Wireless Networks manuscript No.
(will be inserted by the editor)

Data Offloading and Task Allocation for Cloudlet-assisted Ad Hoc Mobile Clouds

Xijuan Guo · Liqing Liu · Zheng Chang · Tapani Ristaniemi

Received: date / Accepted: date

Abstract Nowadays, although the data processing capabilities of the modern mobile devices are developed in a fast speed, the resources are still limited in terms of processing capacity and battery lifetime. Some applications, in particular the computationally intensive ones, such as multimedia and gaming, often require more computational resources than a mobile device can afford. One way to address such a problem is that the mobile device can offload those tasks to the centralized cloud with data centers, the nearby cloudlet or ad hoc mobile cloud. In this paper, we propose a data offloading and task allocation scheme for a cloudlet-assisted ad hoc mobile cloud in which the master device (MD) who has computational tasks can access resources from nearby slave devices (SDs) or the cloudlet, instead of the centralized cloud, to share the workload, in order to reduce the energy consumption and computational cost. A two-stage Stackelberg game is then formulated where the SDs determine the amount of data execution units that they are willing to provide, while the MD who has the data and tasks to offload sets the price strategies for different SDs accordingly. By using the backward induction method, the Stackelberg equilibrium is derived. Extensive simulations are conducted to demonstrate the effectiveness of the proposed scheme.

X. Guo · L. Liu
College of Information Science and Engineering,
Yanshan University,
Qinhuangdao 066004, P. R. China.
Z. Chang (✉) · T. Ristaniemi
Department of Mathematical Information Technology,
University of Jyväskyl? P.O.Box 35,
FIN-40014 Jyväskyl? Finland.
E-mail: zheng.chang@jyu.fi.

Keywords cloud computing · mobile cloud computing · cloudlet · ad hoc mobile cloud · offloading · Stackelberg game

1 Introduction

With the rapid development of ICT industry, user equipment (UE) has become an indispensable part of our daily life and advanced mobile technologies have also led to an explosive growth in mobile application markets. However, due to the restrictions on size, weight, battery life, ergonomics, and heat dissipation and so on, the computing capacity of a mobile device remains limited, which, unfortunately, may hinder the users from fully enjoying the high speed wireless networks and may even disturb the daily work of business users [1].

To expand the limited capabilities of the UEs, one effective solution is to integrate cloud computing technology with mobile UEs to produce so called mobile cloud computing (MCC) platform[2–4]. In MCC, the computational-intensive application tasks or storage-intensive jobs are transferred from the UEs to the cloud which has rich computational resources for executing and processing. After processing is done at the cloud side, the final result will be returned back to the mobile UEs. By such, MCC is able to efficiently address the problems of limited processing capabilities and limited battery of the UEs [5]. Meanwhile, the frequently required images, videos or other multimedia files, can also be delivered to the cloud storage and then transmitted to the UEs whenever they are needed. In this way, the problem of storage limitation can be solved as well [6]. Typically there are three types of mobile cloud architectures, namely the traditional centralized cloud

[7], the recently emerged cloudlet [8] and the peer-based ad hoc mobile cloud [9].

The traditional centralized cloud (such as Amazon EC2 cloud, Microsoft Windows Azure or Rackspace) can provide huge storage, high computation power, as well as reliable security. By offloading data and tasks of mobile application to the cloud server for execution, the performance of mobile applications can be greatly improved and the energy consumption of UEs can be significantly reduced. However, it is worth mentioning that the traditional centralized cloud or data center is usually remotely located and far away from their users. Thus, for latency-sensitive mobile applications, such as high quality video streaming, mobile gaming and so on, offloading to the centralized cloud may not satisfy the requirement of the UEs.

To solve the problem, some researchers propose to add an abstraction tier, named as cloudlet which is considered as an emerging paradigm for providing better support both latency-sensitive and resource-intensive mobile applications. A cloudlet is a set of computing units that is well-connected to the Internet and is available for nearby mobile devices [8]. A key advantage of cloudlets over the cloud, is that the close physical proximity between cloudlets and UEs enables shorter communication delays, thereby improving the user experience of interactive applications. Meanwhile, with the increasing processing capacity of modern UEs, peer-assisted computing is also an emerging topic that attracts growing attention in the research community. This paradigm regards users in vicinity as an ad hoc mobile cloud [9], where neighboring UEs are able to utilize and share resources in a cooperative manner. Such a mechanism provides an advantage of having less offload latency and bandwidth consumption as compared to traditional cloud computing, since mobile devices could communicate with each other via device-to-device (D2D) connections. However, the ad hoc mobile cloud computing platform still confront some challenges such as device mobility, workload distribution, connectivity options, cost estimation and energy limitation and so on, which call for a continuous research effort.

Given the rapid growth of mobile devices and widely deployed cloudlets in the wireless access networks, in this paper, we consider a cloudlet-assisted ad hoc mobile cloud model which is sparely studied in the literature. With this tiered networking architecture, a UE can offload its task to nearby mobile devices or to the cloudlet, rather to the centralized cloud, in order to save energy and maximize its revenue. The main contribution of this paper is summarized as follows:

1. In this work, we explicitly consider different costs during data offloading and task allocation process, such as computing cost, communications cost, and energy consumption etc, which can thoroughly complement and analyze the effects of cloudlet-assisted ad hoc mobile cloud.
2. We jointly optimize the utilization of the resources from the ad hoc mobile cloud and the cloudlet under several Quality of service (QoS) constraints. A two-stage Stackelberg game is then formulated with realistic considerations. With the backward principle, we propose an efficient algorithm to derive the Stackelberg equilibrium, which is the optimal strategy profile in the sense that no player can find better strategy if he deviates from the current strategy unilaterally. The solutions, obtained by using convex optimization approach, is feasible with low complexity.
3. Extensive simulations are conducted to evaluate the properties and effectiveness of the presented schemes.

The reminder of this paper is organized as follows. We first briefly overview the recent works in Section 2. In Section 3, we introduce the details of our proposed architecture and formulate the problem as a two-stage Stackelberg game. Section 4 and Section 5 proves and solves the Stackelberg equilibrium with the backward induction method, respectively. In Section 6, the performance evaluations are presented to verify the proposed schemes and finally, we conclude our work in Section 7.

2 Related Work

In the MCC, the researches on data offloading involve making decisions about whether to accept the mobile service request, where to run the mobile application and how to allocate computing resources if the request is accepted. The offloading decisions are usually constructed by analyzing parameters, including computation speeds, bandwidths, power coefficients, communication cost of all participating devices. Most existing studies focused on offloading of mobile users' tasks to remote clouds by exploring different pricing models or different efficient energy-conserve task scheduling approaches, such as [10–16]. For example, the author of [14] formulates the decentralized computation offload decision making problem among mobile device users as a decentralized computation offloading game, analyzed the structural property of the game and showed that the game always admits a Nash equilibrium. The authors of [15] investigates the problem of how to conserve energy for the resource-constrained mobile device, by optimally executing mobile applications in either the local mobile device or the cloud. Jiang et al. of [16] present a Lyapunov optimization-based scheme for cloud offload-

ing scheduling, as well as download scheduling for cloud execution output, for multiple applications running in a mobile device with a multi-core CPU.

However, the average access delay between users and remote clouds can be prohibitively long. Instead, cloudlets deployed in the vicinities of users have been quickly gaining recognition as alternative offloading destinations due to the short response time and capability of reducing the energy consumption of mobile devices [17–19]. Cloudlet based mobile computing now is also referred to as fog computing with the aim to reduce the access latency between mobile users and remote clouds by providing compute, storage, and networking services within the proximity of mobile users [19–23]. In [19], the authors consider the multi-resource allocation problem for the cloudlet environment with resource-intensive and latency-sensitive mobile applications and derive an efficient algorithm that allows the system to adaptively allocate an optimal amount of wireless bandwidth, cloudlet computing resource, and distant cloud computing resources. In [22] the authors propose a double auction mechanism, which coordinates the resource trading between mobile devices (buyer) and cloudlets (seller), according to spatial locations of cloudlets and their distinct capabilities or hosted resources in order to improve resource utilization of cloudlets. In [23] the authors propose a Markov decision process based dynamic offloading algorithm for the mobile users to find the optimal offloading policy while minimizing the offloading and processing cost.

To date, the ad hoc mobile cloud is a newly emerging concept in the recent years, and there are only a few works on the design of ad hoc mobile cloud. In [24], the authors introduce the architecture and services modes of ad hoc mobile cloud. In [25], Chi et al. propose an ad hoc mobile cloud-based gaming architecture, which considers both progressive and collaborative downloading of game resources as well as cooperative task processing. To our best knowledge, there are few papers studying offloading data and tasks in a cloudlet-assisted ad hoc mobile cloud. Therefore, in this paper, we study the data and task allocating problem with game theory under the considered system. However, it can be noticed that the aforementioned papers scarcely utilize the game theoretic approaches and take the utility into consideration when designing the offloading algorithm in the cloudlet-assist platform. Moreover, one can also find that the development of the ad hoc mobile cloud platform has not received equal attention. Therefore, in this paper, we aim to investigate the data offloading and task allocation problems in a cloudlet-assisted ad hoc mobile cloud platform.

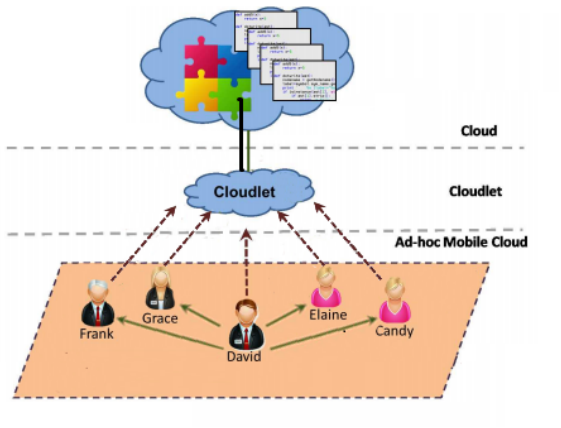

Fig. 1 Cloudlet-assisted ad hoc mobile cloud

3 System Model and Problem Formulation

3.1 System Model

The cloudlet-based ad hoc mobile cloud is presented in Fig. 1. Hereafter we use the term master device (MD) to refer to the originating UE which sends workload to the nearby UEs or to the cloudlet. The UEs which process the workload will be referred to as slave devices (SDs). Without loss of generality, we assume that the MD itself has no processing capacity. However, if it has processing capacity, it can be considered as an extra SD which incurs no communication cost to share the workload.

As shown in Fig. 1, in the considered system, we assume there are one MD U (David) and a set of SDs (Frank, Grace, Elaine, Candy and so on) denoted as $\mathcal{D} = \{d_1, d_2, \cdots d_j, \ \cdots, d_m\}$ which are willing to share their unused resources to the MD for some extra income. For simplicity, we assume that the MD has n homogeneous data units that need to be processed, which are assumed to be independent in terms of execution sequence and can be executed parallel. We also assume that each data unit at the MD requires a data process unit at a SD. The data processing unit can be specified by using the following attributes:

1. Storage consumption, which is the memory space required in order to process the execution unit.
2. CPU unit, which is the CPU resources required in order to compute the task.
3. Size of data sent, which is the size of the offloading task.
4. Size of data received, which is the size of the computation results received by the MD after the task is successfully executed.

In this paper, we assume that the size of each data unit is h, and we also assume the size of output work-

load data is equal to a fraction ρ of input workload data. The MD can use computational resources that are available on nearby SDs or cloudlet to complete the tasks. It is assumed that the MD has to pay a relatively expensive price p_c per data processing unit if it chooses the cloudlet which has more resource than the nearby SDs. If the MD chooses the nearby SDs to execute tasks, it would pay a cheaper price $p_j, \forall j \in \{1, 2, \cdots, m\}$ per data processing unit than the cloudlet price p_c. In the perspective of the MD, it prefers to offload the tasks to the nearby SDs than the cloudlet considering the price and the communication cost. However, the MD has to seek help from the cloudlet when the SDs have not enough resource to process these tasks.

For each SD $d_j \in \mathcal{D}$, we assume that the maximum amount of data processing units it maintains is E_j, which is the total capacity when the SD with full battery and no other tasks processing on it. Additionally, for each SD d_j, we introduce the inconvenience parameter β_j, which denotes the inconvenience incurred by the resource usage of executing tasks for the MD. Here, the resource usage can be the usage of the battery, CPU, memory, or a combination of them. For example, if a SD is with a very limited battery and cannot be timely recharged, its inconvenience parameter would be very high, as executing tasks would accelerate its power outage. The number of data processing units which the SD d_j can provide is e_j, certainly $e_j \leq E_j$. The execution speed is q_j which is related to the inconvenience parameter β_j and intrinsic properties of SD d_j. The MD offers p_j as the price per data processing unit to SD d_j according to the capacity and inconvenience parameter it maintains. In addition, the MD offers different price to different SDs in order to obtain the maximum revenue.Some key notations can also be found in Table 1.

Table 1 Notations

Notations	Meanings
U	the MD in the system
n	the number of data units of the MD
h	the size of data in each data unit
d_j	the j-th SD
p_j	the per data processing unit for SD d_j
E_j	the maximum number of data units of SD d_j
e_j	the number of data units of SD d_j providing
β_j	the inconvenience parameter of SD d_j
η_i	the per unit cost for computation of SD d_j
q_j	the execution speed of SD d_j
r_d	the per unit data transferring cost by D2D
ρ	the data compression ratio for output data
r_w	the per unit data communication cost by WiFi
α_j	the utility level of SD d_j
λ_j^c	the per energy for execution of SD d_j
λ_j^t	the per energy for communication of SD d_j
E_j^{th}	the threshold energy of SD d_j dedicating

3.2 Utility and Cost Function

3.2.1 Utility Function

By offloading e_j data processing units to the SD d_j, the MD obtains an according utility denoted by function U_M^j, here we adopt the logarithmic utility function [27], which is based on the law of diminishing marginal utility in economics and has been widely used in the mobile computing and wireless communications. The utility of the MD obtained from d_j can be denoted as follows:

$$U_M^j = \alpha_j q_j \log_2 (1 + e_j), \tag{1}$$

where α_j denotes the utility level of SD d_j.

In the perspective of SD d_j, given the price p_j, each SD would determine the amount of data processing units that it is willing to provide, by considering both the gained rewards and incurred inconvenience. We first design a utility function for d_j, which should capture the following properties:

1. The larger the value of p_j, the higher the utility of each SD, as higher price often brings higher payoff.
2. The higher the inconvenience parameter, the lower the utility of the SD would obtain.
3. The utility function of each SD should be a concave function of e_j, when e_j is no more than a certain value (e.g., extreme point), the utility function is a non-decreasing function of e_j. Because the SD can gain more payoff from the MD with more e_j, however, when e_j becomes too large, the utility will decrease, as the SD is also constrained with limited resources, and executing too many tasks would sacrifice its own service and also accelerate the power outage.

Based on these properties, we define a utility function [28] for SD d_j as follows:

$$U_j = p_j E_j e_j - \beta_j e_j^2 \ . \tag{2}$$

3.2.2 Computing Cost

Firstly, for executing the assigned data units which the MD offloads to SD d_j, the computing cost of SD d_j is defined as [26]

$$D_j (e_j) = \eta_j \frac{e_j}{q_j}, \tag{3}$$

where η_j is per unit cost for computation on SD d_j and q_j is the execution speed of SD d_j.

3.2.3 Communication Cost

To this end, we are able to formulate the communications cost for the data transfer between the MD and SDs. At first, two assumptions are made to simplify the formulation of communications cost [26]:

1. We assume that the workload must be completely received on the SD before it starts processing and it can only send results back to the MD when it completely finishes the processing. The workload processing can be considered as a three-phase procedure comprising transferring the execution units from the MD to the SD, processing the data processing units on the SDs and returning the results to the MD.
2. We assume that the communications cost is charged by the amount of data transferred but not by the time the mobile devices are connecting. Indeed, as the network bandwidth is a shared resource, many mobile users use this shared resource at the same time. Thus, for the same amount of execution units, the time spent for transferring the data processing units varies depending on the load of the network.

Let r_d denotes the per unit data transmission cost by D2D connectivity, and h denotes the data size in each data processing unit. Thus, the communications cost for the MD to transfer e_j data units to the d_j is denoted as follows:

$$C_M^j\left(e_j\right) = r_d h e_j. \tag{4}$$

The communications cost for the SD d_j to return the result to the MD is denoted as follows:

$$C_j\left(e_j\right) = \rho r_d h e_j. \tag{5}$$

When the MD has to offload some data to the cloudlet as the SDs have no enough data processing units to meet the demand of MD, the communications cost between the MD and the cloudlet is denoted as follows:

$$C_M^c = h r_w \left(n - \sum_{j=1}^{m} e_j\right)^+, \tag{6}$$

where $(.)^+ = \max(., 0)$ and r_w denotes the per unit data communication cost by transferring the data to the cloudlet via WiFi.

3.2.4 Energy Consumption

While accepting to process the workload e_j, SD d_j has to consume energy to accomplish the task. For workload e_j, the energy consumption is denoted as follows:

$$E_j\left(e_j\right) = \lambda_j^c \frac{e_j}{q_j} + \lambda_j^t \frac{h\rho e_j}{b_d}, \tag{7}$$

where λ_j^c and λ_j^t are the per unit energy consumed for processing and communication respectively, and b_d is the bandwidth of the D2D connection. In (7), the first term is the total energy consumed for processing and the second term is the amount of energy consumed for communication. Generally, a SD may not want to drain off its battery for other tasks. It may set a threshold E_j^{th} indicating the maximum amount of energy that it can dedicates. Thus, the energy consumption of the SD d_j must satisfy the condition:

$$E_j\left(e_j\right) \leq E_j^{th}. \tag{8}$$

3.3 Problem Formulation

Obviously, the MD needs to determine the price for the data processing at the SDs. To reduce cost, the MD may offers a lower price. However, lower price would fail to encourage the SDs sharing the resources, and hence, expected performance gain cannot be achieved. In contrast, if the price sets too high, the profit of the MD would decrease. Therefore, it is necessary to set a appropriate price to balance both parties.

Combing the utility function and the defined cost functions which are deduced above, we can obtain the revenue function P_M for the MD denoted as follows:

$$\begin{aligned} \max \quad & P_M = \sum_{j=1}^{m} \alpha_j q_j \log_2\left(1 + e_j\right) - \sum_{j=1}^{m} p_j e_j \\ & - \sum_{j=1}^{m} r_d h e_j - \left(h r_w + p_c\right)\left(n - \sum_{j=1}^{m} e_j\right)^+ \end{aligned} \tag{9}$$

subject to

$$0 < p_j < p_c. \tag{10}$$

Hence, given price p_j, SD d_j determines the amount of data processing units it is willing to provide, the objective function P_S^j of SD d_j is denoted as follows:

$$\max_{e_j} \quad P_S^j = \left(p_j E_j e_j - \beta_j e_j^2\right) - \eta_j \frac{e_j}{q_j} - \rho h r_d e_j. \tag{11}$$

subject to

$$0 \leq e_j \leq E_j, \tag{12}$$

$$\sum_{j=1}^{m} e_j \leq n, \tag{13}$$

$$E_j\left(e_j\right) = \lambda_j^c \frac{e_j}{q_j} + \lambda_j^t \frac{h\rho e_j}{b_d} \leq E_j^{th}. \tag{14}$$

4 Equilibrium Analysis

As discussed above, the SDs determine the amount of data processing units e_j to provide, and the MD sets the prices to different SDs. In order to find the optimal pricing setting and data processing allocation decisions, a two stage Stackelberg game [29] is formulated based on the considered system model, which is denoted as follows:

$$\begin{aligned} \Gamma = \Big\{ & \left(U, \mathcal{D} = \{d_1, d_2, \cdots d_j, \cdots, d_m\}\right), \\ & \left(p_j\right)_{j=1}^m, \left(e_j\right)_{j=1}^m, \left(P_M, P_S^j\right) \Big\}. \end{aligned} \tag{15}$$

In the above formulation, $\{U, \mathcal{D}\}$ is the set of players where MD U is the leader and SD $d_j\,(j = 1, 2, \cdots, m)$ is the follower, $(p_j)_{j=1}^m$ is the strategy vector for the leader, $(e_j)_{j=1}^m$ is the strategy vector for the followers. P_M and P_S^j are the payoff functions for MD and SD respectively.

As the leader, the MD chooses its strategy p_j in the first stage, while each follower independently decides $e_j, \forall j \in \{1, 2, \cdots, m\}$ with p_j in the second stage. Now we specifically describe the two stages in the backward induction method.

Theorem 1 *(Existence of Unique Nash Equilibrium for m-non-cooperative game) For the finite data processing units which the MD need to run, the Stackelberg game admits a unique Nash equilibrium solution* $e^* = \left(e_1^*, \cdots, e_j^*, \cdots, e_m^*\right) = \left(e_j^*, e_{-j}^*\right)$ *and* $p^* = \left(p_1^*, \cdots, p_j^*, \cdots p_m^*\right)$*, among* e_{-j}^* *is the other SDs decision strategy except SD* d_j*, similar to* p_{-j}^**, at the point* (e^*, p^*) *satisfies the following properties:*

$$P_S^j\left(e^*, p^*\right) \geq P_S^j\left(e_j, e_{-j}^*, p^*\right), j = 1, 2, \cdots, m, \tag{16}$$

$$P_M\left(e^*, p^*\right) \geq P_M\left(e^*, p_{-j}^*, p_j\right). \tag{17}$$

Proof Firstly we take the first derivatives of P_S^j respect to $e_j \;\; (j = 1, 2, \cdots, m)$, we obtain that:

$$\frac{\partial P_S^j}{\partial e_j} = p_j E_j - 2\beta_j e_j - \eta_j/q_j - \rho h r_d, \tag{18}$$

when $\frac{\partial P_S^j}{\partial e_j} = 0$, and we have

$$e_j = \frac{p_j E_j - \eta_j/q_j - \rho h r_d}{2\beta_j}. \tag{19}$$

Next, we add to P_S^j and obtain the function P_S for all SDs:

$$\begin{aligned} & P_S\left(e_1, e_2, \cdots, e_m; p_1, p_2, \cdots, p_m\right) \\ & = \sum_{j=1}^{m} \left(p_j E_j e_j - \beta_j e_j^2\right) - \sum_{j=1}^{m} \left(\eta_j/q_j + \rho h r_d\right) e_j. \end{aligned} \tag{20}$$

Thus, we have $\frac{\partial^2 P_S}{\partial e_j^2} = -2\beta_j < 0$ and $\frac{\partial^2 P_S}{\partial e_j e'_j} = 0$. So Hessian matrix of P_S can be derived as follows:

$$H_P = \begin{pmatrix} -2\beta_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & -2\beta_m \end{pmatrix}. \tag{21}$$

It is easy to find that the Hessian matrix of H_P is negative definite, so H_P is strictly concave, which implies that the m-player game has a unique Nash equilibrium.

Similarly, we prove that there existing Nash Equilibrium in the Stackelberg game. From (19), we can obtain

$$p_j = \frac{2\beta_j e_j + \eta_j/q_j + \rho h r_d}{E_j}. \tag{22}$$

Submitting (22) into (9), we have $\frac{\partial^2 P_M}{\partial e_j^2} < 0$ and $\frac{\partial^2 P_M}{\partial^2 e_j} = 0$.

Obviously, the Hessian of P_M is negative. Therefore, as P_M is strictly concave and the constraints are linear, which implies that the game has an unique Stackelberg Equilibrium solution.

5 Proposed Scheme for Achieving Stackelberg Equilibrium

In this section we will give the Nash equilibrium solution of the pricing scheme to different SDs. Given the price strategy $p_j, \forall j \in \{1, 2, \cdots, m\}$, each SD independently decides e_j to maximize his/her revenue.

From above, we can obtain

$$e_j^* = \left(\frac{p_j E_j - \eta_j/q_j - \rho h r_d}{2\beta_j}, 0\right)^+. \tag{23}$$

To tackle this problem, we assume that $e_j^* \leq e_j^{th} = \frac{E_j^{th}}{\lambda_j^c/q_j + \lambda_j^t h\rho/b_d}$, where e_j^{th} is the maximum number execution units that SD d_j can provide under the premise of the consuming energy does not exceed the threshold. It is observed that when $p_j < \frac{n_j/q_j + \rho h r_d}{E_j}$, the SDs will not provide execution units, the occurring delay of the MD would be relatively high, in that situation, the MD has to improve the price in order to offload tasks to other devices.

As mentioned above, the best choice for the MD is to offload its data processing units to the nearby SDs to reduce the cost, and it can be realized under certain situation. So we just make $n - \sum_{j=1}^{m} e_j = 0$. Now we substitute (19) into (22), then the optimization of the MD can be translated as follows:

$$\begin{aligned} \max \quad & P_M = \sum_{j=1}^{m} \alpha_j q_j \log_2 \left(1 + \frac{p_j E_j - \eta_j/q_j - \rho h r_d}{2\beta_j}\right) \\ & - \sum_{j=1}^{m} p_j \frac{p_j E_j - \eta_j/q_j - \rho h r_d}{2\beta_j} \\ & - \sum_{j=1}^{m} r_d h \frac{p_j E_j - \eta_j/q_j - \rho h r_d}{2\beta_j}, \\ s.t. \quad & 0 < p_j < p_c \quad (j = 1, 2, \cdots, m). \end{aligned} \tag{24}$$

Let $\Pi = \frac{-\eta_j/q_j - \rho h r_d}{2\beta_j}$, which is regarded as a constant for it has no relationship with the variable p_j. And we can simplify (24) as follows:

$$\begin{aligned} \min \quad & P_M = \sum_{j=1}^{m} p_j \left(\frac{E_j}{2\beta_j} p_j + \Pi\right) \\ & + \sum_{j=1}^{m} r_d h \left(\frac{E_j}{2\beta_j} p_j + \Pi\right) \\ & - \sum_{j=1}^{m} \alpha_j q_j \log_2 \left(1 + \frac{E_j}{2\beta_j} p_j + \Pi\right), \\ s.t. \quad & 0 < p_j < p_c. \end{aligned} \tag{25}$$

The first derivative of P_M is denoted as follows:

$$\frac{dP_M}{dp_j} = \frac{E_j}{\beta_j} p_j + \Pi + \frac{r_d h E_j}{2\beta_j} - \frac{\alpha_j q_j E_j}{\ln 2 \left(2\beta_j + E_j p_j + 2\Pi\beta_j\right)}, \tag{26}$$

$$\frac{d^2 P_M}{dp^2{}_j} = \frac{E_j}{\beta_j} + \frac{\alpha_j q_j E_j^2}{\ln 2 (2\beta_j + E_j p_j + 2\Pi\beta_j)^2} > 0. \tag{27}$$

Therefore, we can see that P_M is a convex function. Let $\frac{dP_M}{dp_j} = 0$, we can obtain the optimal p_j^* numerically.

Now, the Stackelberg game for the resources offloading with differentiated prices is addressed. The SE for the Stackelberg game is given as $(\boldsymbol{p}^*, \boldsymbol{e}^*)$.

6 Simulation Results

In this section, we present the simulation performance of the proposed method. In the simulation, we assume that the MD has 20 homogeneous data units that need be to processed and 4 SDs exist in the model who are willing to provide their resource for extra income. We set that the MD pays $p_c = 20$ to the cloudlet for per data processing unit, which is also the maximum price it can offer to the SDs. For each execution unit, we assume that it contains $h = 5$ MB size of data, after execution, the output size ratio is set to $\rho = 0.8$, which denotes that the original data is compressed. For the D2D network connection, the cost for transferring unit data is $r_d = 20$. Different SDs have different capacities, execution speed, inconvenience parameter, cost of executing data processing unit which are depicted in Table 1.

Table 2 the SDs Parameter

Parameter	SD_1	SD_2	SD_3	SD_4
E_j	12	13	14	15
q_j	182	184	183	186
n_j	85	100	90	95

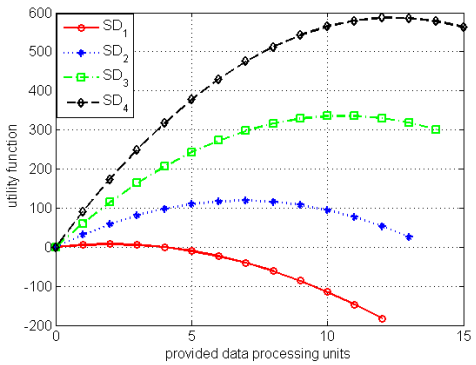

Fig. 2 The utility function of the SD

In Fig. 2, we present the SD utility with respect to e_j given price and inconvenience parameters. From Fig. 2, we can see that the utility of each SD is non-decreasing until a certain value has been reached. However, when e_j becomes too large, the utility will decrease, as the SD is constrained with limited resources, and executing too many tasks would sacrifice its own service. We can derive that SDs would not get the optimal benefits through providing too few or too many data processing units. So it is crucial to for each SD to select a proper e_j.

We also investigate the relationship between the inconvenience parameter β_j and the data processing units e_j which is denoted in Fig. 3. For each SD, under the given price, with inconvenience parameter increasing, provided data processing units are decreasing, which meets with the reality.

As discussed above, the SDs determine the amount of data processing units e_j to provide, and the MD sets the prices p_j $(j = 1, 2, \cdots, 4)$ to different SDs, the link betweene$_j$ and p_j is denoted in (eq-18), we can also display it in Fig. 4. Because of the limit capacity of SD, the number of e_j will not increase unlimitedly. In this situation, we assume the inconvenience parameter of each SD is $\beta = (0.2, 0.3, 0.3, 0.2)$ respectively.

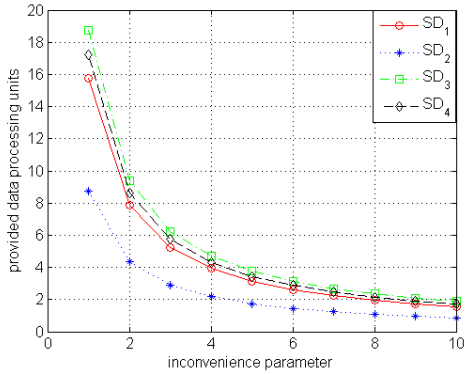

Fig. 3 The effect of inconvenience parameter

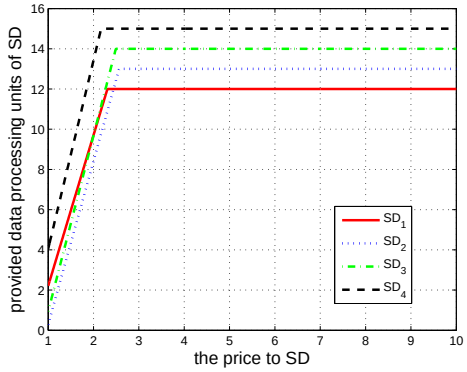

Fig. 4 The impact of the price and provided data processing units

With the concept of Stackelberg equilibrium, the adjusting of the amount of data processing units e_j that each SD willing to provide and the price p_j the MD offering is coupled. We assume that both MD and SDs are rational. On one hand, in order to obtain a larger revenue, the MD would not offer too high price to the SDs; on the other hand, the SDs would not provide too many data processing units as executing too many tasks would accelerate their power outage. Fig. 5 presents the relations among price, provided data processing units of a SD and the revenue of the MD. From Fig. 5, we can see that the proper points (p_j^*, e_j^*) can be achieved at the highest value of the total revenue function. For example, when $p = 10$, with the provided data processing units of SD increasing, we can observe that the revenue of the MD is increased first, and reach its maximum when the number of data processing units is 5 and then decreased. The observations in Fig. 5 demonstrate the necessity of the optimization for the considered problem.

Moreover, we compare our proposed scheme with a modified stochastic workload distribution approach presented in [26]. As we can observe from this figure, our proposed scheme is able to obtain a better revenue for the SDs. The performance gap between the proposed scheme and the stochastic workload distribution approach increases as the number of data units of the MD becomes larger. For example, when the number of data units need to be offloaded is 8, there are merely no difference, while the gap becomes 50 when the number of data units need to be offloaded is 40. This mainly due to the fact that when the MD has a few data to be offloaded, two schemes cannot make too much difference on the revenue performance.

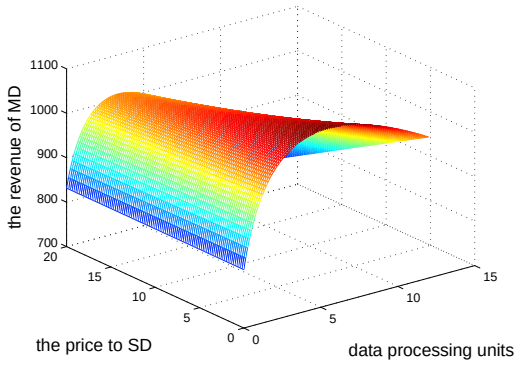

Fig. 5 The total revenue of MD vs. data processing units vs. price

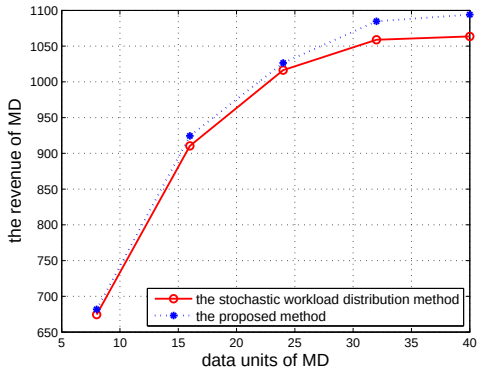

Fig. 6 The comparison between the proposed scheme and the stochastic workload distribution scheme

7 Conclusion

In this paper, we focus on investigating the data offloading and task allocation problem in cloudlet assisted ad hoc mobile cloud. We formulate a two-stage Stackelberg game to benefit both the buyer (MD) and the sellers (SDs) for the data processing units, and show that there exists one unique Stackelberg equilibrium. With the property of objective function, we derive the Stackelberg equilibrium where any player cannot increase the expected profit by changing its strategy unilaterally. That is equivalent to say, the Stackelberg equilibrium

points are the optimal strategies for either the MD or SDs in the system when players do not cooperate with each other. Finally, we demonstrate the efficiency of the proposed scheme through extensive simulations.

References

1. M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Computing*, Vol. 8, no. 4, pp. 14-23, Oct.-Nov.2009.
2. M. R. Rahimi, J. Ren, C. Liu, A. V. Vasilakos, and N. Venkatasubramanian, "Mobile cloud computing: a survey, state of art and future directions," *Mobile Networks and Applications*, Vol. 19, no. 2, pp. 133-143, April 2014.
3. L. Wu, S. K. Garg, and R. Buyya, "SlA-based admission control for a software-as-a-service provider in cloud computing environments," *Journal of Computer and System Science*, Vol. 78, no. 5, pp. 1280-1299, September 2012.
4. F. Liu, P. Shu, H. Jin, L. Ding, J. Yu, D. Niu, and B. Li, "Gearing resource poor mobile devices with powerful clouds: architectures, challenges, and applications," *IEEE Wireless Communications*, Vol. 20, no. 3, pp. 14-22, June 2013.
5. W. Fang, Y. Li, H. Zhang, N. Xiong, J. Lai, and A. V. Vasilakos, "On the throughput-energy tradeoff for data transmission between cloud and mobile devices," *Information Sciences*, Vol. 283, no. 1, pp. 79-93, November 2014.
6. J. Vazifehdan, R. V. Prasad, M. Jacobsson, and I. Niemegeers, "An analytical energy consumption model for packet transfer over wireless links," *IEEE Communications Letters*, vol. 16, no. 1, pp. 30-33, January 2012.
7. K. Kumar and Y. H. Lu, "Cloud computing for mobile users: Can offloading computation save energy," *IEEE Computer*, Vol. 43, no. 4, pp. 51-56, April 2010.
8. Y. Jararweh, L. Tawalbeh, F. Ababneh, and A. Khreishah. "Scalable cloudlet-based mobile computing model," *Procedia Computer Science*, Vol. 34, pp. 434-441, 2014.
9. A. Hasan and J. G. Andrews, "The guard zone in wireless ad hoc networks," *IEEE Transactions on Wireless Communications*, Vol. 6, no. 3, pp. 897-906, March 2007.
10. L. Gu, D. Zeng, A. Barnawi, S. Guo, and I. Stojmenovic. "Optimal task placement with QoS constraints in geo-distributed data centers using DVFS," *IEEE Transactions on Computers*, Vol. 64, no. 7, pp. 2049-2059, July 2015.
11. H. Jin, X. Wang, S. Wu, S. Di, and X. Shi, "Towards optimized fine-grained pricing of IaaS cloud platform," *IEEE Transcation on Cloud Computing*, Vol. 3, no. 4, pp. 436-448, Oct.-Dec. 2014.
12. R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "Cuckoo: a computation offloading framework for smartphones," *Springer Berlin Heidelberg*, Vol. 76, pp. 59-79, 2012.
13. J. Niu, W. Song, and M. Atiquzzaman, "Bandwidth-adaptive partitioning for distributed execution optimization of mobile applications," *Journal of Network and Computer Applications*, Vol. 37, no. 1, pp. 334-347, January 2014.
14. X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transaction on Parallel and Distributed Systems*, Vol. 26, no. 4, pp. 974-984, April 2015.
15. W. W. Zhang, Y. G. Wen, K. Guan, D. Kilper, H. Y. Luo, and D. P. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Transactions on Wireless Communications*, Vol. 12, no. 9, pp. 4569-4581, September 2013.
16. Z. F. Jiang and S. W. Mao, "Energy delay tradeoff in cloud offloading for multi-core mobile devices," *IEEE Access*, Vol. 3, pp. 2306-2316, 2015.
17. W. Cai, V. C. M. Leung, and L. Hu, "A cloudlet-assisted multiplayer cloud gaming system," *Journal of Mobile Networks and Applications*, Vol. 19, no. 2, pp. 144-152, 2014.
18. Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, "Heterogeneity in mobile cloud computing: taxonomy and open challenges," *IEEE Communications Surveys Tutorials*, Vol. 16, no. 1, pp. 369-392, First Quarter 2014.
19. Y. C. Liu and M. J. Lee, "Adaptive multi-resource allocation for cloudlet-based mobile cloud computing system," *IEEE Transactions on Mobile Computing*, DOI 10.1109/TMC. 2015. 2504091.
20. J. T. Su, F. H. Lin, X. W. Zhou, and X. Lu, "Steiner tree based optimal resource caching scheme in fog computing," *China Communications*, Vol. 12, no. 8, pp. 161-168, 2015.
21. W. Cai, Z. Hong, X. F. Wang, H. C. B. Chan, and V. C. M. Leung, "Quality-of-experience optimization for a cloud gaming system with ad hoc cloudlet assistance," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 25, no. 12, pp. 2092-2104, December 2015.
22. K. Kumar, J. Liu, Y. H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Networks and Applications*, Vol. 18, no. 1, pp. 129-140, 2013.
23. Y. Zhang, D. Niyato, and P. Wang, "Offloading in mobile cloudlet systems with intermittent connectivity," *IEEE Transactions on Mobile Computing*, Vol. 14, no. 12, pp. 2516-2530, December 2015.
24. M. Chen, Y. X. Hao, Y. Li, C. F. Lai, and D. Wu, "On the computation offloading at ad hoc cloudlet: architecture and service modes," *IEEE Communications Magazine*, Vol. 53, no. 6, pp. 18-25, June 2015.
25. F. Y. Chi, X. F. Wang, W. Cai, and V. C. M. Leung, "Ad-hoc cloudlet based cooperative cloud gaming," *IEEE Transactions on Cloud Computing*, 2015, DOI 10.1109/TCC.2015.2498936.
26. T. H. Tram, C. K. Tham, and D. Niyato, "A stochastic workload distribution approach for an ad-hoc mobile cloud," *2014 IEEE 6th International Conference on Cloud Computing Technology and Science (CloudCom)*, Singapore, Dec. 2014.
27. L. Tang, and H. Chen, "Joint pricing and capacity planning in the IaaS cloud market," *IEEE Transactions on Cloud Computing*, in press, 2014, DOI: 10.1109/TCC.2014.2372811.
28. Z. Zhou, F. Liu, H. Jin, B. Li, and H. Jiang, "On arbitrating the power-performance tradeoff in SaaS clouds," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 25, no. 10, pp. 2648-2658, Oct. 2014.
29. S. Moya and A. S. Poznyak, "Extraproximal method application for a Stackelberg nash equilibrium calculation in static hierarchical games," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol. 39, no. 6, pp. 1493-1504, December 2009.