Multi-device task offloading with time-constraints for energy efficiency in mobile cloud computing

Kaiyang Liu, Jun Peng *, Heng Li, Xiaoyong Zhang, Weirong Liu

School of Information Science and Engineering, Central South University, Changsha 410083, China

HIGHLIGHTS

- A task offloading decision method is proposed among multi-devices for energy saving.
- The problem is formalized as a 0–1 nonlinear integer programming problem.
- An iterative decoupling algorithm that combines with decision-variable relaxation and convex optimization is proposed for near-optimal decisions.

ABSTRACT

Nowadays, in order to deal with the increasingly complex applications on mobile devices, mobile cloud offloading techniques have been studied extensively to meet the ever-increasing energy requirements. In this study, an offloading decision method is investigated to minimize the energy consumption of mobile device with an acceptable time delay and communication quality. In general, mobile devices can execute a sequence of tasks in parallel. In the proposed offloading decision method, only parts of the tasks are offloaded for task characteristics to save the energy of multi-devices. The issue of the offloading decision is formulated as an NP-hard 0–1 nonlinear integer programming problem with time deadline and transmission error rate constraints. Through decision-variable relaxation from the integer to the real domain, this problem can be transformed as a continuous convex optimization. Based on Lagrange duality and the Karush–Kuhn–Tucker condition, a solution with coupled terms is derived to determine the priority of tasks for offloading. Then, an iterative decoupling algorithm with high efficiency is proposed to obtain near-optimal offloading decisions for energy saving. Simulation results demonstrate that considerable energy can be saved via the proposed method in various mobile cloud scenarios.

1. Introduction

Nowadays, we have witnessed an explosive growth of mobile devices. According to the Cisco Visual Networking Index report, the number of mobile devices will exceed the number of people on earth by the end of 2015 [1]. Furthermore, numerous mobile applications have been emerging, changing the lifestyles of people and bringing us considerable convenience. Nevertheless, mobile devices have their inherent problems, such as finite computing power, low connectivity, and particularly limited battery life [2]. In fact, many applications operated on mobile devices are resource-intensive, such as navigation, high-definition image processing, face recognition, online mobile games, and sensor data processing [3–5]. In recent years, mobile cloud computing technologies have been envisioned as promising and challenging technologies to overcome these limitations [6].

As a newly emerging computing paradigm, mobile cloud computing brings a new idea to augment the capabilities of mobile devices by offloading computing tasks to resource-rich (e.g., CPU, storage, and network bandwidth) cloud servers [7]. Nowadays, various mobile cloud platforms have been deployed for energy-hungry applications. Reducing the energy consumption of mobile devices can lead to various benefits such as long battery life, no overheating, and increased system reliability. For instance, a mobile cloud service can improve the energy efficiency of wireless body sensors in e-Health systems [8].

In all recent mobile cloud schemes, the cloud offloading mechanism is treated as one of the most powerful and indispensable techniques that can potentially save energy for mobile users [9]. However, the current research efforts for the offloading mechanism are still limited and have defects that cannot be ignored [10]. The foremost challenge is that cloud offloading with multiple
mobile devices may cause considerable mutual communication interference [11], which will cause a reduction of the wireless communication quality. It is worth noting that an inferior wireless communication quality may potentially increase the running latency of the cloud offloading process [12]. Previous researches have shown that the existing cloud offloading systems, such as Gaïkal, can cause communication time delays of up to 400 ms [13]. Moreover, a relatively long running latency and inferior wireless communication quality may result in increased energy consumption by mobile devices. In conclusion, if a considerably large number of mobile devices are offloading tasks to the cloud, with a relatively long time delay and inferior communication quality, cloud offloading will lose the advantage of energy conservation for mobile devices.

Hence, several important issues arise: Can all the above mentioned limitations be overcome to optimize the energy efficiency of multiple mobile devices? Can mobile devices obtain a cloud offloading service with a relatively low time latency and acceptable wireless communication quality? In general, the task offloading mechanism is leveraged to ease these issues [14]. In fact, most of the available mobile devices support multitasking. Thus, a mobile device may process parallel tasks simultaneously [15]. Recent advances in cloud offloading show that not all mobile applications are suitable to be offloaded for their different task characteristics, i.e., computation-intensive or communication-intensive [16]. The computational workload densities of computation-intensive tasks are relatively higher comparing with communication-intensive tasks. If more computation-intensive applications, e.g., mobile augmented reality [17], automatic target recognition [18] and 3D video games [19], are offloaded, more energy can be saved with cloud offloading service. Thus, to optimize the energy efficiency of mobile devices, a novel task offloading decision method that determines which tasks can be chosen for offloading is investigated in this study. The major contributions of this paper can be summarized as follows:

• To optimize the energy efficiency of mobile devices, we go deep into the task level to investigate the issue of energy conservation. The issue of the task offloading decision on a mobile device is formulated as a constrained 0–1 nonlinear integer programming problem, under the time delay and transmission error rate constraints.

• Because of its NP-hard complexity, we simplify this issue by relaxing assignment variables 0–1 from the integer to the real domain, transforming it as a standard convex optimization problem. Then, an equation with coupled terms is derived to determine the priorities of tasks for offloading by using the Lagrange duality method and the Karush–Kuhn–Tucker (KKT) techniques.

• By taking the worst case as the initial state, we have proposed an iterative decoupling algorithm with high efficiency in order to approximate the optimal offloading decisions for each mobile device. The proposed iterative algorithm combining linear relaxation and the convex optimization method can solve the NP-hard problem at a considerably lower cost, revealing near-optimal offloading decisions to minimize the energy consumption of mobile devices. The computation process is completed by the powerful cloud side, providing task offloading guidance for energy saving.

The rest of this paper is organized as follows: Section 2 briefly discusses the related work. Section 3 presents the preliminaries, defines the mobile cloud computing model, and introduces the task offloading flow path and the wireless fading channel. Then, Section 4 provides the problem formulation for the task offloading decision. Section 5 discusses efficient algorithms for solving the constrained optimization problem. Section 6 presents numerical simulations for various mobile cloud scenarios. Finally, Section 7 presents the conclusion and the future research direction.

2. Related work

2.1. Mobile cloud offloading

Recently, cloud offloading techniques have attracted considerable attention as promising and powerful techniques to augment the capabilities of mobile devices and improve user experience [20]. These works mainly focus on three aspects: (i) minimizing the energy consumption of mobile devices [9,21], (ii) minimizing the communication cost to the cloud [22,23], and (iii) minimizing the total application execution time [11,24,25].

As a pioneer research work, Kumar et al. [9] provided a trade-off analysis between the energy consumption of a mobile device and the energy consumed by sending the input data to the cloud. However, the assumption in it may not meet the reality that simply lets a single value reflect the computing capacity for all applications given that each application has its own computational requirements. Zhang et al. [21] investigated a threshold offloading method to decide whether an entire application should be offloaded or locally executed on a mobile device, aiming at the reduction of the mobile device’s energy consumption. However, this work only considers single-device offloading, ignoring the mutual interference caused by multiple mobile devices.

Furthermore, Barbera et al. [22] considered the bandwidth and the energy cost of mobile computer offloading, investigating the feasibility of both mobile cloud offloading and mobile data backup systems in a real setting. In vehicular cyber-physical systems, Wang et al. [23] modeled mobile data traffic offloading as a multi-objective optimization problem for the simultaneous minimization of mobile data traffic and QoS-aware service provision.

More recently, considering the overhead of cloud offloading in terms of both processing time and energy, Chen et al. [11] proposed a game-theoretic approach to solve the decentralized offloading decision making problem among multiple mobile devices. To save the energy of mobile devices and meet the requirement of application execution time, Huang et al. [24] proposed an effective dynamic offloading algorithm based on Lyapunov optimization, which provides a near-optimal solution with low complexity. Zhang et al. [25] investigated the collaborative task execution problem in the linear topology of a task model and formulated the offloading decision as a time deadline-constrained shortest path problem. An effective algorithm based on Lagrangian relaxation was proposed to minimize the energy consumption approximately.

Inspired by previous work and considering the mutual interference caused by multiple devices, in this study, we investigate the task offloading decision problem to minimize a mobile device’s energy consumption with a time delay constraint and acceptable communication quality. The offloading priorities of all tasks are derived so that more computation-intensive but not communication-intensive tasks are offloaded to the cloud side for the minimization of the energy consumption of a mobile device.

2.2. Mathematical programming method

In the field of mobile cloud computing, mathematical programming methods have been adopted widely for the optimization of the offloading performance [26]. For instance, MAUI [27] allows code offloading to a resource-rich computing infrastructure for energy efficiency. The offloading decision issue is formulated as a 0–1 integer linear programming problem. However, the complexity of the solving process makes it unsuitable for practical application. Al-Kanj et al. [28] formulated the optimal cellular offloading problem as a mixed integer linear programming problem in device-to-device communication networks. To adapt the mobility of mobile devices, a dynamic programming approach is adopted for solving
the original problem recursively. However, both problems are NP-complete, and therefore, polynomial time greedy algorithms are proposed as fast solutions with acceptable solution performance. Kang et al. [29] formulated the mobile data offloading problem as an NP-hard integer programming problem. The near-optimal centralized data offloading scheme is derived through Lagrangian relaxation. However, the quality of the data offloading service is not taken into consideration while designing the incentive offloading mechanism.

In contrast to the existing schemes, the proposed approach can minimize the mobile device energy consumption by modeling the mobile task offloading decision as a nonlinear integer programming problem, while meeting the offloading QoS guarantee with a time delay constraint and acceptable communication quality. The nonlinearity of the optimization problem makes it more difficult to solve this problem with the existing solution methods. Therefore, combinatorial methods blending decision-variable relaxation, convex optimization, and iterative decoupling algorithms are proposed to ease the NP-hard complexity with near-optimal offloading decisions.

3. Preliminaries

In this section, we present some preliminary knowledge for the mobile cloud offloading service, including the mobile cloud computing model, the task offloading flow path, and the wireless fading channel. Further, Table 1 lists all the important symbols used in this paper.

### 3.1. Mobile cloud computing model

As illustrated in Fig. 1, the mobile cloud service architecture consists of the following three key components: a mobile cloud provider, the wireless access point, and $N$ mobile devices. On the mobile device, a parallel task topology is employed. We assumed that mobile device $i$ processes $M_i$ parallel tasks, $i = 1, 2, \ldots, N$. In general, task $j$ on mobile device $i$ has computing workload $o_{i,j}$, where $j = 1, 2, \ldots, M_i$. The input data for communication are denoted as $\eta_{i,j}$, where the input data include the computing environment settings, program codes, and initial parameters. The cloud provider has a cloud controller, a computation module, and several resource-rich data servers. The cloud controller collects the requests from cloud users and manages a variety of resources from the data servers to provide the corresponding cloud services through the wireless access point. Moreover, the computation module uses the powerful parallel computing ability of clouds to execute the complex offloading decision calculations for mobile devices.

**Assumption 1.** The mobile cloud service is defined as a new paradigm that the processing of mobile applications is moved from the mobile device to powerful computing platforms located in cloud data centers. The cloud data centers provide the hardware facilities and infrastructures, enabling mobile users to reserve corresponding computing, storage, and networking components for real-time cloud service provision [30]. As the scope of the paper mainly focuses on the energy saving for mobile devices through cloud offloading, the service cost and the energy usage of the cloud server are not stressed, without affecting the completeness of the paper.

To offer rapid application and storage service, the cloud controller, servers, and wireless access points are linked with high-speed wired networks. In contrast, cloud users and wireless access points can communicate with each other via the wireless access network. To achieve better communication and interaction, we assume that mobile devices select the nearest wireless access points. Through combined link methods, cloud users and cloud providers can obtain reliable intercommunication. Similar to previous studies on mobile cloud computing [14,31], we assume a quasi-static scenario where all mobile devices move in slow motion. That is, the distance between the mobile device and the wireless access point remains the same in one offloading period (e.g., a few seconds).

### Table 1

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>The number of mobile device, $i = [1, \ldots, N]$</td>
</tr>
<tr>
<td>$M_i$</td>
<td>The number of parallel tasks on device $i$, $j = [1, \ldots, M_i]$</td>
</tr>
<tr>
<td>$\rho_{i,j}$</td>
<td>Binary variable: to offload task $j$ from device $i$ to cloud side ($\rho_{i,j} = 0$) or not ($\rho_{i,j} = 1$)</td>
</tr>
<tr>
<td>$\xi_{i,j}$</td>
<td>Binary variable: device $i$ can access cloud service ($\xi_{i,j} = 1$) or not ($\xi_{i,j} = 0$)</td>
</tr>
<tr>
<td>$\gamma_{i,j}$</td>
<td>The signal to interference plus noise ratio (SINR) in wireless communication channel</td>
</tr>
<tr>
<td>$H_i$</td>
<td>The channel gain of mobile device $i$</td>
</tr>
<tr>
<td>$G$</td>
<td>The coding gain</td>
</tr>
<tr>
<td>$R_i$</td>
<td>The channel capacity of device $i$</td>
</tr>
<tr>
<td>$e_i$</td>
<td>The bit error rate (BER)</td>
</tr>
<tr>
<td>$P_i^m$</td>
<td>The local computing energy consumption of device $i$</td>
</tr>
<tr>
<td>$P_i^c$</td>
<td>The data transmission power of device $i$</td>
</tr>
<tr>
<td>$P_i^s$</td>
<td>The standby power consumption of device $i$</td>
</tr>
<tr>
<td>$e_i^m$</td>
<td>The energy consumption of device $i$ if task $j$ is offloaded onto the cloud</td>
</tr>
<tr>
<td>$e_i^c$</td>
<td>The energy consumption of device $i$ if task $j$ is executed locally on the device</td>
</tr>
<tr>
<td>$C_i^m$</td>
<td>The computing capacity of device $i$</td>
</tr>
<tr>
<td>$C_i^c$</td>
<td>The computing capacity assigned from the cloud</td>
</tr>
<tr>
<td>$T_i^m$</td>
<td>The tasks execution time on device $i$</td>
</tr>
<tr>
<td>$T_i^c$</td>
<td>The tasks transmission and execution time of device $i$ with cloud service</td>
</tr>
<tr>
<td>$o_{i,j}$</td>
<td>The computing workload of task $j$</td>
</tr>
<tr>
<td>$\eta_{i,j}$</td>
<td>The input data size for communication of task $j$</td>
</tr>
<tr>
<td>$\Phi_i$</td>
<td>The overall energy consumption of device $i$</td>
</tr>
<tr>
<td>$\Gamma_{i,j}$</td>
<td>The error rate of total data sending process</td>
</tr>
<tr>
<td>$T_d$</td>
<td>The task execution time deadline of device $i$</td>
</tr>
<tr>
<td>$e_d$</td>
<td>The maximize error rate of input sending</td>
</tr>
<tr>
<td>$\Psi_{i,j}$</td>
<td>The offloading priority of task $j$ on device $i$</td>
</tr>
</tbody>
</table>
Therefore, all tasks are executed locally. Otherwise, if $\gamma$ derived from [11] as follows:

$$\gamma = \frac{P_i H_i}{\sigma^2 + \sum_{r=1, r \neq i}^{N} \xi_r P_r H_r}. \quad (2)$$

In Fig. 2, the task offloading flow-path of mobile device $i$ in mobile cloud computing scenario.

### 3.2. Task offloading flow path

As illustrated in Fig. 2, the time-constrained task offloading flow path for a mobile device is divided into three phases. To begin with, all mobile devices collect the characteristic information of all tasks, $\{\alpha_{ij}\}$ and $\{\eta_{ij}\}$. Then, this task information is sent to the cloud controller for decision making. In the second phase, the computation module in the cloud undertakes the complicated calculations of the offloading decision for each mobile device. Let the binary variable $\rho_{ij} \in \{0, 1\}$ be the offloading flag of the tasks. $\rho_{ij} = 1$ denotes that task $j$ is executed on mobile device $i$, while $\rho_{ij} = 0$ denotes that task $j$ is offloaded onto the cloud side. According to the offloading decisions, an appropriate number of VMs are deployed in the data servers for cloud execution. Moreover, the offloading decisions are sent back to all mobile devices. Finally, the mobile devices receive the offloading decisions and decide whether to build a wireless link for offloading. We let $\xi_{ij} \in \{0, 1\}$ be the service flag of the mobile devices, denoting whether a mobile device can access a cloud service.

$$\xi_i = 1 - \prod_{j=1}^{M_i} \rho_{ij}. \quad (1)$$

For mobile device $i$, if $\xi_{ij} = 0$, no wireless connection is needed; therefore, all tasks are executed locally. Otherwise, if $\xi_{ij} = 1$, the wireless link is built and all the selected tasks are offloaded to the data servers. As soon as the executions on the mobile device and the cloud side are completed, the mobile device receives the output data from the application servers and pools the output data.

### 3.3. Wireless fading channel over cellular networks

We assume that a mobile device can access a cloud service in a wireless fading channel over cellular networks. The wireless fading model is then introduced to evaluate the performance of cloud offloading. Without any loss of generality, the wireless access points can be a wireless base station in the cellular network [32]. Considering the mutual interference caused by other mobile devices and the background interference, the signal-to-interference-plus-noise ratio (SINR) in the $\gamma_1$ channel can be derived from [11] as follows:

$$\gamma_1 = \frac{P_i H_i}{\sigma^2 + \sum_{r=1, r \neq i}^{N} \xi_r P_r H_r}. \quad (2)$$

Here, $P_i$ represents the data transmission power of mobile devices. $H_i = d_i^{-a}$ denotes the channel gain, where $d_i$ indicates the distance between mobile device $i$ and the wireless access point. Further, $\alpha$ denotes the path loss factor. As can be seen in formula (2), an increase in the number of mobile devices produces additional interference. However, this effect can be suppressed by using signal processing and channel coding techniques to produce a positive coding gain [29]. With the coding gain $G$ (dB), the improved SINR $\gamma'_j$ is calculated as follows [33]:

$$\gamma'_j = \gamma_1 \cdot 10^\frac{G}{10}. \quad (3)$$

Then, according to the Shannon principle, the channel capacity of mobile device $i$ can be derived as follows:

$$R_i = W \log_2 (1 + \gamma'_j). \quad (4)$$

where $W$ denotes the bandwidth. Moreover, the theoretical analysis in [34] indicates that the number of transmission errors is equivalent to the SINR in a digital baseband. The bit error rate (BER) $\epsilon_1$ can be expressed as follows:

$$\epsilon_1 = Q(\sqrt{2\gamma'_j}), \quad (5)$$

where $Q(x)$ is used for computing the probability of error in communication systems. Normalized to a zero mean and a unit variance, $Q(x)$ can be defined as follows [31]:

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2} dt. \quad (6)$$

### 4. Problem formulation for task execution

In this section, we consider all four energy consumption and time delay models involved in the task execution process. Then, we formulate the constrained optimization problem for energy saving.

#### 4.1. Energy consumption and time delay models

**Task execution on mobile devices**: For local task execution, we consider a mobile device that can handle multiple tasks sequentially. In general, modern processors of mobile devices have the dynamic voltage and frequency scaling (DVFS) ability. Therefore, the processors can adjust their computing capacity $C_{ij}$ according to the task (the frequently-used unit of computing capacity is cycles/Hz, i.e., Hz [9,11]). In general, the CPU clock can be adjusted from the bottom frequency $C_{ij \min}$ to the topmost frequency $C_{ij \max}$ in steps; i.e., $C_{ij} \in [C_{ij \min}, \ldots, C_{ij \max}]$. Each task requires a different CPU computing capacity per bit. For instance, a 3D video game application has considerably higher computation requirements per bit than an image processing application. If task $j$ is executed on a mobile device, with its computing workload $\alpha_{ij} \in \mathbb{N}$ (in cycles), the task execution time $t_{ij}^m$ can be calculated as follows:

$$t_{ij}^m = \frac{\alpha_{ij}}{C_{ij}}. \quad (7)$$

Then the classic CPU energy consumption model of mobile devices is adopted [35,36]. The energy consumption of task $j$ executed locally on mobile device $i$ can be calculated as follows:

$$e_{ij} = (\alpha_j(C_{ij}^{\gamma_j} + \beta_j)C_{ij}). \quad (8)$$

where the exponent $\gamma_j$ ranges from 2 to 3, and $\alpha_j$ and $\beta_j$ are the parameters determined by the CPU processing model. In Section 6, all the energy consumption parameters measured from real-world mobile devices are specified for a realistic simulation.
**Task execution on cloud side:** Let $C^C_i$ be the computing capacity assigned from the cloud, which is assumed to be more powerful than that of the mobile device, namely $C^C_i > C^M_i$. Therefore, the task execution time $t^{c}_{ij}$ on the cloud side can be calculated as follows:

$$ t^{c}_{ij} = \frac{\omega_0}{C^C_i} \cdot P^c_{ij}. $$

(9)

During this process, the mobile device stays idle with the standby power consumption of $P^s_i$. Therefore, the mobile device's standby energy consumption $e^{s}_{ij}$ is calculated as follows:

$$ e^{s}_{ij} = \frac{\omega_0}{C^C_i} \cdot P^s_i. $$

(10)

**Input data sending:** If offloading is necessary, the mobile device should send the selected tasks to the cloud before the execution begins. Thus, with the input data for transmission $\eta_{ij}$, the transmission time $t^{s}_{ij}$ can be calculated as follows:

$$ t^{s}_{ij} = \frac{\eta_{ij}}{R_i}. $$

(11)

Let $P^s_i$ be the data transmission power consumption. Further, in practice, the input data transmission over cellular networks consumes the tail energy after transferring data for a few seconds [37]. Let $P^s_i$ be the cellular tail power consumption of mobile device $i$. Let $t^s_{ij}$ be its tail time. The total energy consumption of the data transmission $e^{s}_{ij}$ can be expressed as follows:

$$ e^{s}_{ij} = \eta_{ij}P^s_i + P^s_i t^s_{ij}. $$

(12)

**Output data receiving:** In many mobile applications, as the size of the output data is often considerably smaller than that of the input data, we choose to ignore the output data receiving process for all mobile devices, as in some of the previous studies [8,11, 24]. For example, the typical input dataset size for a virus scanning task could be hundreds of kilobytes to tens of megabytes, including the system settings, program codes, and data files. In contrast, the output dataset size could be just a few bytes, including the information regarding whether the data file is infected by a virus or not.

4.2. Energy saving optimization problem formulation

In this section, we formulate the energy consumption of the task offloading process for each mobile device as a 0–1 nonlinear integer programming problem. The energy consumption of a mobile device can be divided into two categories:

• If the task is executed locally on the mobile device, only the task execution process consumes energy $e^{o}_{ij}$.

• If the task is offloaded onto the cloud server through the wireless access point, the energy consumption $e^{c}_{ij}$ consists of two parts: the input data sending energy consumption and the standby energy consumption.

Thus, the offloading energy consumption $e^{o}_{ij}$ can be expressed as follows:

$$ e^{o}_{ij} = \eta_{ij}P^s_i + \frac{\omega_0}{C^C_i} \cdot P^c_{ij}. $$

(13)

The energy consumption of mobile device $\theta_i$ is calculated as follows:

$$ \theta_i = \sum_{j=1}^{M_i} \rho_{ij}e^{o}_{ij} + \sum_{j=1}^{M_i} (1 - \rho_{ij})e^{s}_{ij}. $$

(14)

In order to guarantee a fast and relatively less error-prone mobile cloud service, we need to consider the time constraint and the transmission error rate constraint.

**Time constraint:** For mobile device $i$, we set $T^{m}_{i}$ as the task execution time on the mobile device and $T^c_{ij}$ as the task transmission and execution time with the cloud service.

$$ T^{m}_{i} = \sum_{j=1}^{M_i} \rho_{ij}T^{m}_{ij} $$

(15)

$$ T^c_{ij} = (1 - \rho_{ij})(t^{s}_{ij} + t^{c}_{ij}). $$

(16)

As mobile cloud users cannot tolerate a considerably long running latency, the total completion time of the application execution should be less than the time deadline $T^d_i$, i.e., $T^m_i \leq T^d_i$ and $T^c_{ij} \leq T^d_i$.

**Transmission error rate constraint:** As the transmission contains a number of parallel tasks, the error rate of the complete data sending process can be expressed as follows:

$$ \Gamma_i = 1 - \prod_{j=1}^{M_i} (1 - (1 - \rho_{ij})\epsilon_i). $$

(17)

In the flow path of task offloading, the data transmission should meet the error rate constraints, i.e., $\Gamma_i \leq \epsilon_i$, where $\epsilon_i$ implies the maximum error rate of input sending that the mobile device can tolerate, $0 < \epsilon_i < 1$.

To minimize the energy consumption of task execution for each mobile device, the constrained optimization problem can be formulated as follows:

$$ \min_{\rho_{ij}} \theta_i = \sum_{j=1}^{M_i} \rho_{ij}e^{o}_{ij} + \sum_{j=1}^{M_i} (1 - \rho_{ij})e^{s}_{ij} $$

(18)

s.t. $ \sum_{j=1}^{M_i} \rho_{ij}T^{m}_{ij} \leq T^d_i $  

(19)

$ \sum_{j=1}^{M_i} (1 - \rho_{ij})(t^{s}_{ij} + t^{c}_{ij}) \leq T^d_i $  

(20)

$ 1 - \prod_{j=1}^{M_i} (1 - (1 - \rho_{ij})\epsilon_i) \leq \epsilon_i $  

(21)

$ \rho_{ij} \in \{0, 1\}. $  

(22)

This issue is a 0–1 nonlinear integer programming problem. The non-linearity of the objective function and the constraint condition make this problem difficult to solve [38]. In this case, we can enumerate all possible $\{|\rho_{ij}\}$, choosing the set that has the minimal energy consumption. Although determining its solution space is not difficult as in the case of a continuous optimization problem, this brute-force search will lead to the complexity of $O(2^{M_i})$. Therefore, we can infer that this constrained optimization problem has the NP-complexity characteristic. Hence, a highly efficient algorithm with combined methods is proposed to reduce its complexity and reveal near-optimal offloading decisions.

5. Solution to task offloading method

To solve this NP-complexity problem, an efficient task offloading decision-making method is proposed. With the 0–1 assignment variable relaxation, this problem can be converted into a continuous optimization problem and can then be solved effectively for the offloading priority by using the standard convex optimization
method. Because of the coupled terms in the priority expression, an iterative decoupling method is investigated next for making near-optimal offloading decisions.

5.1. 0–1 assignment variable relaxation

To begin with, it is worth noting that some tasks on mobile devices may consume more energy with the task offloading policy. That is, the mobile device may have no interest in offloading these tasks to the cloud for energy saving. Thus, we could divide tasks into two categories for each mobile device: (i) tasks that could save energy through task offloading in set \( L_i \), and (ii) tasks that always consume more energy with offloading in set \( K_i \).

\[
\begin{align*}
\text{if} \quad & e_{ij}^0 \geq e_{ij}^m, \quad \text{let} \quad j \in L_i, \\
\text{if} \quad & e_{ij}^0 < e_{ij}^m, \quad \text{let} \quad j \in K_i. 
\end{align*}
\]  

(23)

Task \( j \in L_i \) has to be executed locally on mobile device \( i \); therefore, the offloading decision \( \rho_{ij} = 1 \) always holds. Further, for task \( j \in K_i \), the assignment variable mapping function is leveraged for decision making. As mentioned above, \( \rho_{ij} \) denotes discrete integer parameters that make the problem difficult to solve. For complexity reduction, we assume that \( \rho_{ij} \) is a continuous parameter in a real domain.

\[
\rho_{ij} \in [0, 1], \quad j \in K_i. 
\]  

(24)

It is rational to consider \( \rho_{ij} \) to be the priority of tasks for offloading. The smaller the parameter \( \rho_{ij} \), the higher is the priority of task \( j \) for offloading on mobile device \( i \). Therefore, the constrained 0–1 nonlinear integer programming problem defined in (18)–(22) can be converted as follows:

\[
\begin{align*}
\min_{\rho_{ij}} & \, \Theta_i = \sum_{j \in K_i} \rho_{ij} e_{ij}^m + \sum_{j \in L_i} (1 - \rho_{ij}) e_{ij}^0 + \sum_{j \in L_i} e_{ij}^m \\
n\text{s.t.} & \, \sum_{j \in K_i} \rho_{ij} t_{ij}^m + \sum_{j \in L_i} t_{ij}^d \leq T_i^d \\
& \, \sum_{j \in K_i} (1 - \rho_{ij}) (t_{ij}^m + t_{ij}^d) \leq T_i^d \\
& \, 1 - \sum_{j \in K_i} (1 - (1 - \rho_{ij}) \varepsilon_i) \leq \varepsilon_i \\
& \, \rho_{ij} \in [0, 1], \quad j \in K_i \\
& \, \rho_{ij} = 1, \quad j \in L_i. 
\end{align*}
\]  

(25–30)

It is a continuous optimization problem. Therefore, standard optimization methods are used for solving this issue.

5.2. Convex optimization-based offloading priority determination method

Now, we will prove that the continuous optimization issue is a standard convex optimization problem.

**Lemma 1.** The continuous optimization problem defined in (25)–(30) has a convex property.

Proof of Lemma 1 is given in Appendix A.

Therefore, the transformed optimization problem can be solved systematically with convex optimization techniques [39]. The Lagrange duality is leveraged, and the Lagrangian function can be expressed as follows:

\[
L(\rho_{ij}, u, v, k) = \Theta_i + u_i (T_i^m - T_i^d) - v_i (T_i^d - T_i^m) - k_i (\varepsilon_i - \Gamma_i),
\]  

(31)

where \( u_i, v_i, \) and \( k_i \) denote the Lagrangian multipliers for constraints (26)–(28). Therefore, the derivative of the Lagrange function can be expressed as follows:

\[
\frac{\partial L(\rho_{ij}, u, v, k)}{\partial \rho_{ij}} = e_{ij}^m - e_{ij}^0 + u_i t_{ij}^m - v_i (t_{ij}^m + t_{ij}^d) - k_i \varepsilon_i \sum_{r \neq j} (1 - (1 - \rho_{ij}) \varepsilon_i) = 0, \quad j \in K_i.
\]  

(32)

The Karush–Kuhn–Tucker (KKT) conditions indicate the following:

\[
\frac{\partial L(\rho_{ij}, u, v, k)}{\partial \rho_{ij}} = 0, 
\]  

(33)

\[
u_i(T_i^m - T_i^d) = 0, 
\]  

(34)

\[
u_i(T_i^d - T_i^m) = 0, 
\]  

(35)

\[
k_i \left( \varepsilon_i - 1 + \sum_{j=1}^{M} (1 - (1 - \rho_{ij}) \varepsilon_i) \right) = 0, 
\]  

(36)

\[
u_i \geq 0, \quad v_i \geq 0, \quad k_i \geq 0, \quad \forall j \in K_i.
\]  

(37)

**Lemma 2.** The optimal Lagrangian multipliers \( u_i, v_i, \) and \( k_i \) must satisfy \( u_i = 0, \) \( v_i = 0, \) and \( k_i > 0; \) \( u_i = 0, \) \( v_i > 0, \) and \( k_i = 0; \) or \( u_i = 0, \) \( v_i > 0, \) and \( k_i > 0. \)

Proof of Lemma 2 is given in Appendix B.

Assuming that \( v_i \) and \( k_i \) are already known, we can derive all unknown \( \rho_{ij} \) through the same number of equations as in (32). However, this issue is also difficult to solve. We find that the ratio of \( \rho_{ij} / \rho_{i'j} \) for any pair of tasks \( j \) and \( j' \) can also reflect the priority of the task for offloading. Thus, we choose to compute this ratio. From (32) and Lemma 1, we have the following:

\[
\frac{\partial L}{\partial \rho_{ij}} = e_{ij}^m - e_{ij}^0 - v_i (t_{ij}^m + t_{ij}^d) - k_i \varepsilon_i \sum_{r \neq j} (1 - (1 - \rho_{ij}) \varepsilon_i) = 0, 
\]  

(38)

\[
\frac{\partial L}{\partial \rho_{i'j'}} = e_{i'j'}^m - e_{i'j'}^0 - v_i (t_{i'j'}^m + t_{i'j'}^d) - k_i \varepsilon_i \sum_{r \neq j'} (1 - (1 - \rho_{i'j'}) \varepsilon_i) = 0. 
\]  

(39)

By converting (38) and (39), we obtain the following:

\[
k_i \varepsilon_i^2 \rho_{ij} = \frac{e_{ij}^m - e_{ij}^0 - v_i (t_{ij}^m + t_{ij}^d) - k_i \varepsilon_i (1 - \varepsilon_i)}{(1 - (1 - \rho_{ij}) \varepsilon_i)}, \quad j \in K_i \\
k_i \varepsilon_i^2 \rho_{i'j'} = \frac{e_{i'j'}^m - e_{i'j'}^0 - v_i (t_{i'j'}^m + t_{i'j'}^d) - k_i \varepsilon_i (1 - \varepsilon_i)}{(1 - (1 - \rho_{i'j'}) \varepsilon_i)}. 
\]  

(40–41)

Thus, the ratio \( \rho_{ij} / \rho_{i'j'} \) can be expressed as follows:

\[
\rho_{ij} = \frac{e_{ij}^m - e_{ij}^0 - v_i (t_{ij}^m + t_{ij}^d) - k_i \varepsilon_i (1 - \varepsilon_i)}{(1 - (1 - \rho_{ij}) \varepsilon_i)} \sum_{r \neq j} \frac{1}{(1 - (1 - \rho_{ij}) \varepsilon_i)}, \quad j \in K_i \\
\rho_{i'j'} = \frac{e_{i'j'}^m - e_{i'j'}^0 - v_i (t_{i'j'}^m + t_{i'j'}^d) - k_i \varepsilon_i (1 - \varepsilon_i)}{(1 - (1 - \rho_{i'j'}) \varepsilon_i)} \sum_{r \neq j'} \frac{1}{(1 - (1 - \rho_{i'j'}) \varepsilon_i)}. 
\]  

(42)

We define \( \psi_{ij} \) as the offloading priority of task \( j \) for mobile device \( i \). By analyzing (42) above, we can calculate \( \psi_{ij} \) as follows:

\[
\psi_{ij} = \frac{1}{e_{ij}^m - e_{ij}^0 - v_i (t_{ij}^m + t_{ij}^d) - k_i \varepsilon_i (1 - \varepsilon_i)} \sum_{r \neq j} \frac{1}{(1 - (1 - \rho_{ij}) \varepsilon_i)}. 
\]  

(43)
If only two tasks \( j \) and \( j' \) are compared at any point of time (i.e., the other tasks are not taken into consideration), the offloading priority \( \Psi_{ij} \) in (43) can be simplified as follows:

\[
\Psi_{ij} = \frac{1}{e^{\xi_j} - e^{\xi_j} - \psi_i(t_i^j + t_j^i)} - k_i e_i (1 - e_i).
\] (44)

As for every task in mobile device \( i \), \( k_i e_i (1 - e_i) \) remains the same. Therefore, it can be omitted from \( \Psi_{ij} \). Then, \( \Psi_{ij} \) in (44) can be simplified as follows:

\[
\Psi_{ij} = \frac{1}{e^{\xi_j} - e^{\xi_j} - \psi_i(t_i^j + t_j^i)}.
\] (45)

By sorting \( \Psi_{ij} \) in the descending order, we obtain the offloading priority for each task. Finally, by using the following optimal Lagrangian multiplier bisection searching algorithm, we investigate the optimal Lagrangian multipliers \( v_i \) and determine which tasks should be selected for offloading in the meanwhile.

**Algorithm 1 Optimal Lagrangian Multiplier Bisection Searching Algorithm**

**Input**: The offloading priority \( \Psi_{ij} \), a tolerance number \( \epsilon \).

**Output**: The Lagrangian Multiplier \( v_i \), the number of tasks that should be executed on mobile devices \( q_i \), the offloading decision \( \rho_i \).

1. Initialize \( q_i = 0 \), set tolerance \( \epsilon \) a sufficiently small positive number, initialize \( v_{\min} = 0 \) and \( v_{\max} \) as in (47);
2. while \((v_{\max} - v_{\min}) \leq \epsilon \) do
3. Set \( v_i = \frac{(v_{\min} + v_{\max})}{2} \), calculate \( \Psi_{ij} \);
4. for \( q_i = 0 \) to StrLen(\( K_i \)) do
5. Set \( q_i \) numbers of \( \rho_i \) with bigger \( \Psi_{ij} \) as 1;
6. if \( \Gamma_i > \psi_i \) and \( \& \& \xi^m \leq T^c_i \) then
7. continue;
8. else if \( \Gamma_i \leq \psi_i \) and \( \& \& \xi^m > T^c_i \) then
9. Let \( q_i = q_i - 1 \) and break;
10. else
11. break;
12. end if
13. end for
14. Set \( q_i \) numbers of \( \rho_i \) with bigger \( \Psi_{ij} \) as 1;
15. if \( T^c_j > T^c_j \) then
16. Let \( v_{\min} = v_i \);
17. else if \( T^c_j < T^c_j \) then
18. Let \( v_{\max} = v_i \);
19. end if
20. end while

First, we set the searching bounds \( v_{\min} \) and \( v_{\max} \) to guarantee that the optimal \( v_i \) can be found at an acceptable cost. \( v_{\min} \) can be set to 0, and \( v_{\max} \) should be set as a sufficiently large number so that the offloading priority is determined only by \( t_i^j + t_j^i \). We sort \( t_i^j + t_j^i \) for all tasks in the descending order. Therefore, for any task \( j \in K_i \), it must satisfy the following:

\[
e^{\xi_j} - e^{\xi_j} - v_{\max}(t_i^j + t_j^i)
\leq e^{\xi_j} - e^{\xi_j} - v_{\max}(t_i^{j-1} + t_j^{j-1}).
\] (46)

That is, \( v_{\max} \) can be expressed as follows:

\[
v_{\max} \geq \max \left( \frac{e^{\xi_j} - e^{\xi_j} - e^{\xi_j} - e^{\xi_j}}{t_i^{j-1} + t_j^{j-1} + t_i^{j-1} + t_j^{j-1}} \right), \quad \forall j \in K_i.
\] (47)

Guided by the bisection searching method, we let \( v_i = \frac{(v_{\min} + v_{\max})}{2} \). The offloading priority \( \Psi_{ij} \) in (45) is calculated and sorted in the descending order. Then, we define \( q_i \) as the number of tasks that should be executed on the mobile device. Thus, \( q_i \) number of \( \rho_i \) with bigger \( \Psi_{ij} \) should be set as 1.

The optimal \( q_i \) can be determined by gradually increasing it until constraints (26)–(28) are satisfied. Further, in Algorithm 1, if \( T^c_i < T^c_j \), we let \( v_{\max} = v_i \); that is, \( T^c_i \) increases along with \( v_i \). In contrast, if \( T^c_i > T^c_j \), we let \( v_{\min} = v_j \) so that \( T^c_j \) decreases along with \( v_j \). Then, we set a sufficiently small positive number \( x \) to determine the end of an iteration. Finally, the bisection search terminates when \((v_{\max} - v_{\min}) \leq \epsilon \) is satisfied; it reveals optimal results when \( T^c_i = T^c_j \). The computation time complexity of the Lagrangian multiplier bisection searching algorithm is \( M_i \log_2 (v_{\max} - v_{\min})/x \).

5.3. Iterative decoupling algorithm and analysis

With the convex optimization method discussed above, the offloading priority \( \Psi_{ij} \) with coupled terms can be derived. As can be seen in (2), (3), (5), and (13), the offloading energy consumption \( e^{\xi_j} \) and BER \( \xi_j \) are \( \xi_j \)-related functions. Further, \( \xi_j \) is determined by the offloading decisions \( \{\rho_i\} \) of mobile device \( i \). Therefore, the offloading priority can still not be obtained directly because of the coupled terms \( \xi_i \).

**Algorithm 2 Iterative Decoupling Algorithm for Offloading Decisions**

**Input**: \( \tilde{\epsilon}_j, \tilde{\xi}_j, \epsilon_j \).

**Output**: Offloading decision \( \{\rho_i\} \).

1. Initialize \( \{\rho_i\} = 0, i \in K_i, \xi_i = 1 \).
2. repeat
3. for mobile device \( i = 1 \) to \( N \) do
4. Set \( q_i \) number of \( \rho_i \) with bigger \( \Psi_{ij} \) as 1;
5. Invoke Algorithm 1 for \( v_i \) and \( \{\rho_i\} \);
6. Update sets \( L_i \) and \( K_i \);
7. Update \( \xi_i \) as in (5), update \( e^{\xi_j} \) as in (13);
8. Update \( \xi_i \) as in (1) according to current \( \{\rho_i\} \);
9. Update \( \gamma_i \) as in (3);
10. end for
11. until \( \xi_i \) stop changing in the next iteration

Therefore, for decoupling, we initialize \( \{\xi_i\} \). Temporary 0–1 offloading decisions \( \{\rho_i\} \) can be made according to Algorithm 1. Then, \( \{\xi_i\} \) can be updated as in (1) on the basis of the temporary \( \{\rho_i\} \). We iterate the entire process several times. Ultimately, when \( \{\xi_i\} \) remains unchanged, the iteration comes to an end and reveals the final offloading decisions \( \{\xi_i\} \), whose near-optimality can be proved. Therefore, the iterative decoupling algorithm for offloading decision-making is proposed as above.

As can be seen in Algorithm 2, we initialize the iteration by setting all \( \rho_i = 0, j \in K_i \), i.e., \( \xi_i = 1 \). Under this condition, task offloading through the wireless channel is in the worst case when all mobile devices choose to offload tasks. The worst SNR can be calculated as follows:

\[
\gamma_i = 10 \log_{10} \left( \frac{P_i H_i}{\sigma^2 + \sum_{r=1, r \neq i} P_r H_r} \right).
\] (48)

Further, the initial BER and the initial offloading energy consumption can be expressed as follows:

\[
\tilde{\xi}_i = Q(\sqrt{2 \gamma_i}),
\]

\[
\tilde{e}^{\xi_j} = \frac{H_i P_i^j}{W \log_2 (1 + \gamma_i)} + \frac{\omega_{ik} P_r j^j}{C_i}.
\] (50)

With the initial state \( \gamma_i \) and \( \tilde{e}^{\xi_j} \), we calculate the Lagrangian multiplier \( v_i \) and the offloading decision \( \{\rho_i\} \) by using Algorithm 1 for mobile device \( i \). The binary variable service flag \( \{\xi_i\} \) is updated.
then. We repeat the process until $\{\xi_j\}$ remains unchanged in the iterations. Finally, we prove that the near-optimal offloading decisions $\{p_{i,j}\}$ to the relaxed 0–1 nonlinear integer programming problem can be derived from the proposed iterative decoupling algorithm.

**Lemma 3.** The proposed iterative decoupling algorithm reveals near-optimal offloading decisions $\{p_{i,j}\}$ to the relaxed 0–1 nonlinear integer programming problem.

Proof of **Lemma 3** is given in Appendix C.

According to the above analysis, the computation time complexity of Algorithm 1 is $M_i \cdot \log_2((v_i \cdot \max - v_i \cdot \min)/\lambda)$ for mobile device $i$. Therefore, the algorithm takes up to $N \cdot M_i + N \cdot M_j \cdot \log_2((v_i \cdot \max - v_i \cdot \min)/\lambda)$ iterations for all mobile devices. Further, Algorithm 2 needs up to $M_i$ iterations to converge for mobile device $i$. To sum up, Algorithm 2 provides a near-optimal solution to the energy-saving task offloading method. Its total computation complexity in the entire mobile cloud environment is $O(N \cdot M_i^3)$.

### 6. Numerical simulations

In this section, we describe our experimental setup of the mobile cloud offloading scenario. To begin with, we detail the specific quantities and parameters associated with the 10 tasks for execution. Further, the processing densities and the data sizes of various tasks are presented in Table 4, which are used in several studies [15,27,40]. Obviously, face recognition and virus scanning are the computation-intensive tasks among them. All 10 tasks are deployed on four types of smartphones for execution. On the basis of these various processing densities, the allocated computing capacities $C^*_{i,j}$ are specified for the simulation.

### 6.1. Simulation setup and real-world datasets

**Mobile cloud computing scenario.** First, the mobile cloud computing scenario is constructed over a hexagonal cellular network with radius 2 km. We assume that the wireless access point is located at the center of the hexagonal cell. All mobile devices are randomly deployed around the wireless access point.

**Wireless communication model.** Then, for the wireless communication model, we consider the cellular network scenario, setting the channel bandwidth $W = 2$ MHz and the power of background noise $\sigma^2 = -100$ dBm. According to the physical interference model, the path loss factor $a = 2$. For the availability of a wireless fading connection, the SINR coding gain is set as 16 dB for this simulation scenario.

**Real energy consumption and computing parameters.** Without any loss of generality, we consider four types of smartphones, namely Galaxy Note, Galaxy Note 2, Nexus S, and Galaxy Nexus. However, although Galaxy Note, Galaxy Note 2, and Galaxy Nexus have multi-core processors, for a fair comparison with Nexus S, we assume that all four types of smartphones work in the single-core mode. All 10 tasks are executed on the working single core. The realistic parameters of the computing and communication energy models, including data sending power $P_i^t$, cellular tail power $P_i^\tau$, its tail time $T_i^\tau$, for a cellular network, idle power $P_i^i$, CPU processing parameters $\chi_i$, $\alpha_i$, and $\beta_i$ are specified in Table 2. All the energy consumption parameters are adopted as in [36], which are measured at various clock speeds and in the cellular network scenarios by using a Monsoon power monitor.

Moreover, the computing capacity assigned from the cloud for each mobile device $C_i^C$ is 5 GHz. The transmission error rate constraint is $\eta_i = 8\%$. Besides, we assume the system deadlines $T_i^d$ should not exceed the time delay when all tasks are executed locally on the smartphone. So the time deadlines $T_i^d$ are set to 167 s, 145 s, 240 s, and 197 s, respectively. In particular, in the simulation, the type of mobile device in the mobile cloud computing scenario is randomly selected among them. Moreover, Table 3 lists the computing parameters of each smartphone, including the computing capacity $C_i$, the maximum computing capacity $C_i^\max$, and the topmost clock frequency $C_i^\tau$. For energy saving of mobile devices, DVFS techniques are adopted for various tasks. Typically, the DVFS adjusts the clock frequency in six steps. That is, the DVFS program adjusts the frequency to fixed 180 MHz, 200 MHz, 140 MHz, and 160 MHz each time for the four types of smartphones considered, respectively.

**Task set description.** As shown in Table 4, we consider 10 tasks for execution. Further, the processing densities and the data sizes of various tasks are presented in Table 4, which are used in several studies [15,27,40]. Obviously, face recognition and virus scanning are the computation-intensive tasks among them. All 10 tasks are deployed on four types of smartphones for execution. On the basis of these various processing densities, the allocated computing capacities $C^*_{i,j}$ are specified for the simulation.

### 6.2. Performance evaluation of proposed task offloading method with 30 mobile devices

To begin with, we conduct one case of simulation with the proposed IDA algorithm in a given scenario. As can be seen in Fig. 3, all 30 mobile devices are randomly distributed over the hexagonal cellular network. Table 5 provides the specific location of each mobile device and its corresponding device type. We let 1, 2, 3,
and 4 represent Galaxy Note, Galaxy Note 2, Nexus S, and Galaxy Nexus, respectively. For the four types of mobile devices specified in Table 3, if all tasks are executed locally on the mobile device, the local energy consumption is 109.17 J, 140.28 J, 131.51 J, and 140.54 J, with a time delay of 166.15 s, 144.11 s, 239.75 s, and 196.22 s, respectively.

In the mobile cloud computing simulation scenario discussed above, we implement the proposed IDA algorithm to select the tasks for energy saving, making certain offloading decisions. As can be seen in Fig. 3, 23 among the 30 smartphones, i.e., smartphones 1, 2, 4, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 20, 24, 25, 26, 27, 28, 29, and 30, offload tasks to the cloud server. The specific 0–1 task offloading decisions \( \{ \rho_{ij} \} \) are listed in Table 5. By analyzing \( \{ \rho_{ij} \} \), we find out that the offloading decisions are mainly determined by the task characteristics. In general, a mobile device prefers to offload computation-intensive tasks for energy saving, which also conforms to our common knowledge and experience. For example, on smartphones 10, 12, and 14, if the computation-intensive tasks such as face recognition and virus scanning are offloaded, a significant amount of energy and time can be saved.

With local execution, the smartphones need 130.375 J and 186.56 s to complete all tasks on average. In contrast, the smartphones consume 82.606 J and 148.614 s on average with the proposed IDA algorithm. This implies that the proposed IDA algorithm can simultaneously save a considerable amount of task execution energy and time.

### 6.3 Sensitivity Analysis

In this section, the sensitivity analysis of the simulation parameters is performed. As in the numerical simulations, 4 sensitive parameters, i.e., the size of the area, cloud offloading time deadline, data size and workload density may play important roles in influencing the offloading performance. To investigate which factors have the most influence on the decision of task offloading, the quantitative analysis of each simulation parameter is adopted then. The **size of the area**. With the 30 smartphones simulation scenario described in Section 6.2, the quantitative analysis of network sizes is deployed. We let the data size, time deadline and workload density remain unchanged, then gradually increase the network radius from 1 to 5.5 km. All 30 smartphones are evenly distributed. As can be seen in Eq. (12) and Table 2, the tail power accounts for much of the total data transmission energy consumption when the wireless network is not crowded. The data transmission energy consumption increases slowly along with the increment of

### Table 5

Simulation results for 30-devices in A 50 × 50 m network.

<table>
<thead>
<tr>
<th>Device</th>
<th>Location (m)</th>
<th>Type</th>
<th>0–1 offloading decision ( { \rho_{ij} } )</th>
<th>Energy (J)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(122, 1439)</td>
<td>1</td>
<td>{1 0 1 0 1 0 1 1 1 1 1}</td>
<td>56.61</td>
<td>116.45</td>
</tr>
<tr>
<td>2</td>
<td>(712, 1223)</td>
<td>2</td>
<td>{0 1 0 1 0 1 1 1 1 1 1}</td>
<td>62.15</td>
<td>138.61</td>
</tr>
<tr>
<td>3</td>
<td>(−1734, 166)</td>
<td>3</td>
<td>{1 1 1 1 1 1 1 1 1 1 1}</td>
<td>140.54</td>
<td>196.22</td>
</tr>
<tr>
<td>4</td>
<td>(−873, −76)</td>
<td>4</td>
<td>{1 1 1 1 1 1 1 1 1 1 1}</td>
<td>116.45</td>
<td>138.61</td>
</tr>
<tr>
<td>5</td>
<td>(739, −167)</td>
<td>5</td>
<td>{1 1 1 1 1 1 1 1 1 1 1}</td>
<td>140.54</td>
<td>196.22</td>
</tr>
<tr>
<td>6</td>
<td>(433, −695)</td>
<td>6</td>
<td>{1 1 1 1 1 1 1 1 1 1 1}</td>
<td>82.46</td>
<td>138.61</td>
</tr>
<tr>
<td>7</td>
<td>(−182, 676)</td>
<td>7</td>
<td>{0 0 1 0 1 1 1 1 1 1 1}</td>
<td>54.55</td>
<td>116.45</td>
</tr>
<tr>
<td>8</td>
<td>(1325, 1161)</td>
<td>8</td>
<td>{1 1 1 1 1 1 1 1 1 1 1}</td>
<td>82.46</td>
<td>116.45</td>
</tr>
<tr>
<td>9</td>
<td>(851, −110)</td>
<td>9</td>
<td>{1 1 1 1 1 1 1 1 1 1 1}</td>
<td>82.46</td>
<td>116.45</td>
</tr>
<tr>
<td>10</td>
<td>(23, −780)</td>
<td>10</td>
<td>{0 0 1 0 1 1 1 1 1 1 1}</td>
<td>54.55</td>
<td>116.45</td>
</tr>
<tr>
<td>11</td>
<td>(1159, −1055)</td>
<td>11</td>
<td>{1 1 1 1 1 1 1 1 1 1 1}</td>
<td>140.54</td>
<td>196.22</td>
</tr>
<tr>
<td>12</td>
<td>(−1063, −141)</td>
<td>12</td>
<td>{1 1 1 1 1 1 1 1 1 1 1}</td>
<td>82.46</td>
<td>116.45</td>
</tr>
<tr>
<td>13</td>
<td>(478, 461)</td>
<td>13</td>
<td>{1 1 1 1 1 1 1 1 1 1 1}</td>
<td>82.46</td>
<td>116.45</td>
</tr>
<tr>
<td>14</td>
<td>(−862, 943)</td>
<td>14</td>
<td>{1 1 1 1 1 1 1 1 1 1 1}</td>
<td>82.46</td>
<td>116.45</td>
</tr>
<tr>
<td>15</td>
<td>(−355, 1316)</td>
<td>15</td>
<td>{1 1 1 1 1 1 1 1 1 1 1}</td>
<td>82.46</td>
<td>116.45</td>
</tr>
<tr>
<td>16</td>
<td>(1741, −404)</td>
<td>16</td>
<td>{1 1 1 1 1 1 1 1 1 1 1}</td>
<td>82.46</td>
<td>116.45</td>
</tr>
<tr>
<td>17</td>
<td>(−1791, 85)</td>
<td>17</td>
<td>{0 0 1 0 1 1 1 1 1 1 1}</td>
<td>54.55</td>
<td>116.45</td>
</tr>
<tr>
<td>18</td>
<td>(991, −719)</td>
<td>18</td>
<td>{0 0 1 0 1 1 1 1 1 1 1}</td>
<td>54.55</td>
<td>116.45</td>
</tr>
<tr>
<td>19</td>
<td>(−28, −1113)</td>
<td>19</td>
<td>{1 1 1 1 1 1 1 1 1 1 1}</td>
<td>140.54</td>
<td>196.22</td>
</tr>
<tr>
<td>20</td>
<td>(1757, −71)</td>
<td>20</td>
<td>{1 1 1 1 1 1 1 1 1 1 1}</td>
<td>56.61</td>
<td>116.45</td>
</tr>
<tr>
<td>21</td>
<td>(160, −1116)</td>
<td>21</td>
<td>{1 1 1 1 1 1 1 1 1 1 1}</td>
<td>140.54</td>
<td>196.22</td>
</tr>
<tr>
<td>22</td>
<td>(1278, 1086)</td>
<td>22</td>
<td>{1 1 1 1 1 1 1 1 1 1 1}</td>
<td>140.54</td>
<td>196.22</td>
</tr>
<tr>
<td>23</td>
<td>(737, 627)</td>
<td>23</td>
<td>{1 1 1 1 1 1 1 1 1 1 1}</td>
<td>140.54</td>
<td>196.22</td>
</tr>
<tr>
<td>24</td>
<td>(−303, −40)</td>
<td>24</td>
<td>{1 1 1 1 1 1 1 1 1 1 1}</td>
<td>56.61</td>
<td>116.45</td>
</tr>
<tr>
<td>25</td>
<td>(334, −1667)</td>
<td>25</td>
<td>{0 0 1 0 1 1 1 1 1 1 1}</td>
<td>54.55</td>
<td>116.45</td>
</tr>
<tr>
<td>26</td>
<td>(227, 848)</td>
<td>26</td>
<td>{0 0 1 0 1 1 1 1 1 1 1}</td>
<td>54.55</td>
<td>116.45</td>
</tr>
<tr>
<td>27</td>
<td>(−48, 470)</td>
<td>27</td>
<td>{1 1 1 1 1 1 1 1 1 1 1}</td>
<td>82.46</td>
<td>116.45</td>
</tr>
<tr>
<td>28</td>
<td>(−1145, 583)</td>
<td>28</td>
<td>{1 1 1 1 1 1 1 1 1 1 1}</td>
<td>56.61</td>
<td>116.45</td>
</tr>
<tr>
<td>29</td>
<td>(−477, −1583)</td>
<td>29</td>
<td>{1 1 1 1 1 1 1 1 1 1 1}</td>
<td>82.46</td>
<td>116.45</td>
</tr>
<tr>
<td>30</td>
<td>(−490, −949)</td>
<td>30</td>
<td>{1 1 1 1 1 1 1 1 1 1 1}</td>
<td>56.61</td>
<td>116.45</td>
</tr>
</tbody>
</table>
network size, which means the network size plays a secondary role in influencing the offloading performance.

Cloud offloading time deadline. Originally in Section 6.1, we set the cloud offloading time deadline as 167 s, 145 s, 240 s and 197 s. To evaluate the sensitivity of the deadline constraints, the time deadlines are reset in different steps. As can be seen in Fig. 4(b), the deadline is added with 25 s in each step. At the beginning with harsh cloud offloading deadline constraints, almost no smartphones satisfy the offloading deadline constraints. Thus all tasks on these smartphones need to be executed locally, which cause more energy consumption 129.71 J and time delay 187.65 s. Then with the relaxation of deadline constraints, the energy consumption and time delay dramatically reduced as more tasks can be offloaded. Finally, the energy consumption and time delay reached a stable level at 82.61 J and 148.61 s.

Data size. As described in Section 6.1, the real-world data sizes are 60, 2048, 400, 300, 200, 200, 200, 200, 10, 240 KB. Then for quantitative analysis, the data size is varied with 20 KB in each step. As can be seen in Fig. 4(c), with the increase of data size, the energy consumption and time delay of smartphones increase linearly from 70.89 J, 130.76 s to 105.25 J, 187.72 s in the first 8 steps. The number of offloaded smartphones and tasks stay around 23 and 60. Afterwards, with the further increment of data size, some tasks become unfit for offloading. The number of offloaded smartphones and tasks decline to 15 and 45. Therefore, the energy consumption and time delay increase suddenly to 128.70 J and 217.35 s. The data size influences the performance of task offloading.

Workload density. At the beginning, the workload densities of 10 tasks are set as 31,520, 2480, 1744, 2480, 32,786, 247.8, 423, 1920, 8090 and 360 cycles/bit respectively. To evaluate the influence of workload density on offloading performance, the former 5 denser workloads, i.e., 31,520, 2480, 2480, 32,786, 8090, are added with 40 cycles/bit in each step. On the contrary, the other 5 workloads, i.e., 1744, 247.8, 423, 1920, 360 are subtracted with 40 cycles/bit in each step. So the total workload density remains unchanged. At step 5, the workload densities are 31,680, 2640, 1584, 2640, 32,946, 87.8, 263, 1760, 8250 and 200 cycles/bit, just as described in Section 6.1. During the variation of workload density, the average number of smartphones with cloud offloading service remains around 23, but the total number of offloaded tasks declines from 59 to 52. Because the more denser the workload is, the more likely it will be selected for energy saving and time delay reducing. Due to the constraints of the offloading problem, less number of tasks will be offloaded if denser workloads are selected. As can be seen in Fig. 4(d), the energy consumption and time delay of smartphones decline steadily from 89.26 J, 167.11 s to 72.93 J, 121.23 s, which means the workload density plays an important role in influencing the offloading performance.

6.4. Statistical performance evaluation of proposed task offloading method over 1000 simulations

To further evaluate the statistical performance of the proposed IDA algorithm, we conduct a series of simulations, increasing the number of smartphones from 5 to 50 for the verification. Further, since smartphones are randomly deployed, the mobile network topology has stochastic features. Therefore, we take the average value of 1000 simulations to evaluate the statistical performance of the proposed task offloading policy equitably. Moreover, to verify the superiority of the proposed IDA method for energy saving, two other cloud offloading methods are proposed for a fair comparison:

- All task selection algorithm (ATSA): In ATSA, the smartphones are selected to access a cloud service similar to the proposed IDA method. However, if the time deadline and the transmission error rate constraints are satisfied, all tasks on the smartphones are offloaded onto the cloud side. In contrast, if these constraints are not satisfied, all tasks are executed locally on the mobile devices. The computation complexity is \(O(N \cdot M)\) for all smartphones in the mobile cloud environment.

- Random task selection algorithm (RTSA): In the proposed IDA method, tasks on the smartphones are selected carefully with the linear relaxation and convex optimization methods for energy saving. In RTSA, the mobile devices are selected to access a cloud service as in the proposed IDA method. However,
the tasks on the mobile devices are randomly selected for offloading. As many tasks as possible are selected until the time deadline and transmission error rate constraints (26)–(28) are satisfied. According to the above analysis, the computation complexity is $O(N \cdot M^3)$ for all smartphones in the mobile cloud environment.

Comparing the three proposed cloud offloading methods discussed above, we find that the proposed IDA method makes near-optimal offloading decisions with high efficiency. RTSA and ATSA have lower complexity than IDA to find offloading solutions, with no energy saving performance guarantee. Furthermore, all three cloud offloading methods are compared with the local mobile computing (LMC) method. In this method, all tasks are executed locally on the mobile devices. With no wireless connection, no transmission error will be observed.

**Energy consumption analysis:** As can be seen in Fig. 5, the average energy consumption of smartphones for the four algorithms in 1000 simulation runs is compared. It is shown that the smartphones consume approximately $130.375 \, \text{J}$ of energy with the LMC method. Because of the mobile cloud offloading service, the ATSA, RTSA, and IDA methods exhibit a relatively low energy consumption. At the beginning with five smartphones, the three methods exhibit an energy consumption of 81.46 J, 82.49 J, and 50.35 J, respectively. In the case with few smartphones and low interference, all smartphones can access the cloud service. Particularly, the computation-intensive tasks are offloaded. Thus, a significant amount of energy can be saved via task offloading. Then, with an increased number of smartphones, more mutual interference will be caused. With an inferior wireless communication quality, a greater percentage of tasks will be executed locally. Thus, the average energy consumption of smartphones with a cloud service will gradually approach that of the LMC method. At the number of 50, the ATSA, RTSA, and IDA methods need 127.32 J, 128.51 J, and 98.52 J, respectively.

Then, we compare the energy saving performance of three cloud offloading methods, i.e., RTSA, ATSA, and IDA. Comparing ATSA and IDA, we find that ATSA achieves an energy consumption of 81.46 J on average, while IDA only exhibits that of 50.35 J for five mobile devices. This is attributed to the fact that the proposed IDA algorithm makes near-optimal offloading decisions. While using ATSA, a smartphone will offload all 10 tasks to the cloud only if the time deadline and transmission error rate constraints are satisfied. The harsh terms restrict the number of smartphones for offloading. As shown in Fig. 4(a) and (b), with more smartphones in the scenario causing added interference, the proposed IDA always exhibits more energy saving. Further, although the RTSA and IDA methods select mobile devices similarly for offloading, the tasks are randomly selected in the RTSA method. Therefore, the IDA algorithm always exhibits a lower energy consumption. In conclusion, a mobile cloud offloading method can always save a considerable amount of energy for smartphones. Further, the proposed IDA algorithm can find a better energy saving solution than the other two cloud offloading methods, namely RTSA and ATSA.

**Time delay analysis:** As can be seen in Fig. 6, the average task execution time of smartphones for the four considered algorithms for 1000 simulation runs is compared. A smartphone needs approximately 186.56 s to complete the execution individually with the LMC method. In general, the IDA algorithm always
interference will be caused. Therefore, some more tasks will be assigned to the case of RTSA than in the case of IDA, less communication complexity will be increased from 5 to 50. As fewer smartphones are selected in the average number of tasks for offloading increases from 50.0 to 140.0. Therefore, the average number of tasks for offloading increases from 5.0 to 23.83. Further, by comparing RTSA and IDA, we find that the IDA algorithm can be executed locally in the inferior wireless channel environment.

Thus, the average time delay of smartphones with IDA will gradually approach that of the LMC method. As shown in Fig. 6, the IDA method needs 165.17 s on average at the number of 50. Then, by comparing ATSA and IDA, we find that the IDA algorithm always exhibits a smaller time delay than ATSA. Further, by comparing RTSA and IDA, we find that if the tasks are randomly selected with RTSA, more communication-intensive but not computation-intensive tasks may be selected. Among the three offloading algorithms, namely ATSA, RTSA, and IDA, ATSA exhibits the maximum time delay of 184.26 s with 50 smartphones in the network, which is close to the time delay of the LMC method.

Average characteristics of proposed offloading algorithms: Figs. 7 and 8 illustrate the average number of selected smartphones and tasks for offloading with the proposed task offloading algorithms IDA, RTSA, and ATSA over 1000 simulation runs.

As can be seen in Fig. 7, with the IDA method, as the number of smartphones increases from 5 to 30, the average number of smartphones for offloading increases rapidly from 5.0 to 24.80. Furthermore, as the number of smartphones increases from 30 to 50, with added mutual interference, the average number of smartphones for offloading gradually becomes saturated because of the time and transmission error rate constraints. That is, it increases slowly from 24.80 to 26.07. As seen in Fig. 8, the average number of tasks for offloading also increases from 20.94 to 60.40 as the number of smartphones increases from 5 to 35. However, as the number of smartphones increases from 35 to 50, with more mutual interference, the number of tasks on each smartphone for offloading declines for the time deadline and transmission error rate constraints. The more computation intensive a task is, the more likely is its selection for offloading. The other tasks will be executed locally in the inferior wireless channel environment. Thus, the average number of tasks for offloading rapidly decreases from 60.40 to 47.46.

Similar to IDA, in the RTSA method, the average number of smartphones for offloading increases from 5.0 to 23.83. Further, the average number of tasks for offloading increases from 50.0 to 113.84 and then, decreases to 49.06 as the number of smartphones increases from 5 to 50. As fewer smartphones are selected in the case of RTSA than in the case of IDA, less communication interference will be caused. Therefore, some more tasks will be offloaded in the case of RTSA. However, in the ATSA method, all tasks on the smartphones are offloaded onto the cloud side if the constraints are satisfied. In the case of harsh constraints, fewer smartphones are qualified to access the mobile cloud service. Thus, the average number of smartphones for offloading slowly increases from 5.0 and remains at about 14.00 as the number of smartphones increases from 5 to 50. Therefore, the average number of tasks for offloading increases from 50.0 to 140.0.

With a mobile cloud offloading service, considerable energy consumption for task execution can be saved and the time delay can be reduced simultaneously. Compared with the LMC, RTAS, and ATSA methods, the IDA algorithm always reveals the best energy saving and time delay reduction performance because it always selects appropriate tasks, i.e., computation-intensive tasks, for offloading. Moreover, with the proposed IDA algorithm, more mobile devices can take advantage of the cloud offloading service. In conclusion, the proposed IDA algorithm reveals near-optimal offloading decisions that can always find a better solution to optimize the energy consumption of mobile devices.

7. Conclusion and future work

In this paper, instead of device offloading, we delved deep into the task level to investigate whether every task is suitable for offloading or not in a wireless fading channel over cellular networks. Considering the task execution time and transmission error rate constraints, the energy-saving task offloading decision problem is formulated as an NP-hard 0–1 nonlinear integer programming optimization. An iterative decoupling algorithm that combines variable relation, Lagrange duality, and KKT techniques is proposed for making near-optimal offloading decisions, thereby reducing the computation complexity to the acceptable $O(N \cdot M^3)$. A powerful computation model on the cloud side undertakes the computing process, thus providing guidance to mobile devices for energy saving.

In the future, task offloading in future wireless heterogeneous networks, including cellular and Wi-Fi networks, will be investigated to provide ubiquitous mobile cloud services. Moreover, to ease the mutual interference caused by mobile offloading among mobile devices, a cooperative device-to-device communication mechanism will be introduced for further research so that mobile devices can help each other to offload tasks. This mechanism can improve the channel capacity with a relatively high spectrum efficiency, better exploiting the benefits of task offloading.
Acknowledgments

The authors would like to acknowledge that this work was partially supported by the National Natural Science Foundation of China (Grant Nos. 61379111, 61402538, 61202342, 61403424), and the Fundamental Research Funds for the Central Universities of Central South University under Grant 2015zzts056.

Appendix A

Proof of Lemma 1. According to the previous studies, we derive new sets $K_i$ in which the task may have a chance to save energy through task offloading. Namely, all tasks in set $K_i$ satisfy

$$e_i^m < e_j^m, \quad j \in K_i. \quad (51)$$

It is a useful feature to prove the convexity of the continuous optimization problems. For objective function (26), we have

$$\frac{\partial \rho_l}{\partial \rho_j} = e_i^m - e_j^m > 0, \quad \frac{\partial^2 \rho_l}{\partial^2 \rho_j} = 0. \quad (52)$$

Then for constraints (27)–(29), we have

$$\frac{\partial T_i^m}{\partial \rho_j} = t_i^m > 0, \quad \frac{\partial^2 T_i^m}{\partial^2 \rho_j} = 0. \quad (53)$$

$$\frac{\partial T_i}{\partial \rho_j} = -t_i^m - t_j^m < 0, \quad \frac{\partial^2 T_i}{\partial^2 \rho_j} = 0. \quad (54)$$

$$\frac{\partial \Gamma_i}{\partial \rho_j} = \epsilon_i \prod_j (1 - \rho_j) |\epsilon_i - 1| < 0, \quad \frac{\partial^2 \Gamma_i}{\partial^2 \rho_j} = 0. \quad (55)$$

Thus $T_i^m$ is a convex function, while $T_i$ and $\Gamma_i$ are concave functions.

Appendix B

Proof of Lemma 2. The KKT conditions (34)–(37) above imply that

$$\begin{align*}
\text{if } & T_i^m - T_i^d < 0, \text{ then } u_i = 0 \quad (56) \\
\text{if } & u_i > 0, \text{ then } T_i^m - T_i^d = 0 \\
\text{if } & T_i^m - T_i^d < 0, \text{ then } v_i = 0 \quad (57) \\
\text{if } & v_i > 0, \text{ then } T_i^m - T_i^d = 0 \quad (58) \\
\text{if } & T_i^m - T_i^d < 0 \text{ and } \Gamma_i - \epsilon_i < 0, \text{ then } k_i = 0 \\
\text{if } & k_i > 0, \text{ then } \Gamma_i - \epsilon_i = 0.
\end{align*}$$

If $u_i > 0$, then $T_m^i = T_d^i$ must be satisfied. Besides, it is worth noting that $T_m^i$ are increasing functions of $\rho_j$. It means that we can reduce $\rho_j$ to let $T_m^i < T_d^i$. With $\rho_j$’s reduction, mobile device has smaller energy consumption, which is more suitable for the mobile device. That is to say, $u_i = 0$ and $T_m^i < T_d^i$ are optimal results.

Similarly, if $v_i > 0$ and $T_m^i < T_d^i$ are satisfied, then $T_m^i$ are decreasing functions of $\rho_j$. We can decrease $\rho_j$ so that $T_m^i = T_d^i$ can be satisfied, which will leads to smaller $\rho_j$. In addition, if $k_i = 0$ and $\Gamma_i - \epsilon_i < 0$, then $\rho_j$ are also decreasing functions of $\rho_j$. we can decrease $\rho_j$ so that $\Gamma_i = \epsilon_i$ can be satisfied, which also leads to smaller $\rho_j$. Besides, if $T_m^i = T_d^i$ and $\Gamma_i = \epsilon_i$ are satisfied at the same time, $v_i > 0$, $k_i > 0$. In conclusion, $u_i = 0$, $v_i = 0$, $k_i > 0$ or $u_i > 0$, $v_i > 0$, $k_i = 0$ or $u_i = 0$, $v_i > 0$, $k_i > 0$ are optimal solutions.

Appendix C

Proof of Lemma 3. In the proposed iterative IDA algorithm, we initialize the iteration by considering the worst case when all mobile devices choose to offload tasks, i.e., all $\xi_i = 1$. In the iteration, we may witness two different cases.

The first is that, for certain mobile devices, all tasks may choose to be executed locally on these mobile devices in this iteration. Namely for the certain mobile devices, these $\xi_i$ are modified from 1 to 0. Then we update sets $L_k$ and $K_i$. According to Eq. (2), the SINR $\gamma_i$ keeps increasing in the iteration. Therefore, the channel capacity $R_i$ also increases on the basis of Eq. (4); yet the BER $\epsilon_i$, input data sending time $t_d^i$ and the offloading energy consumption $e_i$ decline on the basis of Eqs. (5), (13), (11). Then with declined $e_i$ and $t_d^i$, $\rho_i$ can be derived from constraints (26)–(28), more tasks can be offloaded on mobile device $i$, i.e., more $\rho_i$ will be set as 0. Subsequently, as can be seen in (14), the energy consumption of mobile device $i$ always declines in each iteration.

The other is that in this iteration all the $\xi_i$ stop changing, which means the mobile device chosen for offloading is determined then. And $c_i$, $T_i$, $e_i$, $e_j$ also remain the same. So the iteration stops. In conclusion, the proposed iterative IDA algorithm reveals lower energy consumption in each iteration, yielding close-to-optimal offloading decisions $\rho_i$ when the iteration terminates.

References


