TRANSACTIONS ON EMERGING TELECOMMUNICATIONS TECHNOLOGIES
Trans. Emerging Tel. Tech. (2015)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/ett.2967

RESEARCH ARTICLE

AGILE: A terminal energy efficient scheduling method in mobile cloud computing

Chao Chen[1]*, Weidong Bao[1], Xiaomin Zhu[1], Haoran Ji[1], Wenhua Xiao[1] and Jianhong Wu[2]

[1] School of Information System and Management, National University of Defense Technology, Changsha, China
[2] Department of Mathematics and Statistics, York University, Toronto, Canada

ABSTRACT

With the development of mobile telecommunication technology, mobile phones have become a necessary tool in daily life and provided us many conveniences. Meanwhile, the huge number of cell phones constitute a potential high performance data processing system, called mobile cloud computing, to strengthen capacity for individual devices. Many researchers have studied about the architectures and scheduling algorithms of mobile cloud computing. However, little work has been performed about how to schedule mobile application tasks in data centers to extend battery life for mobile terminals. To address this issue, we investigate agent models, mobile energy consumption models and data transmission models under different connection environments. Based on which, we propose a novel terminal energy efficient scheduling method (AGILE for short). AGILE compares energy consumption in cloud execution and mobile execution according to the actual wireless environment, then makes energy-efficient decisions. Extensive experiments are conducted to evaluate the performance of the AGILE under different wireless channels, and the performance impact on different parameters are studied. The experimental results indicate that the proposed method can save mobile devices' energy effectively. Copyright © 2015 John Wiley & Sons, Ltd.

***Correspondence**
C. Chen, School of Information System and Management, National University of Defense Technology, Changsha 410073, China.
E-mail: chenchao@nudt.edu.cn

Received 1 April 2015; Revised 8 June 2015; Accepted 24 June 2015

1. INTRODUCTION

Nowadays, mobile communicating and processing devices are more and more common in our daily life and provide us many conveniences. Examples include the Google Android Phone [1], Apple iPhone [2] and netbooks provided by several other manufacturers. Tablets like Apple iPad and Galaxy Note are also used wildly. However, the resource constraints such as battery capacity limitation seriously affect the users' experience. On the other hand, cloud computing has been explosively progressing these years. A Berkeley report [3] revealed:'cloud computing, the long-held dream of computing as a utility, has the potential to transform a large part of the IT industry, making software even more attractive as a service'. With cloud computing, the users need not purchase expensive equipments any more, and they can use the cloud services based on the 'pay as you go' model. The most significant advantage of cloud computing is that it can help end-users offload the heavy computation workload and thus break through the resource limitation of their devices. This advantage makes cloud computing be an effective solution to mobile devices performance constraints. In return, the mobile devices can extend the use of cloud computing owing to its mobility and convenience. As a result, a new research field is emerging, that is, mobile cloud computing (MCC).

Mobile cloud computing is the integration of cloud computing with mobile devices. From the study by Hoang T. Dinh *et al.* [4], the definition of MCC can be described as 'Mobile Cloud Computing at its simplest, refers to an infrastructure where both the data storage and the data processing happen outside of the mobile device. Mobile cloud applications move the computing power and data storage away from mobile phones and into the cloud, bringing applications and mobile computing to not just smartphone users but a much broader range of mobile subscribers'.

Mobile cloud computing has brought us many conveniences. For example, processing heavy computation works without considering the mobile devices's resource constraints, accomplishing jobs through our mobile phones wherever we are and whenever the time is. However, there are also many challenges coming with this new technology. Data privacy and security, the trade-offs between execution in mobile devices and execution in data centers, the selection among the available cloud service providers and the performance under different wireless environments. However, for the battery powered mobile devices, the energy consumption optimization is a negligible aspect among these challenges. And this paper concerns energy for mobile devices most.

Up to now, many scholars have concentrated on energy costs of mobile cloud computing. Many researchers consider cloud computing as a substitute of mobile device itself for the application execution. And a great deal of work focuses on the energy consumption problem for the mobile devices in MCC. In [5–7], computation offloading to cloud to save energy was discussed. However, they seldom take into account the differences among various wireless channels. In [8], the real case study about daily use of mobile devices has been studied, but the continuous backup in cloud will bring users endless bills. Additionally, to the best of our knowledge, seldom, work considered the heterogeneous mobile devices when proposing offloading strategies.

Our goal in this paper is to propose an effective scheduling algorithm aiming at energy consumption optimization for mobile devices in real-life scenarios with the consideration of systems errors, network status, device processing ability, cloud service availability and so on. To deal with the problem about differences among heterogeneous mobiles, we employ the agent-based technology for the environment.

Agent-based technology is derived from distributed artificial intelligence (DAI). According to the work by Wooldridge and Jannings [9], an agent is 'a self-contained program capable of controlling its own decision-making and acting, based on its perception of its environment, in pursuit of one or more objectives'. The agent-based technology shows advantages in heterogeneous environment as it can run independently on the devices, and they can provide the necessary data information in a predefined form.

The major contributions of this work include the following:

- We applied agent-based technologies into MCC, hiding the mobile devices's physical differences, supporting communication and coordination, in a particular information format, among the devices and cloud servers.
- We gave a detailed analysis on the factors relating to the energy consumption for mobile devices, and constructed energy consumption models for mobile devices.
- We studied data transmission under different connection environments and constructed transmission energy models to study energy optimization issues.
- We proposed a terminal energy efficient scheduling method or AGILE to make decisions about mobile application tasks executed by mobile devices or cloud servers.
- We conducted experimental evaluation of the proposed algorithm by simulations based on CloudSim platform.

The rest of this paper is organised as follows. Section 2 summarises related work in literature. Section 3 gives an elaborate introduction about the energy model and designing of algorithm. The simulation approaches and analysis about the experiments are described in Section 4. Section 5 concludes the paper with a summary and future directions.

2. RELATED WORK

In recent years, many researchers have concentrate on the issues to extend the battery duration time.

In [10], the authors summarised four basic approaches from previous literature works for saving energy and extending battery lifetime in mobile devices:

- *Avoid energy waste*. The mobile devices can convert to a less energy consumption mode when the work load is not so heavy. Besides, we can turn off some switches not in need and kill processes not necessary.
- *Slow the CPU speed*. When the application is not so time sensitive, we can slow down the CPU speed by setting the clock speed by half, and in this situation, the execution time doubles, but only one quarter of the energy is consumed.
- *Apply new generation components*. The technological advances in CPU, memory, sensors never stops, which helps us to extend the battery life of our mobile devices. For example, in [11], the new generation CPU is much energy saving.
- *Schedule computation to cloud*. The mobile system does not process the computation work; and the computation is performed somewhere in the cloud computing environment. Under this condition, the mobile devices only transmit data needed, thus extending the mobile systems battery lifetime.

In this work, we mainly concentrate on the last approach, by scheduling task more properly to save energy. In order to achieve energy saving, the factors and energy models should be discussed first. And many researchers have made contributions to the problem.

Before going to design the scheduling algorithm of conserving energy for mobile devices, an energy model for it is needed. In previous works, many models have been proposed. In [10, 12–15], energy models have been constructed, and the per cent of energy saving has been measured. The paper [12] compared the energy consumption of three applications (a face recognition application, a chess game and a video game) and concluded that energy savings from 27% up to 80% can be achieved when using the proposed mobile code offloading system based on their energy model. Balasubramanian *et al.* [13] considered the tail energy overhead, and they achieved energy conservation by about 30–50% measured by their energy models.

In [10], the authors considered application instructions, CPU speed, cloud server execution speed, data transmission and so on and constructed energy consumption models for the mobile devices and cloud servers. They discussed the energy saving for different situations and gave examples to illustrate it. In addition, data encryption was taken into consideration when determining the energy saving. However, experiments and further discussion about the energy models are absent.

Xian *et al.* [14] considered the problem of extending the battery duration time for mobile devices by offloading computation to servers. They hold the idea that the estimation of computation were not reliable and proposed an approach to executing program initially on the mobile devices with a timeout. When the computation is not finished after the timeout, it will be offloaded to servers. They constructed energy models for local execution and server execution, and energy ratio formula was proposed to indicate timeout method types. But the authors did not consider the encryption problem, which is significant in guaranteeing data privacy and security in data transmission. Besides, the mobile device status and network availability were not taken into account.

Zhang *et al.* [15] investigated the problem on how to conserve energy for the resource-constrained mobile device, by optimally executing mobile applications in either the mobile device or the cloud clone. They proposed models for the mobile applications, mobile execution energy consumption, cloud execution energy consumption and optimal application execution policy decision. They considered a stochastic wireless channel rather than the deterministic channel, which is more in line with the actual situation. Nevertheless, Dynamic Voltage Scaling(DVS) may not be able to function as most of mobile devices are not equipped with DVS CPUs. The stochastic wireless channel is a highlight of the paper, but different wireless environments were not paid attention to, such as 2G, 3G, 4G and WIFI. If stochastic wireless channel in different connection environments is considered, that will be more consistent with the actual cases.

The aforementioned energy consumption models have their limitations, and the factors leading to energy consumption should be detailedly discussed for a more practical energy model. Additionally, the user operation habits should also be considered to be consistent with the practical situation. Apart from the energy models, many scholars have devoted to scheduling or partitioning algorithms of the application tasks, trying to decide whether the tasks should be executed locally or in a remote server. In work [16], the authors presented a methodical study that discusses whether tasks can be offloaded to conserve energy.

Some algorithms are based on the previous statistical data to make decisions. The work in [17] discussed on the main factors that affect the energy consumption of mobile devices. An offloading algorithm was presented to determine that the application execute locally or in a remote cloud server. They concluded that trade-offs are highly sensitive to the exact characteristics of the workload, data communication patterns and technologies used. Ohtman *et al.* [18] proposed dynamic load sharing algorithms that learn and adapt its decision based on previous CPU time measurements. Their experiments showed that the benefits of job migration depend on several factors, that is, available bandwidth, CPU utilisation and processor cycle of the mobile relative to the fixed host. And the conclusion of their work is that the benefit of job migration is influenced by CPU utilisation, and a mobile host with high CPU utilisation is more likely to benefit from task offloading.

Many researchers concentrate on offloading of applications in a more complex environment, that is, social related applications. In [19], the researchers proposed SociableSense, a sensing platform that implements a novel adaptive sampling scheme based on learning methods and a dynamic computation distribution mechanism based on decision theory. The system is also able to, in the eye of saving energy, decide whether to perform computation of tasks locally or remotely. Conducted experiments showed that the adaptive sampling and computation distribution schemes balance trade-offs among accuracy, energy, latency and data traffic. Barbera *et al.* [20] proposed the use of opportunistic delegation as a data traffic offload solution to the recent boost up of mobile data consumption in metropolitan areas. Their solution relies on the upgrade of a small, crucial set of VIP nodes that regularly visit all network users and collect (disseminate) data to them on behalf of the network infrastructure. Extensive experiments with several real and synthetic data sets show the proposed methods' effectiveness in offloading, a very small part of network nodes can guarantee most portion of the network offload.

Another solution for the offloading is CloneCloud, which means each real device is associated to a software clone in the cloud. In [21], clones of mobile devices are created in the cloud, and some part of the application would be offloaded to the mobile device clones running in the cloud. The paper by Chun *et al.* [22] presented the design and implementation of CloneCloud, a system that automatically transforms mobile applications to benefit from the cloud. The system overcame design and implementation challenges to achieve basic augmented execution of mobile applications on the cloud. They combined offloading, migration with merging and on-demand instantiation of offloading to address those challenges.

Other kind of solutions has also been studied. The paper by Chen *et al.* [23] discussed offloading algorithms based on java-enabled wireless devices, their work emphasises that the choice of compilation/execution strategy for the machine-neutral Java bytecodes critically impacts the energy consumed by the device. Similarly, Hung *et al.* [24] studied offloading scheme based on android systems, they extend the programming model of their previous work on Android smartphones with a flow-based programming paradigm and created a more flexible way for application offloading. In the work of [25], a linear optimization problem was formulated to minimise the power consumption of mobile devices by offloading. And an algorithm based on Lyapunov optimization was proposed in paper [26] to save energy.

Compared with the previous efforts, this work has several differences. First, we employed agent technologies to shield the difference between mobile devices and between mobile application tasks to support us research from an abstract level. Second, we provided the theoretical framework of mobile execution and cloud execution in order to conserve energy consumption, and we considered the soft error rate, data transmission error rate and encryption rate to simulate the real systems. Third, we proposed a mobile application offloading algorithm from the perspective of energy saving. Finally, we took different connection environments into consideration and studied the energy saving issue under different wireless channels.

3. MODELLING AND ALGORITHM DESIGNING

Modelling about the mobile cloud computing offloading to save energy is proposed in this section. We constructed models for mobile devices, remote cloud virtual machines and mobile application tasks at an abstract level. And the energy consumption models of mobile execution and remote execution for mobile devices have been presented. Following that, the offloading algorithm for mobile application tasks is introduced. As for the heterogeneous mobile devices and cloud virtual machines, agent-based technology is used to support communication among mobile terminals and cloud servers in a specified format.

3.1. Agent model

As is known to all, there are so many brands of mobile terminals. Each device has their own format of information. For example, connection status, power status and so on. The difference between devices makes it hard for us to apply task allocation algorithms in mobile terminal, as the algorithms are based on the understanding of local state information. Thanks to the agent technology, it can provide information we need in a predefined format. That is because we can design agent application regardless of mobile platforms differences (e.g. android OS [1], iOS [2], windows phone OS [27]).

In this work, two kinds of agents are designed, that is, mobile device agent and cloud server agent. Each of them

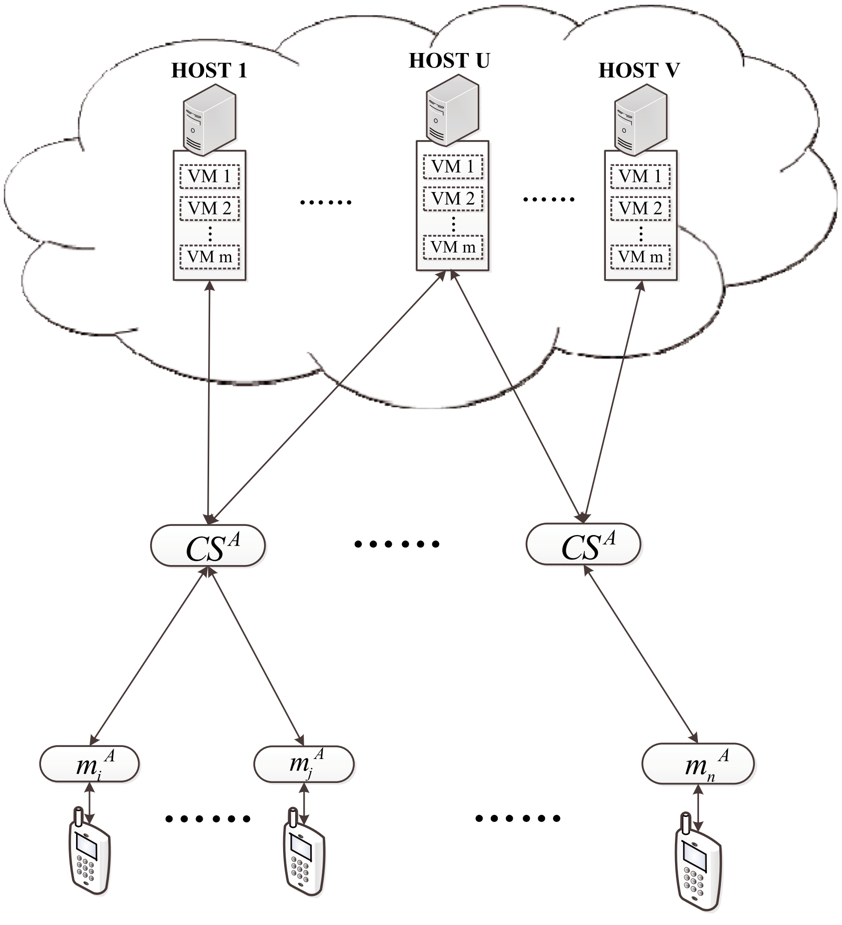


Figure 1. Interaction relationships of agents.

works according to their own rules, more importantly, the mobile device agents can acquire status information of the mobile terminals and the cloud virtual machines information to accomplish task offloading. They are denoted by the following symbols:

- $M^A = \left\{ m_j^A, j = 1, 2, 3, \ldots \right\}$ is the mobile agent set, m_j^A represents the jth mobile device agents, each agent of this category collects the status information of local device and obtains virtual machine information from cloud server agent.
- CS^A is the cloud server agent, it can gather status information about the virtual machines and send to the specified mobile device agent when there is a request.

The interaction relationships of two kinds of agents are depicted in Figure 1.

3.2. Mobile application task model

When a model depicts all aspects of mobile application task and considers many details, it will be complicated and can be narrowly used because its high level of details may not fit other application tasks. Besides, the reusability of the model is absent, which is not consistent with the idea of engineering practice. Consequently, we propose a model for mobile application task from an abstract level, capturing the essential characteristics of a typical application task.

In this work, the mobile application task is abstracted with two parameters, including the following:

- Input data size *Data_Size*: the bytes of input data for an application task, and it mainly affects the transmission of the task.
- Task length *Task_Length*: the number of instructions to be executed, and its completion depends on the mobile CPU speed or virtual machine processing speed.

It should be noted that the two parameters show essential effects on energy consumption of a task. The input data size *Data_Size* mainly influences the transmission energy, the task length *Task_Length* has impact on CPU processing energy consumption. Generally, tasks with larger input data size and shorter task length are more likely to be run locally, because the transmission energy for them may be larger than execution energy. Consequently, we modelled the mobile application task as *T(Data_Size,Task_Length)*. And more details will be discussed in later sections.

3.3. Mobile execution energy model

In the mobile environment, the CPU energy consumption is much larger than that in memory and screen. The mobile execution mainly impacts the CPU workload, which determines the energy consumption. When a mobile application task is executed locally, the energy consumption caused by the components except CPU is lightly influenced, that is, because users' operation habits will not change with the local execution of the task. Consequently, we mainly consider the extra processing energy of executing the application tasks on mobile devices.

The CPU workload increment caused by a mobile application task is determined by its task length, which is measured by the number of instructions to be executed, denoted as *Task_Length*. In the executing process, the instructions error usually exists. We use the symbol α to represent the instruction error rate of a task execution. And in our simulation, α equals to a random value between 10% and 30%. As a result, the number of total instructions of a task execution can be expressed as

$$Total\_Instruction = (1 + \alpha) \times Task\_Length \qquad (1)$$

In this work, we use *EpI* to represent the energy consumption for a single instruction. Therefore, when the application task is executed in the mobile device, the energy consumption can be calculated as

$$E\_Local = EpI \times Total\_Instruction \qquad (2)$$

3.4. Data communication model

Before the model construction, we make some assumptions for the data communication environment.

- The 4G is under popularization, and infrastructures are under construction today, so we only consider the 2G, 3G and WIFI channel.
- The cloud service providers does have applications in their cloud server initially, thus the cloud execution of tasks do not incur additional energy cost except application task data uploading from the perspective of mobile devices.
- The connection environment keeps the same when doing a task offloading process, and we use an average uploading speed to distinct different wireless channels.
- The power of receiving data on a mobile device is often smaller than the uploading power, so we simplify the formulation by setting the receiving energy consumption to be a constant value, as is studied in [17].

As such, scheduling of the output from the cloud will not be considered, and we focus on the task data uploading.

In the Internet environment, security issue cannot be ignored. The task data to uploading should be encrypted to ensure security, which brings us extra data expense. We denote the extra encryption data rate by using the symbol β. Another problem of task uploading is the retransmission rate, which is caused by the transmission error and significantly influenced by connection status. Besides, γ is used

Table I. Transmission speed of 2G, 3G and WIFI.

Connections	Max speed	typical speed interval
2G	300 kbps	(40 kbps , 240 kbps)
3G	42 mbps	(120 kbps , 2 mbps)
WIFI		(800 kbps , 8 mbps)

to represent the retransmission rate. Usually, the value of γ is inversely proportional to the wireless connection conditions. That is to say, the better the connection status is, the smaller the error rate will be. Consequently, the total transmission data size for a task is given by

$$Total\_Transmissiondata = \frac{1+\beta}{1-\gamma} \times Data\_Size \qquad (3)$$

Another problem of the cloud execution is run time, which directly influences the energy consumption. The time expense for the cloud execution of a task consists of three components: uploading time, cloud execution time and result download time. The most typical characteristic of cloud computing is that it can provide users endless computation ability through pooled resources. Besides, the tasks from mobile applications usually are slight from the perspective of cloud virtual machines. So, we ignore the processing time in cloud. As for the result download time, we set it to be a constant value, represented by *Receiving_Time*, for the consideration of a constant energy consumption assumed before. As assumed, we use average transmission speed, denoted by *Ave_Speed*, to measure the connection environment during the task uploading.

It should be noted that the values of *Ave_Speed* is set according to the wireless environment. Referring to [28, 29], we conclude transmission speed of different wireless connections in Table I. Under WIFI environment, there does not exist firm maximum speed limitation because the max speed is determined by the wired network supporting it.

3.5. Cloud execution energy model

When a mobile application task is executed in cloud server, there are three parts of power consumption, that is, uploading energy, cloud server energy consumption and result receiving energy. In this research, we mainly focus on the task offloading algorithm to extend the battery life for mobile devices. So, we only consider the energy consumption of mobile device and ignore the cloud server energy consumption. As for the consumed power for receiving result, we have assumed it to be a constant value in modelling of data communication. As a result, our focus lies on the uploading energy consumption of mobile devices.

As stated in [13], the authors did a measurement study to acquire energy models for power consumption over 2G, 3G and WIFI networks. The model considers both the data size and the uploading time to determine the energy consumption. In this work, we predict the energy consumption of uploading application tasks based on their models.

Table II. Energy model for uploading x bytes of data over 2G, 3G and WIFI networks.

	2G	3G	WIFI
RT(x)	0.036(x)+1.7	0.025(x)+3.5	0.007(x)+5.9
TE	0.25 J/s	0.62 J/s	NA
EFM	0.03 J/s	0.02 J/s	0.05 J/s
T_T	6 s	12 s	NA

The energy consumption model of 2G, 3G and WIFI networks is illustrated in Table II [13]. The consuming power for an application task uploading over 2G or 3G networks consists of three components: ramp energy (energy required to switch to the high-power state), transmission energy and tail energy (energy spent in high-power state after the completion of the transfer). We use *RT(Data_Size)* to represent the sum of ramp energy and transmission energy of a task *T(Data_Size,Task_Length)*, and *TE* denotes the tail energy. For WIFI environment, tail energy does not exist, and *RT(Data_Size)* is used to denote the sum of transmission energy and the scanning and association energy. Additionally, the maintenance energy should not be ignored. We use *EFM* to represent the power consumption caused by interface per second. Finally, *T_T* denotes the tail time, which is the time spent in the high-power state after the transfer.

Based on the energy models, we formulate the energy consumption models for cloud execution of mobile tasks as follows:

$$\begin{aligned} E\_Cloud = {} & RT(Total\_Transmissiondata) + TE \times T\_T \\ & + EFM \times \frac{Total\_Transmissiondata}{Ave\_Speed} \\ & + Energy\_Receiving \end{aligned} \qquad (4)$$

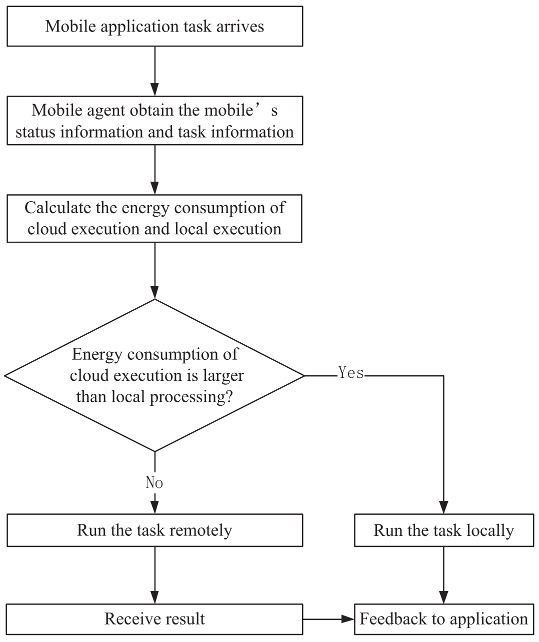
It should be noted that the values of *TE* and *T_T* equal to zero under WIFI environment. The value of *Ave_Speed* in different connection environments is determined based on the models in Table II. Besides, *Energy_Receving* denotes the energy consumption in receiving task results, and we set it to be constant values in different connection environment.

3.6. Task offloading algorithm

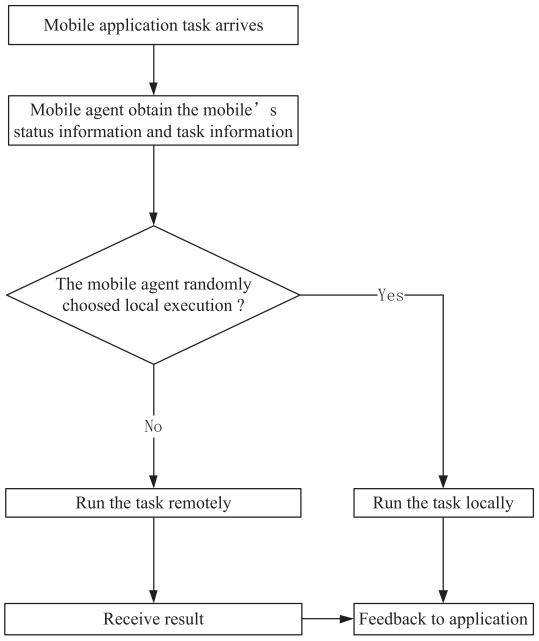
In this section, we design the energy efficient scheduling method (AGILE) for mobile application tasks based on the previously mentioned models.

The framework of AGILE is depicted in Figure 2(a). And a random selection scheduling method (RSSM), whose framework is presented in Figure 2(b), is proposed to be a baseline for AGILE.

The main idea of AGILE is to save energy for mobile devices, thus extending the battery life. The detailed


Mobile application task arrives
Mobile agent obtain the mobile' s status information and task information
Calculate the energy consumption of cloud execution and local execution
Energy consumption of cloud execution is larger than local processing?
Yes
No
Run the task remotely
Run the task locally
Receive result
Feedback to application

(a) The AGILE Framework


Mobile application task arrives
Mobile agent obtain the mobile' s status information and task information
The mobile agent randomly choosed local execution ?
Yes
No
Run the task remotely
Run the task locally
Receive result
Feedback to application

(b) The RSSM Framework

Figure 2. The framework of task scheduling methods. (a) The AGILE framework and (b) The random selection scheduling framework.

AGILE is described in Algorithm 1. In AGILE, when a task T arrives, the mobile agent will obtain the arguments of T and require the interfaces of devices to get status information (connection environment, battery information) of the mobile devices (Lines 1-3). Then, the energy consumption of mobile execution and cloud execution are determined

Algorithm 1: AGILE:Terminal Energy Efficient Scheduling Method

```
1  $m^A \leftarrow Data\_Size, Task\_Length$ when $T$ arrived;
2  $m^A$ query current status information;
3  Select the value of $\alpha, \beta, \gamma$ according to current status;
4  $Total\_Transmissiondata \leftarrow \frac{1+\beta}{1-\gamma} \times Data\_Size$;
5  Determine the value of $Ave\_Speed$ in terms of status
   information;
6  $Total\_Instruction \leftarrow (1+\alpha) \times Task\_Length$;
7  $E\_Local \leftarrow EpI \times Total\_Instruction$;
8  Compute $E\_Cloud$ based on table 2 and Equation(6);
9  if $E\_Cloud > E\_Local$ then
10     Schedule the task to local execution;
11 else
12     Upload task data to $CS^A$ for cloud execution;
13     $m^A$ receives task result from $CS^A$;
```

Algorithm 2: RSSM:Random Selection Scheduling Method

```
1  $m^A \leftarrow Data\_Size, Task\_Length$ when $T$ arrived;
2  $m^A$ query current status information;
3  Select the value of $\alpha, \beta, \gamma$ according to current status;
4  $Total\_Transmissiondata \leftarrow \frac{1+\beta}{1-\gamma} \times Data\_Size$;
5  Determine the value of $Ave\_Speed$ in terms of status
   information;
6  if $m^A$ randomly selected local execution then
7      Schedule the task to local execution;
8  else
9      Upload task data to $CS^A$ for cloud execution;
10     $m^A$ receives task result from $CS^A$;
```

by m^A based on the information acquired before (Lines 4-8). When the cloud execution consumes more energy than mobile execution, T will be run in local device (Lines 9-10). When the local run consumes more energy, m^A sends task T to CS^A for cloud execution (Lines 11-13).

In Algorithm 2, the elaborate steps of RSSM are given. The first several steps are the same with AGILE (Lines 1-5). However, the decision about whether offloading task T into cloud or not is made by m^A randomly rather than by comparing the energy consumption. If task T can be run in a cloud server, then the m^A randomly determines the schedule decision (Line 6). When m^A chooses local execution, T will be run in local device (Line 7). When the m^A determines to offload task into cloud server, T will be send to CS^A by m^A for cloud execution (Lines 8-10).

It should be noted that the random selection of local execution or cloud execution is based on the average speed produced in Line 5. This is because a mobile with higher transmission speed has higher probability to offload tasks to cloud for processing. In our work, we use the following formula to determine the probability (P) of a cloud execution for a task:

$$P = \frac{\lg Ave\_Speed}{\lg maximal\ speed} \tag{5}$$

4. EVALUATION

In Section 3, we proposed an algorithm AGILE for task offloading in mobile cloud computing environment. In order to certify the effectiveness of AGILE, RSSM is put forward to be a baseline. Another baseline, named Local Only (LO), is that the tasks run on mobile device only. The main idea of AGILE is to save energy for mobile devices. Thus, a metric, energy consumption (EC for short), should be taken into account. Another metric considered in this article is the proportion of tasks run in cloud server, we called it cloud execution proportion (CEP for short). The definition of the two metrics are given as follows.

- *Energy consumption*: the energy consumed by all tasks. We only consider the mobile energy consumption, but energy consumption in cloud is not included.
- *Cloud execution proportion*: the proportion of tasks executed in cloud. It can be calculated by Equation (6).

$$CEP = \frac{\textit{number of tasks executed in cloud}}{\textit{total task number}} \times 100\% \quad (6)$$

4.1. Simulation setup

Based on the aforementioned models, we conduct extensive simulation experiments on CloudSim [30]. We tested the performance with the changing of task count under different connection environments. Other simulation settings are as follows:

- *Data_Size* of task *T* is set as follows: *Data_Size* = *Math.round* (*uniform* (0.5,5)), which is measured by *MByte*.
- *Task_Length* is produced by the following function:

$$\textit{Task\_Length} = \textit{Math.round}(\textit{uniform}(6 \times 10^{11}, 14 \times 10^{11}))$$

- Referring to the work in [31], the soft error rate α can be about 10% $\backsim$ 30%. When computing the total instructions, the value of α is set to be a random value between the interval.
- We set the value of *EpI* to be 91 *pJ/instruction* like that in [32].
- The value of data encryption rate β is determined by the function as follows:

$$\beta = \textit{Math.round}(\textit{uniform}(0.05, 0.2))$$

- The connection environment is dynamically determined by m^A at task *T*' arrival. After the 2G, 3G or WIFI environment is obtained, the *Ave_Speed* is randomly produced according to its typical interval by mobile agents.
- As mentioned earlier, retransmission rate γ is inversely proportional to the connection environment. Based on the value of *Ave_Speed*, we set the value of γ by

$$\gamma = \frac{\textit{Math.round}(\textit{uniform}(0.1, 0.3))}{\lg \textit{Ave\_Speed}}$$

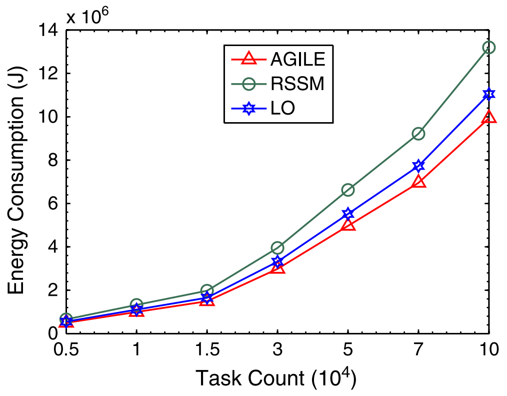
- The receiving energy for a cloud execution task is set to be 10*J*.
- Each group of experiment run five times. And the mean value is calculated to be the experimental results.
- Performance of the algorithm is evaluated under different task count, which varies from 5000 to 50 000.

4.2. Energy consumption under different connection environment

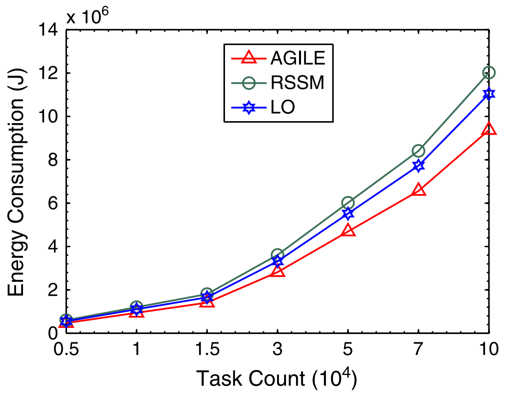
In this group of experiment, the energy consumption is to be measured under 2G, 3G and WIFI environments. The energy consumption by local execution of all tasks is also given to provide baseline for AGILE and RSSM. The fold line marked by AGILE, RSSM and LO represent the experimental results of AGILE algorithm, RSSM algorithm and local execution, respectively. We compare AGILE, RSSM and LO in terms of energy consumption, and Figure 3 illustrates the experimental results.

It can be seen from Figure 3(a) that, under 2G environment, AGILE algorithm achieves energy saving goals compared with LO. This is because the AGILE computes the energy consumption of cloud execution and local execution and then choose the least energy consumption method for the task. However, the RSSM consumes more energy than that of LO. This is because the RSSM randomly selects cloud execution or local execution for a certain task, which results more energy consumption. Although the probability of choosing cloud execution for tasks is highly related to the connection environment (reflected by average transmission speed), it has no idea about offloading the cloud execution fitted tasks. That is to say, the probability of offloading tasks to cloud may be reasonable, but the tasks chose to be offload is not the suitable ones. For example, under random selection scheme, a task with a large data size and few instructions, which is propitious to local execution in terms of energy saving, may be offload to cloud server and bring more energy consumption than executing locally. Another fact that can be observed is that the performance of AGILE in energy saving is always better than that of LO with the increase of task count, which indicates the stability of our proposed algorithm. The experimental results reveal that we can offload tasks into cloud for execution to save energy for mobile device. The AGILE outperforms LO 9.95% in energy saving while LO outperforms RSSM 16.31% under 2G connection environment.

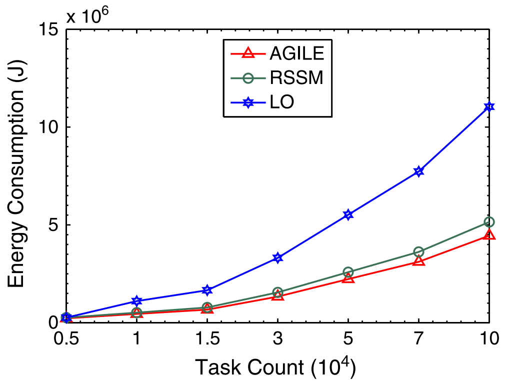
In Figure 3(b), similar analysis is made like that in 2G environment. However, the performances of AGILE and RSSM improved compared with that in Figure 3(a). This is because of the decrease of energy consumed by data transmission. So, more tasks can be offload to decrease



(a) 2G environment



(b) 3G environment



(c) WIFI environment

Figure 3. Energy consumption under different connection environments. (a) 2G environment, (b) 3G environment, and (c) WIFI environment. AGILE, terminal energy efficient scheduling method; RSSM, random selection scheduling method; LO, local execution.

mobile energy consumption. It should be noted that the promotion of RSSM is caused by the the improvement of the connection environment. Under 3G channels, more tasks are suitable for cloud execution, so energy consumption under RSSM decreased to the level around LO. AGILE outperforms LO 14.94% in terms of energy saving, and LO outperforms RSSM 8.21% under 3G channels.

From Figure 3(c), we can see that AGILE and RSSM save much more energy than LO. This can be explained in that energy consumption of transmitting data has rapidly reduced under WIFI environment. From Table II, we can easily find that there is no tail energy under WIFI channels, and the transmission energy for big data size tasks is much less than that under 2G and 3G channels. As a result, much more tasks with larger data size, which are not suitable for cloud execution under 2G and 3G connections to economise energy, are able to be processed in cloud for the sake of energy saving. So, either AGILE or RSSM achieves energy-saving goal under WIFI environment. The AGILE outperforms LO 59.65%, and RSSM outperforms LO 53.26% under WIFI channels.

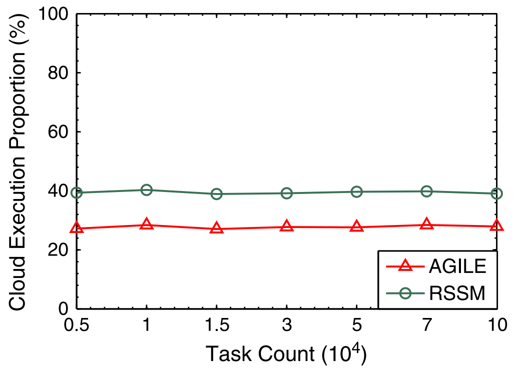
From the comparisons among Figure 3(a), (b) and (c), we can easily find that when the connection environment changes, the energy consumption by LO are basically identical. This can be explained that, under LO strategy, all tasks are processed locally, so the connection environment has no impact on its energy consumption. However, the performances of AGILE and RSSM are significantly improved. This is because the data transmission energy are significantly reduced with the advancement of connection channels, which leads to the decrease of energy consuming for cloud execution. From the experiment results, we can see that the performance of AGILE has improved 45.22% when the connection environment changes from 2G to WIFI. Another observation is that the better the connection environment is, the more likely the task can be offloaded to save mobile device energy, leading to extend the battery life.

Consequently, compared with RSSM, our proposed offloading algorithm AGILE has obvious superiorities in energy consumption. When the connection environment improves, both AGILE and RSSM can get promotion. Besides, with the increase of task count, AGILE can save more energy than LO and RSSM, from which we conclude that AGILE owes superiority and stability.

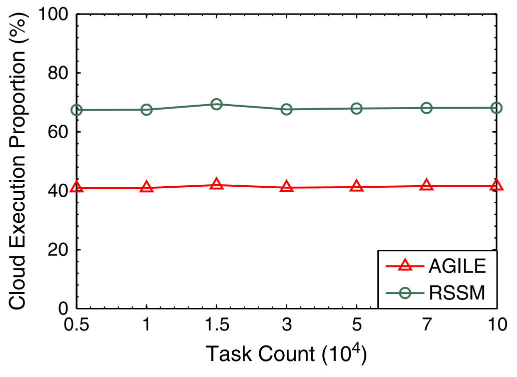
4.3. Cloud execution proportion under different connection environment

In this group of experiment, the percentages of tasks executed on cloud server of AGILE and RSSM are to be measured under different connection environments. The fold lines marked by AGILE and RSSM represent the experimental results of AGILE algorithm and RSSM algorithm, respectively.

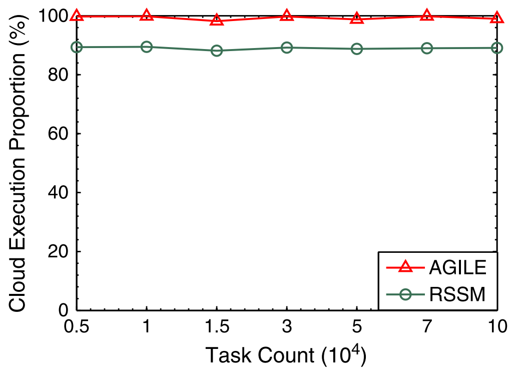
Figure 4 depicts that the values of cloud execution proportion for RSSM and AGILE under different connection environments have much differences. For AGILE, that is because the decision of offloading a mobile application task to cloud server is made based on the task parameters and connection environment. The task arguments are produced by random functions, which makes the arguments vary around their mean value. Thus, there exists a mean



(a) 2G environment



(b) 3G environment



(c) WIFI environment

Figure 4. Cloud execution proportion under different connection environments. (a) 2G environment, (b) 3G environment, and (c) WIFI environment. AGILE, terminal energy efficient scheduling method; RSSM, random selection scheduling method.

value of cloud execution proportion under a given wireless channel. As for RSSM, the probability of offloading a task is proportional to the average speed, and determined by Equation (5). Consequently, the average speed is changing around certain value when the transmission environment keeps the same, and the proportion of offloaded tasks stays in a certain level. As a result, either the cloud execution proportion of RSSM or AGILE in a certain connection environment keeps stable regardless of the changes of task count.

From Figure 4, we can also see that the cloud execution proportion only has slight fluctuation when the task count varies. For AGILE, the offloading of a mobile application task is mainly determined by the connection environment and task parameters, which makes the cloud execution only influenced by the randomness or fluctuation of arguments. Also, we can see that with the increase of task count, the fluctuation becomes smaller and smaller. This can be explained by the law of large numbers, that is to say, the random perturbation of the experiment decreases with the growth of sample numbers. Consequently, there is fluctuations in the experiment results of AGILE, and the curve becomes smoother with the increase of task count. As for RSSM, the offloading of mobile tasks largely depends on the average speed of different connection channels. So, the cloud execution proportion is around a certain level.

By comparing Figures 4(a), (b) and (c), we find that the cloud execution proportion grows with the promotion of the connection environment. This can be explained in that the average transmission speed becomes higher and the energy consumption becomes lower when the connection channel varies from 2G to WIFI. Thus, more and more tasks become appropriate to be executed in cloud from the perspective of energy saving under AGILE. Besides, the growth of average speed leads that more tasks are offloaded to cloud server under RSSM. Consequently, the value of cloud execution proportion increases when the connection environment becomes better.

The observations from Figures 3 and 4 show that the energy saving is not in direct proportion to the number of cloud execution tasks. Take Figures 3(b) and 4(b) as an example, the value of cloud execution proportion of RSSM is higher than that of AGILE, but the energy consumption of RSSM is not less than AGILE. We can conclude that not all the task can be offloaded to save energy, and false offloading rate may bring extra energy consumption. In the latter subsections, the influence on the performance caused by parameter variation will be studied.

4.4. Performance impact on data size

In the former parts, we can get that mobile application task's offloading to cloud server is able to save energy for mobile devices. Besides, data size and task length are concluded to be two main factors, which determine the offload decision. In this group of experiment, we test the performance changes by data size variation.

In the aforementioned definition, data size is produced by the following formula: $Data\_Size = Math.round\ (uniform\ (0.1 * ds, ds))$, and the value of ds is set to be 5. In this group of experiment, we choose the 3G environment, task count is set to be 50 000 and the value of ds varies from 2 to 8.

From Figure 5(a), we observe that energy consumption of LO keeps the same when the value of parameter ds varies. This can be explained that only the number of

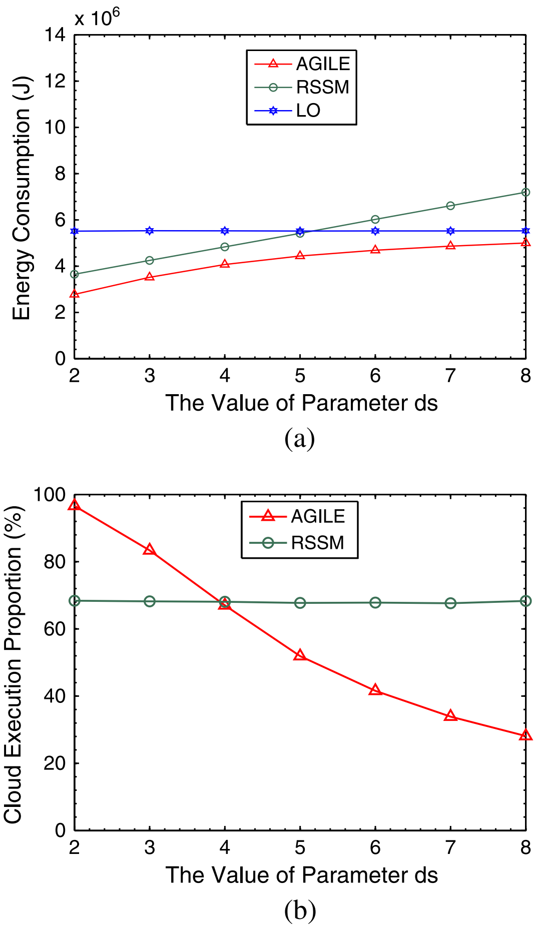


Figure 5. Performance impact on data size. AGILE, terminal energy efficient scheduling method; RSSM, random selection scheduling method; LO, local execution.

instructions makes the difference on energy consuming when all tasks are executed locally, and the changes in data size have no influence on energy consumption under LO. As a result, the energy consumption of LO nearly keeps the same with the increase of *ds*'s value. Unlike LO, the energy consumption of RSSM and AGILE do have changes with the variation of parameter *ds*. For AGILE, it can be explained that the increment of data size brings more energy depletion for the cloud execution form, which results that less tasks can be offloaded to save energy for mobile device. Consequently, energy consumption of mobile application tasks under AGILE increases with the growth of parameter *ds*'s value. As for RSSM, the explanation for the increasing of energy consumption is like that of AGILE. However, the raising speed of RSSM is larger than that of AGILE. It can be explained that when the data size rises to a critical level at which energy depletion of cloud execution and local execution are equal, AGILE chooses local execution to avoid the raise of energy consuming while RSSM does not. Hence, the energy consumption under RSSM grows faster than that of AGILE.

Figure 5(b) shows that cloud execution proportion under RSSM is not influenced by the variation of parameter *ds*' value. This can be explained that the probability to offload a mobile application task to cloud server is determined by the average transmission speed of a mobile device when using RSSM. Moreover, the data size of a tasks is not taken into account. Therefore, the cloud execution proportion keeps stable with the changes in parameter *ds*' value. Different from RSSM, AGILE is influenced by parameter *ds*. From Figure 5(b), we observe that cloud execution proportion of AGILE decreases when the value of *ds* rises. The experimental result can be explained that, for certain tasks, energy consumption for cloud execution becomes larger than that of local execution when the data size rises to a bound level, so the number of tasks that can be offloaded to save energy decreases. As a result, the cloud execution proportion drops down with the growth of parameter *ds*' value.

From Figure 5(a) and (b), we can make the conclusion that mobile application tasks with larger data size are less likely to be able to be offloaded for energy conservation. Besides, negative correlation exists between cloud execution proportion and data size of tasks. Consequently, we ought to offload the tasks with small data size to cloud server from the perspective of energy saving.

4.5. Performance impact on task length

In this group of experiment, we study the performance impact on task length.

As mentioned before, task length is generated by the following formula: $Task\_Length = Math.round(uniform(tl \times 10^{11}, (tl + 8) \times 10^{11}))$, and the value of tl is set to be 6. In this group of experiment, the value of tl varies from 3 to 9. Additionally, 3G environment is selected, and task number is 50 000.

From Figure 6(a), we can find that energy consumptions of all the three offloading strategies grow when the value of parameter *tl* increases. For LO, that is because the energy consumption is calculated based on task length and energy consumption per instructions, and a larger *tl* leads to more energy consumption for certain tasks. Thus, the energy consumption of LO increases with the growth of *tl*'s value. As for AGILE, this can be explained that when task length becomes longer, local execution for mobile application tasks consume more energy, which results in more tasks with large data size being offloaded to save energy. Consequently, the power consumption of AGILE rises. Besides, under RSSM, the locally executed tasks consume more energy with the variation of *tl* value. Hence, the performance of the three offloading algorithms have the same trend, that is, growing with the increase of *tl*'s value.

Another fact that can be concluded is that the energy consumption of LO has the highest increasing speed, while RSSM is the opposite. The reason for this experimental result is that, when the value of *tl* grows, the energy consumption for local executing increases, which is totally reflected in LO as all tasks are processed in mobile devices.

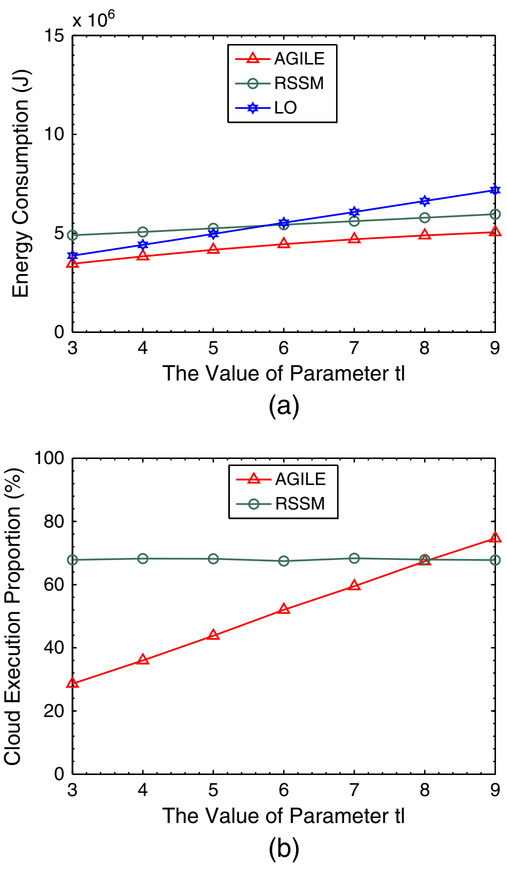


Figure 6. Performance impact on task length. AGILE, terminal energy efficient scheduling method; RSSM, random selection scheduling method; LO, local execution.

However, only part of the increasing is reflected in RSSM because *tl*'s variation merely has influence on the locally executed proportion. As for AGILE, the local executing proportion is like the performance of RSSM, but the cloud execution is also influenced by *tl*'s variation, and some tasks can be offloaded to save part of the energy. So, the cloud execution proportion of AGILE grows faster than RSSM while slower than LO.

Figure 6(b) depicts that the cloud execution proportion under RSSM is not influenced by parameter *tl*'s value. This can be explained like that in Figure 5(b). However, the performance of AGILE suffers the impact of *tl*'s variation. We observe from Figure 6(b) that the cloud execution proportion of AGILE increases with the growth of *tl*'s value. The reason is that the energy consumption for cloud execution becomes lager than that of local execution when the task length exceeds bound level, so the number of cloud execution tasks increases. As a result, the cloud execution proportion increases with the growth of parameter *ds*'s value.

We can conclude from Figure 6 that mobile application tasks with larger task length are more likely to be offloaded to cloud server for energy saving. Thereby, we should offload the tasks with larger task length to cloud server to save mobile devices' power, thus to achieve the goal of extending battery life.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we give the elaborate design of system models and algorithm, and agent model is proposed to shield the heterogeneity of different mobile platforms and to support the communication among mobile devices and cloud servers. Besides, the mobile execution energy model and data transmission energy consumption model are presented. Different connection environments like 2G, 3G and WIFI are considered to simulate the real world. Further more, we suggested the offloading algorithm AGILE and RSSM based on the proposed models. Extensive simulation experiments were conducted to evaluate the performance of AGILE and RSSM, and the experimental results show that AGILE has advantages on solving mobile application tasks' offloading problem and can efficiently save energy.

In our further study, three issues will be considered. First, we will further study software error, transmission error, encryption rate and so on, and redefine the producing method of these parameters to make the simulation more consistent with the real world. Second, multi-data centers are to be taken into account, and the trade-off between them will be investigated. Third, we will consider the experiment with real-world traces, and to test the performance in our daily-used mobiles.

ACKNOWLEDGEMENT

This work is supported by the Research Fund for the Doctoral Program of Higher Education(RFDP) of China with No.20134307110029, the National Natural Science Foundation of China(Grant No.61374185 and No.11428101).

REFERENCES

1. 2012. Avaialble from: https://www.google.com/mobile/android/ [accessed on 30 June 2014].
2. 2014. Avaialble from: http://www.apple.com/iphone/?cid=oas-us-domains-iphone.com [accessed on 30 June 2014].
3. Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, Zaharia M. A view of cloud computing. *Communications of the ACM* 2010; **53**(4): 50–58.
4. Dinh HT, Lee C, Niyato D, Wang P. A survey of mobile cloud computing: architecture, applications, and approaches. *Wireless Communications and Mobile Computing* 2013; **13**(18): 1587–1611.

5. Chun BG, Maniatis P. Dynamically partitioning applications between weak devices and clouds. In *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*, San Francisco, CA, USA, 2010; 7.
6. Kosta S, Aucinas A, Hui P, Mortier R, Zhang X. Thinkair: dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In *2012 Proceedings IEEE International Conference on Computer Communications*, Orlando, FL, USA, 2012; 945–953.
7. Ottaviani V, Lentini A, Grillo A, Di Cesare S, Italiano GF. Shared backup & restore: save, recover and share personal information into closed groups of smartphones. In *New Technologies, Mobility and Security (NTMS), 2011 4th IFIP International Conference on*, Paris, France, 2011; 1–5.
8. Barbera MV, Kosta S, Mei A, Stefa J. To offload or not to offload? The bandwidth and energy costs of mobile cloud computing. In *2013 Proceedings IEEE International Conference on Computer Communications*, Turin, Italy, 2013; 1285–1293.
9. Wooldridge M, Jennings NR. Intelligent agents: theory and practice. *The Knowledge Engineering Review* 1995; **10**(2): 115–152.
10. Kumar K, Lu YH. Cloud computing for mobile users: Can offloading computation save energy? *Computer* 2010; **43**(4): 51–56.
11. 2012. Avaialble from: http://www.theenvironmentalblog.org/2012/08/intels-ivy-bridge-processors-energy-efficient-cpus/ [accessed on 30 June 2014].
12. Cuervo E, Balasubramanian A, Cho D, Wolman A, Saroiu S, Chandra R, Bahl P. MAUI: making smartphones last longer with code offload. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, California, USA, 2010; 49–62.
13. Balasubramanian N, Balasubramanian A, Venkataramani A. Energy consumption in mobile phones: a measurement study and implications for network applications. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, Barcelona, Spain, 2009; 280–293.
14. Xian C, Lu YH, Li Z. Adaptive computation offloading for energy conservation on battery-powered systems. In *2007 International Conference on Parallel and Distributed Systems*, Hsinchu, Taiwan, 2007; 2, 1–8.
15. Zhang W, Wen Y, Guan K, Kilper D, Luo H, Wu DO. Energy-optimal mobile cloud computing under stochastic wireless channel. *IEEE Transactions on Wireless Communications* 2013; **12**(9): 4569–4581.
16. Koshy KI, Juby AM, Namboodiri V, Overcash M. Can cloud computing lead to increased sustainability of mobile device? In *Proceedings of 2012 IEEE International Symposium on the Sustainable Systems and Technology (ISSST)*, Nanjing, Jiangsu Province, China, 2010; 1–4.
17. Miettinen AP, Nurminen JK. Energy efficiency of mobile clients in cloud computing. In *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*, Boston, MA, 2010; 4.
18. Othman M, Hailes S. Power conservation strategy for mobile computers using load sharing. *ACM SIGMOBILE Mobile Computing and Communications Review* 1998; **2**(1): 41–45.
19. Rachuri KK, Mascolo C, Musolesi M, Rentfrow PJ. SociableSense: exploring the trade-offs of adaptive sampling and computation offloading for social sensing. In *Proceedings of the 17th Annual International Conference on Mobile Computing and Networking*, Las Vegas, Nevada, USA, 2011; 73–48.
20. Barbera MV, Stefa J, Viana AC, Dias de Amorim M, Boc M. VIP delegation: enabling VIPs to offload data in wireless social mobile networks. In *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, Barcelona, Spain, 2011; 1–8.
21. Chun BG, Maniatis P. Augmented smartphone applications through clone cloud execution. *HotOS* 2009; **9**: 8–11.
22. Chun BG, Ihm S, Maniatis P, Naik M, Patti A. CloneCloud: elastic execution between mobile device and cloud. In *Proceedings of the Sixth Conference on Computer Systems*, Salzburg, Austria, 2011; 301–314.
23. Chen G, Kang BT, Kandemir M, Vijaykrishnan N, Irwin MJ, Chandramouli R. Studying energy trade offs in offloading computation/compilation in java-enabled mobile devices. *IEEE Transactions on Parallel and Distributed Systems* 2004; **15**(9): 795–809.
24. Hung SH, Shieh JP, Chen YW. A profile-driven dynamic application offloading scheme for android systems. In *2012 IEEE 1st Global Conference on Consumer Electronics (GCCE)*, Tokyo, Japan, 2012; 540–541.
25. Rong P, Pedram M. Extending the lifetime of a network of battery-powered mobile devices by remote processing: a Markovian decision-based approach. In *Proceedings of the 40th Annual Design Automation Conference*, Anaheim, CA, USA, 2003; 906–911.
26. Huang D, Wang P, Niyato D. A dynamic offloading algorithm for mobile computing. *IEEE Transactions on Wireless Communications* 2012; **11**(6): 1991–1995.
27. 2012. Avaialble from: http://www.windowsphone.com/en-us [accessed on 30 June 2014].

28. 2014. Avaialble from: http://en.wikipedia.org/wiki/4G [accessed on 30 June 2014].
29. 2014. Avaialble from: http://support.en.belgacom.be/app/answers/detail/a_id /13580/~/wi-fi-and-mobile-internet-(2g,-3g,-3g%2B-and-4g) [accessed on 30 June 2014].
30. Calheiros RN, Ranjan R, Beloglazov A, De Rose CAF, Buyya R. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience* 2011; **41**(1): 23–50.
31. Weaver C, Emer J, Mukherjee SS, Reinhardt SK. Techniques to reduce the soft error rate of a high-performance microprocessor. In *Proceedings. 31st Annual International Symposium on Computer Architecture*, Munich, Germany, 2004; 264–275.
32. Goulding-Hotta N, Sampson J, Venkatesh G, Garcia S, Auricchio J, Huang PC, Arora M, Nath S, Bhatt V, Babb J, Swanson S, Taylor M. The GreenDroid mobile application processor: an architecture for silicon's dark future. *IEEE Micro* 2011; **31**(2): 86–95.