin{cases} a^{1-\frac{E_{residual}(u_i,t-1)-\Delta E_r(t)}{E(u_i)}} & \text{if } (E_{residual}(u_i,t-1) \\ & -\Delta E_r(t)) \ge 0, \\ \infty & \text{otherwise} \end{cases} \tag{9}$$

where $a > 1$ is a constant which reflects the penalty strength on the use of energy, i.e., the larger the value of $a$, the energy cost of executing a task is much higher. Recall that $E_{residual}(u_i,t-1)$ is the amount of residual energy of mobile device $u_i$ after time slot $t-1$, $E(u_i)$ is the battery capacity of $u_i$, and $\frac{E_{residual}(u_i,t-1)}{E(u_i)}$ is the residual energy ratio of user $u_i$ at time slot $t-1$. From Eqs. (7), (8), and (9), it can be seen that the value of $\zeta_k(s_{i,t})$ is in the range of $[1,a]$ with $1 \le k \le 3$.

## B. Construction of an auxiliary graph

We assume that task processing proceeds at the beginning of each time slot. Let $t$ be the time slot considered. For each task arrived between time slots $t-1$ and $t$, the algorithm needs to determine whether the task should be executed on its own mobile device, or offloaded to the local cloudlet or the remote cloud at time slot $t$ with an aim to maximize the minimum residual energy ratio after $T$ time slots. We approach this problem by reducing it to a weighted maximum matching problem in a bipartite graph $G(t) = (V_1(t), V_2(t), E(t))$ whose construction is as follows.

We assume that there are $n'$ tasks in set $\mathcal{T}(t) = \{\ s_{i,t} \mid 1 \le i \le n'\}$, and we further assume that $n' \le n$. There is a corresponding node in $V_1(t)$ for each task in $\mathcal{T}(t)$, i.e., $V_1(t) = \{x_i \mid s_{i,t} \in \mathcal{T}(t)\}$ and $|V_1(t)| = |\mathcal{T}(t)| = n'$. Each node in $V_2(t)$ corresponds a possible computing facility allocation: the local cloudlet $C_l$, the remote cloud $C_r$, or a mobile device for each task in $V_1(t)$. $V_2(t)$ is constructed as follows. (i) The remote cloud $C_r$ can be accessed by the mobile devices through the base station $BS$. Thus, $V_2(t)$ contains $n'$ virtual base station nodes with each corresponding to a task in $\mathcal{T}(t)$, i.e., $VR(t) = \{bs_i \mid s_{i,t} \in \mathcal{T}(t)\}$; (ii) The local cloudlet $C_l$ can be accessed through each local access point $AP_h \in \mathcal{AP}$ for all $h$ with $1 \leq h \leq H$. Consider an $AP_h$ and let $rb(h,t)$ be the number of occupied channels of $AP_h$ at time slot $t$, then $AP_h$ has $n_a - rb(h,t)$ channels available to users, i.e., there are at most $n_a - rb(h,t)$ users can access $AP_h$ at time slot $t$. Here $n_a$ is the maximum number of mobile users that can access an $AP$ at the same time. Denote by $VL(t) = \{ap_{h,1}, ap_{h,2}, \ldots, ap_{h,n_a - rb(h,t)} \mid AP_h \in \mathcal{AP}\}$ the virtual Wi-Fi access points. If $AP_h$ is within the transmission range of a user $u_i$, then any of nodes $ap_{h,j} \in VL(t)$ derived from $AP_h$ can be the access point of $u_i$ for all $1 \leq i \leq n$ and $1 \leq j \leq n_a - rb(h,t)$. (iii) There are $n'$ device nodes in $V_2(t)$ with each corresponding to one task in $\mathcal{T}(t)$, i.e., $VM(t) = \{md_i \mid s_{i,t} \in \mathcal{T}(t)\}$. Thus, $V_2(t) = VR(t) \cup VL(t) \cup VM(t)$. It is obvious that $|V_2(t)| = |VR(t)| + |VL(t)| + |VM(t)| \leq n' + n_a \cdot H + n' \leq 2n + n_a \cdot H$.

There is an edge in $E(t)$ between each node in $V_1(t)$ corresponding to task $s_{i,t}$ and a node in $V_2(t)$ if $s_{i,t}$ is executed on the node in $V_2(t)$ without violating its SLA. In other words, there is an edge between a node in $V_1(t)$ for $s_{i,t}$ and a virtual node in $V_2(t)$ if the delay calculated by Eq. (1) or Eq. (2) is no greater than its pre-defined threshold $D(s_{i,t})$. The weight associated with each edge is the energy cost defined by Eq. (7), (8), or (9).

We here use an example to illustrate the construction of the auxiliary graph $G(t)$. Assume that there are two access points $AP_1$ and $AP_2$, where $AP_1$ and $AP_2$ have one and two channels available. We further assume that there are 5 mobile users in the system, and there are 4 task arrivals from user 1, user 3, user 4 and user 5 at time slot $t$, respectively. Assume that $AP_1$ and $AP_2$ are within the transmission range of user 1 and user 4, while both $AP_1$ and $AP_2$ are within the transmission ranges of both user 3 and user 5. We further assume that user 5 cannot connect to $AP_1$; otherwise, its SLA requirement will be violated. The auxiliary bipartite graph $G(t)$ derived from the requests at time slot $t$ is shown by Fig. 2.

### C. Algorithm

Having the weighted bipartite graph $G(t)$, we now detail the algorithm for the problem. We need to find a weighted maximum matching in $G(t)$ such that the weighted sum of all matched edges is minimized. Clearly, each node in $V_1(t)$ is incident to a matched edge. For each matched edge $(x_i, y_j)$ with $x_i \in V_1(t)$ and $y_j \in V_2(t)$, if $y_j \in VR(t)$, this implies task $s_{i,t}$ is to be offloaded to the remote cloud; otherwise, if $y_j \in VL(t)$, this implies that task $s_{i,t}$ will be offloaded to the local cloudlet through the AP represented by $y_j$. Otherwise, the task will be executed by itself. The detailed algorithm is described in `Algorithm` 1, which is referred to as `Alg-MBM` for short.

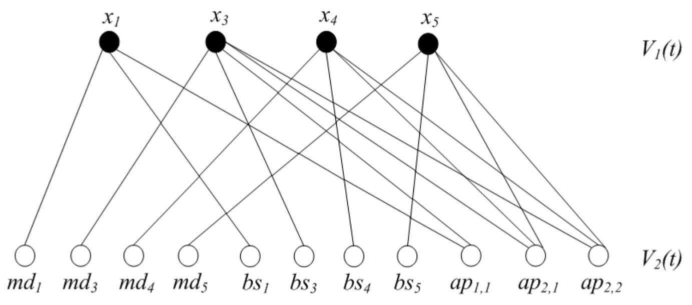


Fig. 2. An example of the construction of graph $G(t)$.

**Algorithm 1** Algorithm for the location-aware mobile application offloading problem

**Input:** A set of mobile users $\{u_i \mid 1 \leq i \leq n\}$, base station $BS$, a set of Wi-Fi access points $\mathcal{AP}$, a remote cloud $C_r$, and a local cloudlet $C_l$.
**Output:** maximize the minimum residual energy ratio among mobile devices at each time slot $t$.

```
1: for each time slot $t$ do
2:     Let $\mathcal{T}(t) = \{s_{i,t} \mid 1 \leq i \leq n'\}$ be the set of tasks arriving between time slot $t-1$ and time slot $t$ and each task is represented by a 3-tuple consisting of its workload, location and arrival time;
3:     Construct the weighted bipartite graph $G = (V_1(t), V_2(t), E(t))$ for the tasks in $\mathcal{T}(t)$;
4:     Finding a weighted maximum matching in $G$ such that the weighted sum of all matched edges is minimized;
5:     Offload tasks to different computing facilities: local cloudlet, remote cloud or mobile devise itself, using the matched edges in the maximum matching;
6:     return the minimum residual energy ratio among the mobile devices.
7: end for
```

*Theorem 1:* Given a two-tiered MCC environment, and a given time period, a set of mobile users to offload their tasks to the clouds, assume that users have no prior knowledge when and where they have tasks to be executed. There is an efficient algorithm for the mobile application offloading problem, which takes $O((n+n_aH)^3) = O((n+n_a)^3)$ time if $H << n$, where $n$ is the maximum number of users at each time slot, $n_a$ is the maximum number of channels of an access point, while $H$ is the number of APs for accessing the local cloudlet.

*Proof:* We first show that the proposed algorithm delivers a feasible solution for all mobile users to fairly share the use of cloud resources by contradiction. Consider two mobile users $A$ and $B$ with residual energy ratios being $r_A(t-1)$ and $r_B(t-1)$ after time slot $t-1$ respectively. Assume that $r_A(t-1) < r_B(t-1)$ and user $A$ has the lowest residual energy ratio among the mobile users after time slot $t-1$. We have $r_A(t) = \frac{E_{residual}(u_A,t-1) - \Delta E_x^A(t)}{E(u_A)} = r_A(t-1) - \frac{\Delta E_x^A(t)}{E(u_A)}$ where $x$ = '$l$', or, $x$ = '$r$', or $x$ = '$d$' depending on which computing facility the task will be executed, and $\Delta E_x^A(t)$ is the amount of energy consumed of mobile device of $A$ at time slot $t$ when the task is executed at computing facility $x$. $r_B(t) = r_B(t-1) - \frac{\Delta E_x^B(t)}{E(u_B)}$ can be defined similarly. We assume that $r_A(t) \leq r_B(t)$ when $x =$'$l$' and both $A$ and $B$ compete for the local cloudlet while the cloudlet can only accommodate one of them at this moment. If $A$ succeeds, then $A$ consumes less energy than that of its execution in its mobile device or in the remote cloud, and user $B$ must offload his/her task to the remote cloud or execute in his/her mobile device. In other words, in terms of the matched pair of $A$ and $B$ in the solution, the edge weight of $A$ is less than that of $B$. Otherwise (assume that $B$ succeeds), this implies that there is another maximum matching in $G(t)$ that leads to a less weighted sum by exchanging the matched pairs of $A$ and $B$ in the current maximum matching, which contradicts that the current maximum matching is a minimum weighted maximum matching.

We then analyze the time complexity of the proposed algorithm. The construction of the auxiliary graph $G(t)$ takes $O(|VR(t)| + |VM(t)| + |V_1(t)| \cdot |VL(t)|) = O(n + n + n \cdot n_a \cdot H) = O(n \cdot n_a \cdot H)$ time as the number of edges in $G$ is bounded by $O(n \cdot n_a \cdot H)$, and the number of nodes is bounded by $O(n + n_a H)$, where $n$ is the maximum number of mobile users at any time slot, and $n_a$ is the maximum number of channels at each access point, while $H$ is the number of APs in the two-tiered mobile cloud computing environment. Finding a weighted maximum matching in $G$ thus takes $O((3n + n_a \cdot H)^3) = O((n + n_a)^3)$ time, using efficient weighted matching algorithms [6] as we assume that $H << n$. ■

## V. Simulation

In this section, we evaluate the performance of the proposed algorithm through empirical simulation. We also investigate the impact of important parameters on the algorithm performance.

### A. Simulation environment

We consider a two-tiered mobile cloud environment consisting of a remote cloud, a local cloudlet, and a set of mobile users as depicted in Fig. 1. The local cloudlet consists of 8 servers, the service rate (CPU capacity) of each server is 2.99 $GHz$. The service rate of the remote cloud is 5.5 $GHz$ [8].The service rates of mobile devices are randomly drawn in the range of $[0.8, 2.2]$ $GHz$, their battery capacities are in the range of $[500, 1,000]$ $joule$.

Following previous studies [3], [16], we consider a 3G wireless network covering a $1,000 \times 1,000$ $m^2$ region. The bandwidth provided by 3G is 75 $Mbps$ [12]. There are 16 wireless access points randomly distributed in the region with transmission radius of 250 $m$ [14]. We adopt IEEE 802.11$g$ [9] as the protocol and standard for Wi-Fi, and thus set channel capacity of each Wi-Fi in the range of [$6Mbps$, $54Mbps$]. The minimum 3G bandwidth reserved to each admitted task is $0.1Mbps$ for maintenance message exchange [19], while the minimum Wi-Fi bandwidth assigned to each admitted task is $1Mbps$ [9] to sustain data transmission. We vary the size of data related to a task in the range of $[1, 5]$ $Mb$ as adopted in [16]. If a mobile user offloads tasks to the local/remote cloud, the assigned bandwidth and power consumptions in different status are generated randomly within ranges listed in Table. I [13].

TABLE I
Bandwidth and power settings of mobile devices

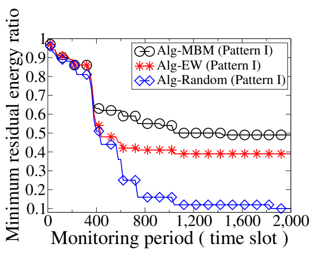
| Type/Status | Wi-Fi | 3G/4G |
|---|---|---|
| Bandwidth capacity | $[6, 54]$ $Mbps$ | 75 $Mbps$ |
| Minimum bandwidth assigned | 1 $Mbps$ | 0.1 $Mbps$ |
| Power consumed on idle | $[0.5, 1]$ $W$ | $[0.6, 1.2]$ $W$ |
| Power consumed on transmitting data | $[1, 1.8]$ $W$ | $[1.5, 2.2]$ $W$ |

In our simulation environment, we adopt a mobility model, in which mobile users travel anywhere without any specific trajectory. We assume that each time slot lasts 5 seconds [17] and the system monitoring period is $T = 2,000$ time slots. For each mobile user, we assume that only a single task is generated at each time slot, while the number of tasks of each mobile user is randomly generated within the range of $[5, 15]$ for the entire monitoring period. Each value in the following figures is the average of the results by applying the nominated algorithm for 20 instances. Unless otherwise specified, we will adopt these default settings in our simulations.

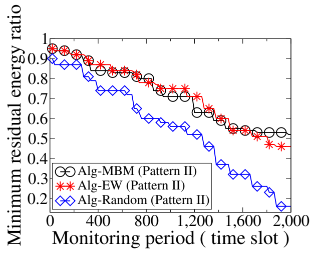
In our experiments the distribution of task arrival mainly follows a Poisson process over the whole monitoring period. In addition, we also consider the other three task patterns: (1) uniform task arrival; (2) tasks arriving mainly during the first half of $T$; (3) tasks arriving mainly during the last half of $T$. For the sake of simplicity, we refer to these four task patterns as *Pattern $I$*, *Pattern $II$*, *Pattern $III$* and *Pattern $IV$*, respectively. We compare the proposed algorithm `Alg-MBM` against two benchmark heuristics. The first heuristic is similar to algorithm `Alg-MBM`, the only difference between them is that this heuristic uses the actual energy consumption of a user as edge weights of the constructed bipartite graph. The second one randomly chooses a computing facility to execute the task while satisfying the SLA requirements of users. For simplicity, we refer to these two benchmark algorithms as `Alg-EW` and `Alg-Random`.

### B. Algorithm performance evaluation

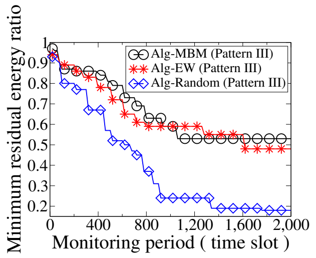
We first evaluate the proposed algorithm `Alg-MBM` against algorithms `Alg-EW` and `Alg-Random` by varying the monitoring period $T$. Fig. 3 plots the performance curves of the mentioned algorithms. From Fig. 3(b), we can see that the minimum residual energy ratio of *Pattern $II$* decreases gradually, due to the uniform arrivals of tasks. It can be seen from Fig. 3(a), Fig. 3(c) and Fig. 3(d) that the minimum residual energy ratio significantly decreases, because of too many task arrivals within these time slots. In addition, it shows from Fig. 3 that algorithm `Alg-MBM` outperforms its counterparts `Alg-EW` and `Alg-Random`. For example, the minimum residual energy ratio of `Alg-MBM` is around 10% higher than that of `Alg-EW` with *Pattern $I$*, and 35% higher than that



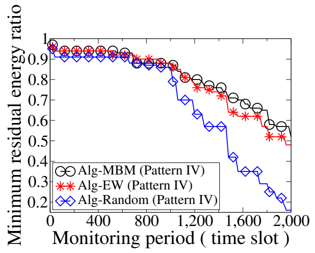
(a) The minimum residual energy ratio of different algorithms with *Pattern I* tasks



(b) The minimum residual energy ratio of different algorithms with *Pattern II* tasks



(c) The minimum residual energy ratio of different algorithms with *Pattern III* tasks



(d) The minimum residual energy ratio of different algorithms with *Pattern IV* tasks

Fig. 3. Performance evaluations of different algorithms with various task patterns over different monitoring periods.

of `Alg-Random` after 600 time slots. The minimum residual energy ratio of `Alg-MBM` is around 6% higher than that of `Alg-EW` under *Pattern II*, *III*, *IV* and 25% higher than that of `Alg-Random`. The rationale behind is that algorithm `Alg-MBM` takes the residual energy of each mobile device seriously into consideration. In contrast, algorithm `Alg-MBM` just focuses on finding a maximum matching with minimum weight (energy consumption) without considering the residual energy of mobile devices, while algorithm `Alg-Random` considers neither the total energy consumption nor the residual energy of devices.

## C. *Impact of important parameters on algorithm performance*

We then study the impact of different parameters on algorithm performance.

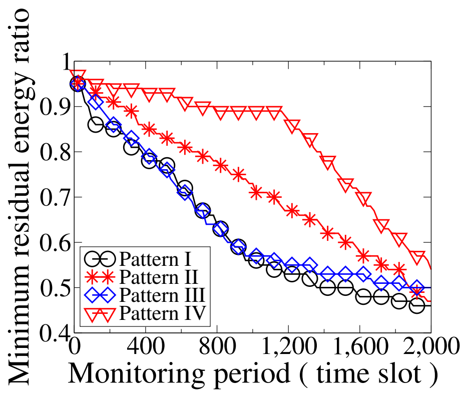


Fig. 4. The minimum residual energy ratios of algorithm `Alg-MBM` with different task patterns over different monitoring periods.

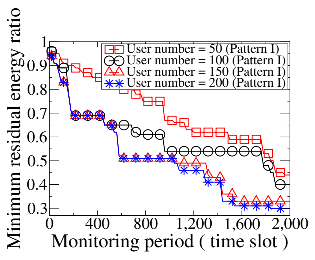
**Impact of task patterns:** We first investigate the impact of different task patterns on the performance of algorithm `Alg-MBM` under the four different task arrival patterns. Fig. 4 depicts that the minimum residual energy ratios under *Pattern I* and *Pattern III* decrease dramatically in the first half of the monitoring period $T$ and decrease slightly in the last half of $T$. In contrast, the minimum residual energy ratio under *Pattern IV* decreases slightly in the first half of $T$ but has a significant decrease in the last half of $T$, while the minimum residual energy ratios under *Pattern II* reduces gradually during $T$. The arguments are similar as we did in Fig. 3.

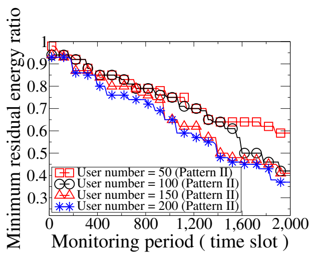
**Impact of the number of mobile users:** We then evaluate the impact of the number of mobile users on the minimum residual energy ratio by varying the number of users from 50 to 200. Fig. 5 plots the performance curves of algorithm `Alg-MBM` under different user numbers and task patterns, from which we can see that the minimum residual energy ratio decreases significantly with the increase of user numbers. The reason is that the admission capacities of Wi-Fi access points are limited, as the number of users increases, tasks of these users tend to be processed by the remote cloud through 3G or the mobile devices themselves rather than by the local cloudlet though Wi-Fi APs, since the remote cloud usually has a long delay and a low data transfer rate, and mobile devices have limited processing abilities, more energy is thus consumed.

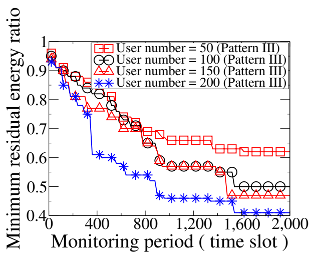
**Impact of parameter $a$:** We finally evaluate the impact of the value of $a$ on the minimum residual energy ratio by varying $a$ from 2 to $2^9$. Fig. 6 plots the performance curves of algorithm `Alg-MBM` with various values of $a$ and task patterns over different monitoring periods, from which we can see that the impact of different values of $a$ on the algorithm performance is insignificant under various task patterns in the early stage of monitoring period $T$. However, it does impact the algorithm performance in the last half of $T$. The rationale behind is that the residual energy of each mobile device is high in the early stage of $T$, in this case, the executing cost of executing a task is approximate to 1 no matter what the value of $a$ is. As time goes, the residual energy of devices drops a lot, the value of $a$ now greatly affects the energy cost which is approximate to $a$ when there is no residual energy left. From Fig. 6, we can see that there is the highest minimum residual energy ratio when $a = 2^7$. We thus set $a = 2^7$ in our default setting.

## VI. Conclusion

In this paper, we considered the location-aware mobile application offloading problem in a multi-tiered mobile cloud computing environment. We formulated a novel online offloading optimization problem and developed an efficient on line algorithms for the problem. We also conducted extensive experiments by simulations to evaluate the performance of


(a) The impact of different numbers of mobile users with `Pattern I` tasks


(b) The impact of different numbers of mobile users with `Pattern II` tasks


(c) The impact of different numbers of mobile users with `Pattern III` tasks

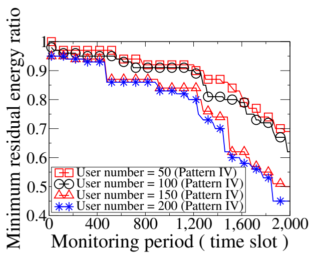
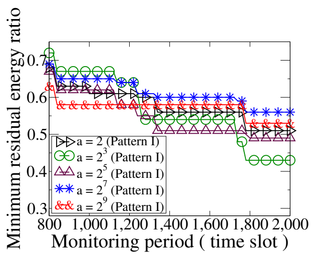
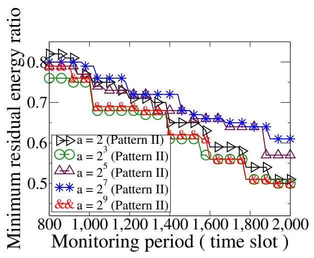
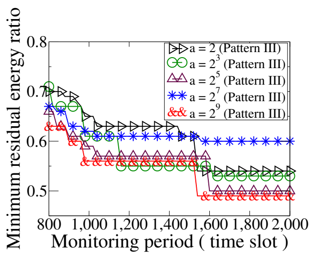

(d) The impact of different numbers of mobile users with `Pattern IV` tasks

Fig. 5. The impact of different numbers of mobile users on the minimum residual energy ratio of `Alg-MBM` with various task patterns over different monitoring periods.


(a) The impact of different values of $a$ with *Pattern I* tasks


(b) The impact of different values of $a$ with *Pattern II* tasks


(c) The impact of different values of $a$ with *Pattern III* tasks

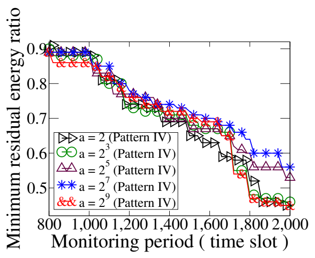

(d) The impact of different values of $a$ with *Pattern IV* tasks

Fig. 6. The impact of values of $a$ on the minimum residual energy ratio of algorithm `Alg-MBM` with various task patterns over different monitoring periods.

the proposed algorithm. Experimental results demonstrate that the proposed algorithm is promising and outperforms other heuristics.

## References

[1] M. V. Barbera, S. Kosta, A. Mei, and J. Stefa. To offload or not to offload? the bandwidth and energy costs of mobile cloud computing. *Proc. of INFOCOM'13*, IEEE, 2013.

[2] E. Cuervo, A. Balasubramanian, D. Cho, A Wolman, S. Saroiu, R. Chandra, and P. Bahl. MAUI: making smartphones last longer with code offload. *Proc. of MobiSys'10*, ACM, 2010.

[3] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Journal of Wireless Communications and Mobile Computing*, Wiley, Vol. 2, No. 5, pp. 483–502, 2002.

[4] S. Clinch, J. Harkes, A. Friday, N. Davies, and M. Satyanarayanan. How close is close enough? understanding the role of cloudlets in supporting display appropriation by mobile users. *Proc. of PerCom'12*, IEEE, 2012.

[5] B. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti. CloneCloud: elastic execution between mobile device and cloud. *Proc. of EuroSys'11*, ACM, 2011.

[6] T. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms.* 3rd Ed., MIT Press, 2009.

[7] E. Gelenbe, R. Lent, and M. Douratsos. Choosing a local or remote cloud. *Symposium on NCCA'12*, IEEE, 2012.

[8] IBM zEC12 (microprocessor). en.wikipedia.org/wiki/IBM_zEC12_(microprocessor)

[9] en.wikipedia.org/wiki/IEEE_802.11.

[10] Press Release: Over 160 Billion Consumer Apps to be Downloaded in 2017, Driven by Free-To-Play Games, Juniper Research Finds. http://www.juniperresearch.com/viewpressrelease.php?pr=383

[11] K. Kumar and Y. Lu. Cloud computing for mobile users: can offloading computation save energy. *Journal of Computer*, Vol. 43, No. 4, pp. 51-56, 2010.

[12] S. Sesia, I. Toufik, and M. Baker. *LTE - the UMTS long term evolution: from theory to practice*. John Wiley, 2011.

[13] G. P. Perrucci, F. H. P. Fitzek, and J. Widmer. Survey on energy consumption entities on the smartphone platform. *Proc. of Vehicular Technology'11*, IEEE, 2011.

[14] A. Raniwala and T. Chiueh. Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network. *Proc. of INFOCOM'05*, IEEE, 2005.

[15] M. R. Rahimi, N. Venkatasubramanian, S. Mehrotra, and A. V. Vasilakos. MAPCloud: mobile applications on an elastic and scalable 2-tier cloud architecture. *Proc. of UCC'12*, IEEE, 2012.

[16] M. R. Rahimi, N. Venkatasubramanian, and A. V. Vasilakos. MuSIC: mobility-aware optimal service allocation in mobile cloud computing. *Proc. of Cloud'13*, IEEE, 2013.

[17] M. Satyanarayanan, R. Bahl, R. Cáceres, and N. Davies. The case for vm-based cloudlets in mobile computing. *Pervasive Computing*, IEEE, Vol. 8, No. 4, pp. 14-23, 2009.

[18] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman. Cloud-Vision: real-time face recognition using a mobile-cloudlet-cloud acceleration architecture. *ISCC'12*, IEEE, 2012.

[19] J. Tang, G. Xue, and W. Zhang. Maximum throughput and fair bandwidth allocation in multi-channel wireless mesh networks. *Proc. of INFOCOM'06*, IEEE, 2006.

[20] T. Verbelen, P. Simoens, F. D. Turck, and B. Dhoedt. Cloudlets: bringing the cloud to the mobile user. *Proc. of MCS'12*, ACM, 2012.

[21] Q. Xia, W. Liang, and W. Xu. Throughput maximization for online request admissions in mobile cloudlets. *Proc. of LCN'13*, IEEE, 2013.