Strategy Configurations of Multiple Users Competition for Cloud Service Reservation

Chubo Liu, Kenli Li, *Member, IEEE*, Chengzhong Xu, *Senior Member, IEEE*, and Keqin Li, *Fellow, IEEE*

Abstract—In this paper, we focus on strategy configurations of multiple users to make cloud service reservation. We consider the problem from a game theoretic perspective and formulate it into a non-cooperative game among the multiple cloud users, in which each user is informed with incomplete information of other users. For each user, we design a utility function which combines the net profit with time efficiency and try to maximize its value. We solve the problem by employing variational inequality (VI) theory and prove that there exists a Nash equilibrium solution set for the formulated game. Then, we propose an iterative proximal algorithm (IPA), which is designed to compute a Nash equilibrium solution. The convergence of the IPA algorithm is also analyzed and we find that it converges to a Nash equilibrium if several conditions are satisfied. Finally, we conduct some numerical calculations to verify our theoretical analysis. The experimental results show that our proposed IPA algorithm converges to a stable state very quickly and improves the utilities of all users to certain extent by configuring a proper request strategy.

Index Terms—Cloud service reservation, nash equilibrium, non-cooperative game theory, variational inequality theory

1 Introduction

1.1 Motivation

Cloud computing has recently emerged as a new paradigm for a cloud provider to host and deliver computing services to enterprises and consumers [1]. Usually, the provided services mainly refer to Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS), which are all made available to the general public in a *pay-as-you-go* manner [2], [3]. In most systems, the service provider provides the architecture for users to make reservations in advance [4], [5]. When making reservations for a cloud service, multiple users and the cloud provider need to reach an agreement on the costs of the provided service and make planning to use the service in the reserved time slots, which could lead to a competition for the usage of limited resources [6]. Therefore, it is important for a user to configure his/her task requests in different time slots without complete information of other users, such that his/her utility is maximized.

For a cloud provider, the income (i.e., the revenue) is the service charge to users [7]. When providing services to multiple cloud users, a suitable pricing model is a significant factor that should be taken into account. The reason lies in that a proper pricing model is not just for the profit of a cloud provider, but for the appeals to more cloud users in the market to use cloud service. Specifically, if the per request charge is too high, a user may refuse to use the cloud service, and choose another cloud provider or just finish his/her tasks locally. On the contrary, if the charge is too low, the aggregated requests may be more than enough, which could lead to low service quality (long task response time) and thus dissatisfies its cloud users.

A rational user will choose a strategy to use the service that maximizes his/her own net reward, i.e., the utility obtained by choosing the cloud service minus the payment [1]. On the other hand, the utility of a user is not only determined by the importance of his/her tasks (i.e., how much benefit the user can receive by finishing the tasks), but also closely related to the urgency of the task (i.e., how quickly it can be finished). The same task, such as running an online voice recognition algorithm, is able to generate more utility for a cloud user if it can be completed within a shorter period of time in the cloud [1]. However, considering the energy saving and economic reasons, it is irrational for a cloud provider to provide enough computing resources to satisfy all requests in a time slot. Therefore, multiple cloud users have to compete for the cloud service reservation. Since the payment and time efficiency of each user are affected by decisions of other users, it is natural to analyze the behavior of such systems as strategic games [4].

1.2 Our Contributions

In this paper, we focus on the strategy choices of multiple users to make cloud service reservation. We consider the problem from a game theoretic perspective and formulate it into a non-cooperative game among the multiple cloud users, in which each cloud user is informed with incomplete information of other users. At the beginning, we design a mechanism for the cloud provider and its multiple users to establish negotiations, which is beneficial to both the cloud provider and its users. Specifically, the cloud provider can evaluate proper dynamic pricing parameters to maximize

- *C. Liu and K. Li are with the College of Information Science and Engineering, Hunan University, and National Supercomputing Center in Changsha, Hunan, China 410082. E-mail: {liuchubo, lkl}@hnu.edu.cn.*
- *C. Xu is with the Department of Electrical and Computer Engineering, Wayne State University, Detroit, MI 48202. E-mail: czxu@wayne.edu.*
- *K. Li is with the College of Information Science and Engineering, Hunan University, and National Supercomputing Center in Changsha, Hunan, China 410082 and the Department of Computer Science, State University of New York, New Paltz, NY 12561. E-mail: lik@newpaltz.edu.*

Manuscript received 23 Sept. 2014; revised 7 Jan. 2015; accepted 27 Jan. 2015. Date of publication 29 Jan. 2015; date of current version 20 Jan. 2016. Recommended for acceptance by S. Yu.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2015.2398435

its profit according to the aggregated requests and the users can decide whether to use the cloud service and configure proper strategies which are suitable for their own. For each user, we design a utility function which combines the net profit with time efficiency and try to maximize its value. We solve the problem by employing variation inequality (VI) theory and prove that there exists a Nash equilibrium solution set for the formulated game. Then, we propose an iterative proximal algorithm (IPA), which is designed to compute a Nash equilibrium solution. The convergence of the IPA algorithm is also analyzed and we find that the proposed algorithm converges to a Nash equilibrium if several conditions are satisfied. Finally, we conduct some numerical calculations to verify our theoretical analysis. The experimental results show that our proposed IPA algorithm converges to a stable state very quickly and improves the users' utilities to certain extent by configuring proper request strategies.

1.3 Related Works

In many scenarios, a service provider provides the architecture for users to make reservations in advance [4], [5], [6]. One of the most important aspects that should be taken into account by the provider is its pricing scheme, which is closely related to its profit and the appeals to market users.

Many works have been done on the pricing scheme in the literature [7], [8], [9], [10], [11], [12]. In [7], Cao et al. proposed a time dependent pricing scheme, i.e., the charge of a user is dependent on the service time of his/her requirement. However, we may note that the service time is not only affected by the amount of his/her own requirement, but also influenced by other factors such as the processing capacity of servers and the requirements of others. Mohsenian-Rad et al. [8] proposed a dynamic pricing scheme, in which the per price (the cost of one request or one unit of load) of a certain time slot is set as an increasing and smooth function of the aggregated requests in that time slot. That is to say, when the aggregated requests are quite much in a time slot, the users have to pay relatively high costs to complete the same amount of requests, which is an effective way to convince the users to shift their peak-time task requests. Similar studies and models can be found in [9], [10], [11], [12]. However, these models are only applied to control energy consumption and different from applications in cloud services, since there is no need to consider time efficiency in them.

Game theory is a field of applied mathematics that describes and analyzes scenarios with interactive decisions [13], [14], [15]. It is a formal study of conflicts and cooperation among multiple competitive users [16] and a powerful tool for the design and control of multiagent systems [17]. There has been a growing interest in adopting cooperative and noncooperative game theoretic approaches to modeling many communications problems [18], [19], [20], [21]. A more general framework suitable for investigating and solving various optimization problems and equilibrium models, even when classical game theory may fail, is the variation inequality method that is applicable to a very general class of problems in nonlinear analysis [13], [22]. For more works on game theory, the reader is referred to [8], [12], [21], [23], [24], [25].

1.4 Organization

The rest of the paper is organized as follows. Section 2 describes the models of the system and presents the problem to be solved. Section 3 formulates the problem into a non-cooperative game and solves the problem by employing variational inequality theory. Many analyses are also presented in this section. Section 4 is developed to verify our theoretical analysis and show the effectiveness of our proposed algorithm. We conclude the paper with future work in Section 5.

2 Model Formulation and Analyses

To begin with, we present our system model in the context of a service cloud provider, and establish some important results. In this paper, we are concerned with a market with a service cloud provider and n cloud users, who are competing for the cloud service reservation. We denote the set of users as $\mathcal{N} = \{1, \ldots, n\}$. The arrival of requests from cloud user i ($i \in \mathcal{N}$) is assumed to follow a Poisson process. The cloud provider is modeled by an M/M/m queue, serving a common pool of cloud users with m homogeneous servers. Similar to [26], [27], we assume that the request profile of each user is determined in advance for H future time slots. Each time slot can represent different timing horizons, e.g., one hour of a day.

2.1 Request Profile Model

We consider a user request model motivated by [9], [12], where the user i's ($i \in \mathcal{N}$) request profile over the H future time slots is formulated as

$$\boldsymbol{\lambda}_i = \left(\lambda_i^1, \ldots, \lambda_i^H\right)^T, \tag{1}$$

where λ_i^h ($i \in \mathcal{N}$) is the arrival rate of requests from user i in the hth time slot and it is subject to the constraint $\sum_{h=1}^{H} \lambda_i^h = \Lambda_i$, where Λ_i denotes user i's total requests. The arrivals in different time slots of the requests are assumed to follow a Poisson process. The individual strategy set of user i can be expressed as

$$Q_i = \left\{ \boldsymbol{\lambda}_i \,\middle|\, \sum_{h=1}^{H} \lambda_i^h = \Lambda_i \text{ and } \lambda_i^h \geq 0, \forall h \in \mathcal{H} \right\}, \tag{2}$$

where $\mathcal{H} = \{1, \ldots, H\}$ is the set of all H future time slots.

2.2 Load Billing Model

To efficiently convince the users to shift their peak-time requests and fairly charge the users for their cloud services, we adopt the instantaneous load billing scheme, which is motivated by [9], [12], where the request price (the cost of one request) of a certain time slot is set as an increasing and smooth function of the total requests in that time slot, and the users are charged based on the instantaneous request price. In this paper, we focus on a practical and specific polynomial request price model. Specifically, the service price for one unit of workload of the hth time slot is given by

$$C\left(\lambda_{\Sigma}^h\right) = a\left(\lambda_{\Sigma}^h\right)^2 + b, \tag{3}$$

where a and b are constants with $a, b > 0$, and λ_Σ^h is the aggregated requests from all users in time slot h, i.e., $\lambda_\Sigma^h = \sum_{i=1}^n \lambda_i^h$.

2.3 Cloud Service Model

The cloud provider is modeled by an M/M/m queue, serving a common pool of multiple cloud users with m homogeneous servers. The processing capacity of each server is presented by its service rate μ_0. We denote μ as the total processing capacity of all m servers and Λ as the aggregated requests from all cloud users, respectively. Then we have $\mu = m\mu_0$, and $\Lambda = \sum_{i=1}^n \Lambda_i$.

Let p_i be the probability that there are i service requests (waiting or being processed) and $\rho = \Lambda/\mu$ be the service utilization in the M/M/m queuing system. With reference to [7], [28], we obtain

$$p_i = \begin{cases} \frac{1}{i!}(m\rho)^i p_0, & i < m; \\ \frac{m^m \rho^i}{m!} p_0, & i \geq m; \end{cases} \tag{4}$$

where

$$p_0 = \left\{ \sum_{k=0}^{m-1} \frac{1}{k!}(m\rho)^k + \frac{1}{m!}\frac{(m\rho)^m}{1-\rho} \right\}^{-1}. \tag{5}$$

The average number of service requests (in waiting or in execution) is

$$\bar{N} = \sum_{i=0}^{\infty} k p_i = \frac{p_m}{1-\rho} = m\rho + \frac{\rho}{1-\rho} P_q, \tag{6}$$

where P_q represents the probability that the incoming requests need to wait in queue.

Applying Little's result, we get the average response time as

$$\bar{T} = \frac{\bar{N}}{\Lambda} = \frac{1}{\Lambda}\left(m\rho + \frac{\rho}{1-\rho} P_q \right). \tag{7}$$

In this paper, we assume that all the servers will likely keep busy, because if not so, some servers could be shutdown to reduce mechanical wear and energy cost. For analytical tractability, P_q is assumed to be 1. Therefore, we have

$$\begin{aligned} \bar{T} = \frac{\bar{N}}{\Lambda} &= \frac{1}{\Lambda}\left(m\rho + \frac{\rho}{1-\rho} \right) \\ &= \frac{m}{\mu} + \frac{1}{\mu - \Lambda}. \end{aligned} \tag{8}$$

Now, we focus on time slot h ($h \in \mathcal{H}$). We get that the average response time in that time slot as

$$\bar{T}^h = \frac{m}{\mu} + \frac{1}{\mu - \lambda_\Sigma^h}, \tag{9}$$

where $\lambda_\Sigma^h = \sum_{i=1}^n \lambda_i^h$. In this paper, we assume that $\lambda_i^h < \mu$ ($\forall h \in \mathcal{H}$), i.e., the aggregated requests in time slot h never exceeds the total capacity of all servers.

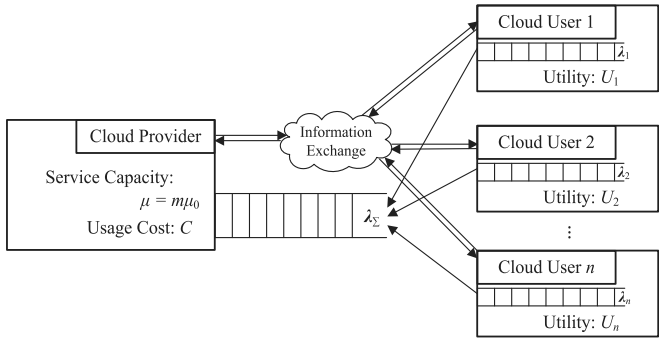


Fig. 1. Architecture model.

2.4 Architecture Model

In this section, we model the architecture of our proposed service mechanism, in which the cloud provider can evaluate proper charge parameters according to the aggregated requests and the cloud users can make proper decisions through the information exchange module. As shown in Fig. 1, each user i ($i \in \mathcal{N}$) is equipped with a utility function (U_i) and the request configuration ($\boldsymbol{\lambda}_i$), i.e., the service reservation strategy over H future time slots. All requests enter a queue to be processed by the cloud computing. Let $\boldsymbol{\lambda}_\Sigma$ be aggregated request vector, then we have $\boldsymbol{\lambda}_\Sigma = \sum_{i=1}^n \boldsymbol{\lambda}_i$. The cloud provider consists of m homogeneous servers with total processing rate μ, i.e., $\mu = m\mu_0$, where μ_0 is the service rate of each server, and puts some information (e.g., price parameters a and b, current aggregated request vector $\boldsymbol{\lambda}_\Sigma$) into the information exchange module. When multiple users try to make a cloud service reservation, they first get information from the exchange module, then compute proper strategies such that their own utilities are maximized and send the newly strategies to the cloud provider. The procedure is terminated until the set of remaining users, who prefer to make cloud service reservation, and their corresponding strategies are kept fixed.

2.5 Problem Formulation

Now, let us consider user i's ($i \in \mathcal{N}$) utility in time slot h. A rational cloud user will seek a strategy to maximize its expected net reward by finishing the tasks, i.e., the benefit obtained by choosing the cloud service minus its total payment. Since all cloud users are charged based on the instantaneous load billing and how much tasks they submit, we denote the cloud user i's payment in time slot h by P_i^h, where $P_i^h = C(\lambda_\Sigma^h)\lambda_i^h$ with $C(\lambda_\Sigma^h)$ denoting the service price for one unit of workload in time slot h. On the other hand, since a user will be more satisfied with much faster service, we also take the average response time into account. Note that time utility will be deteriorated with the delay of time slots. Hence, in this paper, we assume that the deteriorating rate of time utility is δ ($\delta > 1$). Denote $\bar{T}^h$ the average response time and T^h the time utility of user i in time slot h, respectively. Then we have $T^h = \delta^h \bar{T}^h$. More formally, the utility of user i ($i \in \mathcal{N}$) in time slot h is defined as

$$\begin{aligned} U_i^h(\lambda_i^h, \boldsymbol{\lambda}_{-i}^h) &= r\lambda_i^h - P_i^h(\lambda_i^h, \boldsymbol{\lambda}_{-i}^h) - w_i T^h(\lambda_i^h, \boldsymbol{\lambda}_{-i}^h) \\ &= r\lambda_i^h - P_i^h(\lambda_i^h, \boldsymbol{\lambda}_{-i}^h) - w_i \delta^h \bar{T}^h(\lambda_i^h, \boldsymbol{\lambda}_{-i}^h), \end{aligned} \tag{10}$$

where $\boldsymbol{\lambda}^h_{-i} = (\lambda^h_1, \ldots, \lambda^h_{i-1}, \lambda^h_{i+1}, \ldots, \lambda^h_n)$ denotes the vector of all users' request profile in time slot h except that of user i, r $(r > 0)$ is the benefit factor (the reward obtained by one task request), and w_i $(w_i > 0)$ is the waiting cost factor, which reflects its urgency. If a user is more concerned with task completion, then the associated waiting factor w_i might be larger.

For simplicity, we use P^h_i and $\bar{T}^h$ to denote $P^h_i(\lambda^h_i, \boldsymbol{\lambda}^h_{-i})$ and $\bar{T}^h(\lambda^h_i, \boldsymbol{\lambda}^h_{-i})$, respectively. Following the adopted request price model, the total utility obtained by user i $(i \in \mathcal{N})$ over all H future time slots can thus be given by

$$\begin{aligned} U_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}) &= \sum_{h=1}^{H} U^h_i\big(\lambda^h_i, \boldsymbol{\lambda}^h_{-i}\big) \\ &= \sum_{h=1}^{H} \big(r\lambda^h_i - P^h_i - w_i \delta^h \bar{T}^h\big), \end{aligned} \tag{11}$$

where $\boldsymbol{\lambda}_{-i} = (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{i-1}, \boldsymbol{\lambda}_{i+1}, \ldots, \boldsymbol{\lambda}_n)$ denotes the $(n-1)$ $H \times 1$ vector of all users' request profile except that of user i.

We consider the scenario where all users are selfish. Specifically, each user tries to maximize his/her total utility over the H future time slots, i.e., each user i $(i \in \mathcal{N})$ tries to find a solution to the following optimization problem (OPT_i):

$$\text{maximize} \;\; U_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}), \;\; \boldsymbol{\lambda}_i \in Q_i. \tag{12}$$

3 Game Formulation and Analyses

In this section, we formulate the considered scenario into a non-cooperative game among the multiple cloud users. By employing variational inequality theory, we analyze the existence of a Nash equilibrium solution set for the formulated game. And then we propose an iterative proximal algorithm to compute a Nash equilibrium. We also analyze the convergence of the proposed algorithm.

3.1 Game Formulation

Game theory studies the problems in which players try to maximize their utilities or minimize their disutilities. As described in [21], a non-cooperative game consists of a set of players, a set of strategies, and preferences over the set of strategies. In this paper, each cloud user is regarded as a player, i.e., the set of players is the n cloud users. The strategy set of player i $(i \in \mathcal{N})$ is the request profile set of user i, i.e., Q_i. Then the joint strategy set of all players is given by $Q = Q_1 \times \cdots \times Q_n$.

As mentioned before, all users are considered to be selfish and each user i $(i \in \mathcal{N})$ tries to maximize his/her own utility or minimize his/her disutility while ignoring the others. In view of (12), we can observe that user i's optimization problem is equivalent to

$$\begin{aligned} \text{minimize} \quad & f_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}) = \sum_{h=1}^{H} \big(P^h_i + w_i \delta^h \bar{T}^h - r\lambda^h_i\big), \\ s.t. \quad & (\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}) \in Q. \end{aligned} \tag{13}$$

The above formulated game can be formally defined by the tuple $G = \langle Q, \boldsymbol{f} \rangle$, where $\boldsymbol{f} = (f_1, \ldots, f_n)$. The aim of user i $(i \in \mathcal{N})$, given the other players' strategies $\boldsymbol{\lambda}_{-i}$, is to choose an $\boldsymbol{\lambda}_i \in Q_i$ such that his/her disutility function $f_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i})$ is minimized. That is to say, for each user i $(i \in \mathcal{N})$,

$$\boldsymbol{\lambda}^*_i \in \underset{\boldsymbol{\lambda}_i \in Q_i}{\arg\min}\, f_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}^*_{-i}), \;\; \boldsymbol{\lambda}^* \in Q. \tag{14}$$

At the Nash equilibrium, each player cannot further decrease its disutility by choosing a different strategy while the strategies of other players are fixed. The equilibrium strategy profile can be found when each player's strategy is the best response to the strategies of other players.

3.2 Billing Parameters Analysis

It is important to investigate the way the cloud provider decides load billing scheme. In our proposed model, the request charge changes according to the total load during different time slots. The cloud provider needs to decide the proper pricing parameters a and b. The reason lies in that if the per request charge (the cost of one task request) is too high, some users may refuse to use the cloud service, and choose to finish his/her tasks locally. On the contrary, if the charge is low, the aggregated requests may be more than enough, which could lead to low service quality (long task response time). In this paper, we assume that each user i $(i \in \mathcal{N})$ has a reservation value v_i. That is to say, cloud user i will prefer to use the cloud service if $U_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}) \geq v_i$ and refuse to use the service otherwise. If the cloud provider wants to appeal all n cloud users to use its service while charging relatively high, then it must guarantee that the obtained utility of each user i $(i \in \mathcal{N})$ is equal to his/her reservation value v_i, i.e., $U_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}) = v_i$ $(\forall i \in \mathcal{N})$, which implies that

$$\sum_{h=1}^{H} \big(r\lambda^h_i - P^h_i - w_i \delta^h \bar{T}^h\big) = v_i, \forall i \in \mathcal{N}. \tag{15}$$

Considering all users together, (15) is equivalent to

$$r\Lambda - P_T - w_\Sigma \sum_{h=1}^{H} \delta^h \bar{T}^h = \sum_{i=1}^{n} v_i, \tag{16}$$

where $\Lambda = \sum_{i=1}^{n} \Lambda_i$, $w_\Sigma = \sum_{i=1}^{n} w_i$, and $P_T = \sum_{i=1}^{n} \sum_{h=1}^{H} P^h_i$.

For the cloud provider, its objective is trying to decide proper pricing parameters a and b such that its net reward, i.e., the charge to all cloud users (P_T) minus its cost (e.g., energy cost and machine maintenance cost), is maximized. In this paper, we denote π as the net profit and γ_h the cost in time slot h. When total capacity μ is determined, γ_h is assumed to be constant. Then the cloud provider's problem is to try to maximize the value π. That is

$$\begin{aligned} \text{maximize} \quad & \pi = P_T(\boldsymbol{\lambda}) - \sum_{h=1}^{H} \gamma_h, \\ \text{s.t.} \quad & r\Lambda - P_T(\boldsymbol{\lambda}) - w_\Sigma \sum_{h=1}^{H} \delta^h \bar{T}^h = \sum_{i=1}^{n} v_i, \end{aligned} \tag{17}$$

$$\Lambda = \sum_{i=1}^{n} \Lambda_i = \sum_{h=1}^{H} \lambda^h_\Sigma, \tag{18}$$

$$\mu > \lambda^h_\Sigma \geq 0, \forall h \in \mathcal{H}. \tag{19}$$

The above optimization problem is equivalent to

$$\begin{aligned} \text{maximize} \quad & \pi = r\Lambda - w_\Sigma \sum_{h=1}^{H} \delta^h \bar{T}^h - \sum_{i=1}^{n} v_i - \sum_{h=1}^{H} \gamma_h, \\ \text{s.t.} \quad & \Lambda = \sum_{i=1}^{n} \Lambda_i = \sum_{h=1}^{H} \lambda_\Sigma^h, \\ & \mu > \lambda_\Sigma^h \geq 0, \forall h \in \mathcal{H}. \end{aligned} \tag{20}$$

Theorem 3.1. *For the cloud provider, the profit is maximized when the billing parameters (a and b) satisfy the constraint (17) and*

$$\lambda_\Sigma^h = \mu - \frac{(H\mu - \Lambda)\left(1 - \delta^{1/2}\right)\delta^{(h-1)/2}}{\left(1 - \delta^{H/2}\right)}, \tag{21}$$

where $h \in \mathcal{H}$.

Proof. We can maximize π in (20) by using the method of Lagrange multiplier, namely,

$$\frac{\partial \pi}{\partial \lambda_\Sigma^h} = -w_\Sigma \delta^h \frac{\partial \bar{T}^h}{\partial \lambda_\Sigma^h} = -\varphi,$$

where φ is the Lagrange multiplier. That is,

$$\frac{w_\Sigma \delta^h}{\left(\mu - \lambda_\Sigma^h\right)^2} = \varphi,$$

for all $1 \leq h \leq H$, and $\sum_{h=1}^{H} \lambda_\Sigma^h = \Lambda$. After some algebraic calculation, we have

$$\varphi = \frac{w_\Sigma \delta \left(1 - \delta^{H/2}\right)^2}{(H\mu - \Lambda)^2 \left(1 - \delta^{1/2}\right)^2}.$$

Then we can obtain

$$\lambda_\Sigma^h = \mu - \frac{(H\mu - \Lambda)(1 - \delta^{1/2})\delta^{(h-1)/2}}{(1 - \delta^{H/2})},$$

and the result follows. □

Note that the obtained result (21) must satisfy the constraint (19), that is to say,

$$\begin{cases} \mu - \frac{(H\mu - \Lambda)(1-\delta^{1/2})\delta^{(H-1)/2}}{(1-\delta^{H/2})} \geq 0, & h = H; \\ \mu - \frac{(H\mu - \Lambda)(1-\delta^{1/2})}{(1-\delta^{H/2})} < \mu, & h = 1; \\ H\mu - \Lambda > 0. \end{cases} \tag{22}$$

We obtain

$$\begin{cases} \mu \leq \frac{c\Lambda}{cH-1}, \\ H\mu > \Lambda, \end{cases} \tag{23}$$

where

$$c = \frac{(1 - \delta^{1/2})\delta^{(H-1)/2}}{1 - \delta^{H/2}}.$$

Then we have

$$\frac{H}{\Lambda} < \mu \leq \frac{\Lambda}{H - 1/c}, \tag{24}$$

where

$$c = \frac{(1 - \delta^{1/2})\delta^{(H-1)/2}}{1 - \delta^{H/2}}.$$

As mentioned before, we assume that the aggregated requests do not exceed the capacity of all the servers, i.e., $H\mu > \Lambda$. In addition, if $\mu > \frac{\Lambda}{H-1/c}$, it is possible to shutdown some servers such that μ satisfies the constraint (24), which can also save energy cost. Therefore, in this paper, we assume that the total processing capacity μ satisfies constraint (24).

From Theorem 3.1, we know that if the cloud provider wants to appeal all the n cloud users to use its service, then proper pricing parameters a and b can be selected to satisfy constraint (17). Specifically, if b (a) is given, and a (b) is higher than the computed value from (17), then there exist some users who refuse to use the cloud service, because their obtained utilities are less than their reservation values.

3.3 Nash Equilibrium Analysis

In this section, we analyze the existence of Nash equilibrium for the formulated game $G = \langle Q, \boldsymbol{f} \rangle$ and prove the existence problem by employing variational inequality theory. Then we propose an iterative proximal algorithm. The convergence of the proposed algorithm is also analyzed. Before address the problem, we show three important properties presented in Theorems 3.2, 3.3, and 3.4, which are helpful to prove the existence of Nash equilibrium for the formulated game.

Theorem 3.2. *For each cloud user i ($i \in \mathcal{N}$), the set Q_i is convex and compact, and each disutility function $f_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i})$ is continuously differentiable in $\boldsymbol{\lambda}_i$. For each fixed tuple $\boldsymbol{\lambda}_{-i}$, the disutility function $f_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i})$ is convex in $\boldsymbol{\lambda}_i$ over the set Q_i.*

Proof. It is obvious that the statements in the first part of above theorem hold. We only need to prove the convexity of $f_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i})$ in $\boldsymbol{\lambda}_i$ for every fixed $\boldsymbol{\lambda}_{-i}$. This can be achieved by proving that the Hessian matrix of $f_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i})$ is positive semidefinite [9], [29]. Since $f_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}) = \sum_{h=1}^{H} (P_i^h + w_i \delta^h \bar{T}^h - r\lambda_i^h)$, we have

$$\begin{aligned} \nabla_{\boldsymbol{\lambda}_i} f_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}) &= \left[\frac{\partial f_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i})}{\partial \lambda_i^h}\right]_{h=1}^{H} \\ &= \left(\frac{\partial f_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i})}{\partial \lambda_i^1}, \ldots, \frac{\partial f_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i})}{\partial \lambda_i^H}\right). \end{aligned}$$

and the Hessian matrix is expressed as

$$\begin{aligned} \nabla^2_{\boldsymbol{\lambda}_i} f_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}) &= \operatorname{diag}\left\{\left[\frac{\partial^2 f_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i})}{\partial (\lambda_i^h)^2}\right]_{h=1}^{H}\right\} \\ &= \operatorname{diag}\left\{\left[2a\left(2\lambda_\Sigma^h + \lambda_i^h\right) + \frac{2w_i\delta^h}{\left(\mu - \lambda_\Sigma^h\right)^3}\right]_{h=1}^{H}\right\}. \end{aligned} \tag{25}$$

Obviously, the diagonal matrix in (25) has all diagonal elements being positive. Thus, the Hessian matrix of $f_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i})$ is positive semidefinite and the result follows. The theorem is proven. □

Theorem 3.3. *The Nash equilibrium of the formulated game G is equivalent to the solution of the variational inequality (VI) problem, denoted by $\mathrm{VI}(Q, \mathbf{F})$, where $Q = Q_1 \times \cdots \times Q_n$ and*

$$\mathbf{F}(\boldsymbol{\lambda}) = (\mathbf{F}_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}))_{i=1}^n, \tag{26}$$

with

$$\mathbf{F}_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}) = \nabla_{\boldsymbol{\lambda}_i} f_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}). \tag{27}$$

Proof. According to [30, Property 4.1], we know that the above claim follows if two conditions are satisfied. First, for each user i ($i \in \mathcal{N}$), the strategy set Q_i is closed and convex. Second, for every fixed $\boldsymbol{\lambda}_{-i}$, the disutility function $f_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i})$ is continuously differentiable and convex in $\boldsymbol{\lambda}_i \in Q_i$. By Theorem 3.2, it is easy to know that both the mentioned two conditions are satisfied in the formulated game G. Thus, the result follows. □

Theorem 3.4. *If both matrices $\mathcal{M}_1$ and $\mathcal{M}_2$ are semidefinite, then the matrix $\mathcal{M}_3 = \mathcal{M}_1 + \mathcal{M}_2$ is also semidefinite.*

Proof. As mentioned above, both matrices $\mathcal{M}_1$ and $\mathcal{M}_2$ are semidefinite. Then we have $\forall \boldsymbol{x}$

$$\boldsymbol{x}^T \mathcal{M}_1 \boldsymbol{x} \geq 0 \text{ and } \boldsymbol{x}^T \mathcal{M}_2 \boldsymbol{x} \geq 0.$$

We obtain $\forall \boldsymbol{x}$,

$$\boldsymbol{x}^T \mathcal{M}_3 \boldsymbol{x} = \boldsymbol{x}^T \mathcal{M}_1 \boldsymbol{x} + \boldsymbol{x}^T \mathcal{M}_2 \boldsymbol{x} \geq 0.$$

Thus, we can conclude that $\mathcal{M}_3$ is semidefinite and the result follows. □

Recall that the objective of this section is to study the existence of Nash equilibrium for the formulated game $G = \langle Q, \boldsymbol{f} \rangle$ in (13). In the next theorem, we prove that if several conditions are satisfied, the existence of such Nash equilibrium is guaranteed.

Theorem 3.5. *If $\max_{i=1,\ldots,n}(w_i) \leq 1/n$, there exists a Nash equilibrium solution set for the formulated game $G = \langle Q, \boldsymbol{f} \rangle$.*

Proof. Based on Theorem 3.3, the proof of this theorem follows if we can show that the formulated variational inequality problem $\mathrm{VI}(Q, \mathbf{F})$ in Theorem 3.3 possesses a solution set. According to [30, Theorem 4.1], the $\mathrm{VI}(Q, \mathbf{F})$ admits a solution set if the mapping $\mathbf{F}(\boldsymbol{\lambda})$ is monotone over Q, since the feasible set Q is compact and convex.

To prove the monotonicity of $\mathbf{F}(\boldsymbol{\lambda})$, it suffices to show that for any $\boldsymbol{\lambda}$ and $\boldsymbol{s}$ in Q,

$$(\boldsymbol{\lambda} - \boldsymbol{s})^T (\mathbf{F}(\boldsymbol{\lambda}) - \mathbf{F}(\boldsymbol{s})) \geq 0,$$

namely,

$$\sum_{h=1}^{H} \sum_{i=1}^{n} \left(\lambda_i^h - s_i^h\right)\left(\nabla_{\lambda_i^h} f_i(\boldsymbol{\lambda}) - \nabla_{s_i^h} f_i(\boldsymbol{s})\right) \geq 0. \tag{28}$$

Let $\boldsymbol{\lambda}^h = (\lambda_1^h, \ldots, \lambda_n^h)^T$ and $\boldsymbol{s}^h = (s_1^h, \ldots, s_n^h)^T$, then we can write (28) as

$$\sum_{h=1}^{H} \left(\boldsymbol{\lambda}^h - \boldsymbol{s}^h\right)\left(\nabla_{\boldsymbol{\lambda}^h} f^h\left(\boldsymbol{\lambda}^h\right) - \nabla_{\boldsymbol{s}^h} f^h\left(\boldsymbol{s}^h\right)\right) \geq 0, \tag{29}$$

where

$$f^h(\boldsymbol{\lambda}^h) = \sum_{i=1}^{n} \left(P_i^h + w_i \delta^h \bar{T}^h - r\lambda_i^h\right),$$

and

$$\nabla_{\boldsymbol{\lambda}^h} f^h(\boldsymbol{\lambda}^h) = (\nabla_{\lambda_1^h} f^h(\boldsymbol{\lambda}^h), \ldots, \nabla_{\lambda_n^h} f^h(\boldsymbol{\lambda}^h))^T.$$

We can observe that if

$$(\boldsymbol{\lambda}^h - \boldsymbol{s}^h)(\boldsymbol{g}^h(\boldsymbol{\lambda}^h) - \boldsymbol{g}^h(\boldsymbol{s}^h)) \geq 0, \ \forall h \in \mathcal{H}, \tag{30}$$

where $\boldsymbol{g}^h\left(\boldsymbol{\lambda}^h\right) = \nabla_{\boldsymbol{\lambda}^h} f^h(\boldsymbol{\lambda}^h)$, then equation (29) holds.

Recall the definition of a monotone mapping, we can find that (30) holds if the mapping $\boldsymbol{g}^h\left(\boldsymbol{\lambda}^h\right)$ is monotone. With reference to [30], the condition in (30) is equivalent to proving the Jacobian matrix of $\boldsymbol{g}^h\left(\boldsymbol{\lambda}^h\right)$, denoted by $\mathbf{G}\left(\boldsymbol{\lambda}^h\right)$, is positive semidefinite.

After some algebraic manipulation, we can write the (i, j)th element of $\mathbf{G}\left(\boldsymbol{\lambda}^h\right)$ as

$$\left[\mathbf{G}\left(\boldsymbol{\lambda}^h\right)\right]_{i,j} = \begin{cases} 2a\left(2\lambda_{\Sigma}^h + \lambda_i^h\right) + \frac{2w_i\delta^h}{\left(\mu - \lambda_{\Sigma}^h\right)^3}, & \text{if } i = j; \\ 2a\left(\lambda_{\Sigma}^h + \lambda_i^h\right) + \frac{2w_i\delta^h}{\left(\mu - \lambda_{\Sigma}^h\right)^3}, & \text{if } i \neq j. \end{cases}$$

Since the matrix $\mathbf{G}\left(\boldsymbol{\lambda}^h\right)$ may not be symmetric, we can prove its positive semidefiniteness by showing that the symmetric matrix

$$\begin{aligned} \mathbf{G}(\boldsymbol{\lambda}^h) + \mathbf{G}(\boldsymbol{\lambda}^h)^T = \\ 2a \underbrace{\left(\boldsymbol{\lambda}^h \mathbf{1}_{n\times 1}^T + \mathbf{1}_{n\times 1}\left(\boldsymbol{\lambda}^h\right)^T + 2\lambda_{\Sigma}^h \mathbf{1}_{n\times n} + 2\lambda_{\Sigma}^h \mathbf{E}_n\right)}_{\mathcal{M}_1} \\ + 2a\sigma \underbrace{\left(\boldsymbol{w}\mathbf{1}_{n\times 1}^T + \mathbf{1}_{n\times 1}\boldsymbol{w}^T\right)}_{\mathcal{M}_2} \end{aligned}$$

is positive semidefinite [31], where

$$\sigma = \frac{\delta^h}{a\left(\mu - \lambda_{\Sigma}^h\right)^3},$$

$\boldsymbol{w} = (w_1, \ldots, w_n)^T$, $\mathbf{1}_{r\times s}$ is a $r \times s$ matrix with every element of 1, and $\mathbf{E}_n$ is an identity matrix. This is equivalent to showing that the smallest eigenvalue of this matrix is non-negative.

With referring to [9], [31], we obtain the two non-zero eigenvalues of $\mathcal{M}_1$ as follows:

$$\eta_{\mathcal{M}_1}^1 = (n+3)\lambda_{\Sigma}^h + \sqrt{n\sum_{i=1}^{n}\left(\lambda_i^h + \lambda_{\Sigma}^h\right)^2},$$

$$\eta_{\mathcal{M}_1}^2 = (n+3)\lambda_{\Sigma}^h - \sqrt{n\sum_{i=1}^{n}\left(\lambda_i^h + \lambda_{\Sigma}^h\right)^2}.$$

Let

$$A(\boldsymbol{\lambda}^h) = (n+3)\lambda_{\Sigma}^h,$$

and

$$B(\boldsymbol{\lambda}^h) = \sqrt{n\sum_{i=1}^{n}\left(\lambda_i^h + \lambda_{\Sigma}^h\right)^2},$$

and $\eta_{\min}$ be the minimal eigenvalue of matrix $\mathcal{M}_1$. Then, we have $\eta_{\min} = \min\{A(\boldsymbol{\lambda}^h) - B(\boldsymbol{\lambda}^h), 2\lambda_{\Sigma}^h\}$. Furthermore, we can derive that

$$\begin{aligned}\left(A(\boldsymbol{\lambda}^h)\right)^2 - \left(B(\boldsymbol{\lambda}^h)\right)^2 &= (4n+9)\left(\lambda_{\Sigma}^h\right)^2 - n\sum_{i=1}^{n}\left(\lambda_i^h\right)^2 \\ &\geq n\left(\left(\sum_{i=1}^{n}\lambda_i^h\right)^2 - \sum_{i=1}^{n}\left(\lambda_i^h\right)^2\right) \geq 0.\end{aligned}$$

Hence, we can obtain $\eta_{\min} \geq 0$ and conclude that $\mathcal{M}_1$ is semidefinite. Similar to the semidefinite proof of $\mathcal{M}_1$, we can also obtain that if $\max_{i=1,\ldots,n}(w_i) \leq 1/n$, then $\mathcal{M}_2$ is semidefinite. By Theorem 3.4, we can conclude that the matrix $\mathbf{G}(\boldsymbol{\lambda}^h)$ is semidefinite, and the result follows. □

3.4 An Iterative Proximal Algorithm

Once we have established that the Nash equilibria of the formulated game $\mathbf{G} = \langle Q, \boldsymbol{f}\rangle$ exists, we are interested in obtaining a suitable algorithm to compute one of these equilibria with minimum information exchange between the multiple users and the cloud provider.

Note that we can further rewrite the optimization problem (13) as follows:

$$\begin{aligned}\text{minimize} \quad & f_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{\Sigma}) = \sum_{h=1}^{H}\left(P_i^h + w_i\delta^h\bar{T}^h - r\lambda_i^h\right), \\ \text{s.t.} \quad & \boldsymbol{\lambda}_i \in Q_i,\end{aligned} \tag{31}$$

where $\boldsymbol{\lambda}_{\Sigma}$ denotes the aggregated request profile of all users over the H future time slots, i.e., $\boldsymbol{\lambda}_{\Sigma} = \sum_{i=1}^{n}\boldsymbol{\lambda}_i$. From (31), we can see that the calculation of the disutility function of each individual user only requires the knowledge of the aggregated request profile of all users ($\boldsymbol{\lambda}_{\Sigma}$) rather than that the specific individual request profile of all other users ($\boldsymbol{\lambda}_{-i}$) , which can bring about two advantages. On the one hand, it can reduce communication traffic between users and the cloud provider. On the other hand, it can also keep privacy for each individual user to certain extent, which is seriously considered by many cloud users.

Since all users are considered to be selfish and try to minimize their own disutilities while ignoring the others. It is natural to consider an iterative algorithm where, at every iteration k, each individual user i ($\forall i \in \mathcal{N}$) updates his/her strategy to minimize his/her own disutility function $f_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{\Sigma})$. However, following [30, Theorem 4.2], it is not difficult to show that their convergence cannot be guaranteed in our case if the users are allowed to simultaneously update their strategies according to (31).

To overcome this issue, we consider an iterative proximal algorithm, which is based on the proximal decomposition Algorithm 4.2 [30]. The proposed algorithm is guaranteed to converge to a Nash equilibrium under some additional constraints on the parameters of the algorithm. With reference to [30], consider the regularized game in which each user i ($i \in \mathcal{N}$) tries to solve the following optimization problem:

$$\begin{aligned}\text{minimize} \quad & f_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{\Sigma}) + \frac{\tau}{2}\left\|\boldsymbol{\lambda}_i - \bar{\boldsymbol{\lambda}}_i\right\|^2, \\ \text{s.t.} \quad & \boldsymbol{\lambda}_i, \bar{\boldsymbol{\lambda}}_i \in Q_i.\end{aligned} \tag{32}$$

That is to say, when given the aggregated requests, we must find a strategy vector $\boldsymbol{\lambda}_i^*$ for user i ($i \in \mathcal{N}$) such that

$$\boldsymbol{\lambda}_i^* \in \arg\min_{\boldsymbol{\lambda}_i \in Q_i}\left\{f_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{\Sigma}) + \frac{\tau}{2}\left\|\boldsymbol{\lambda}_i - \bar{\boldsymbol{\lambda}}_i\right\|^2\right\}, \tag{33}$$

where $\tau(\tau > 0)$ is a regularization parameter and may guarantee the convergence of the best-response algorithm [30, Corollary 4.1] if it is large enough. The idea is formalized in Algorithm 1.

Algorithm 1. Iterative Proximal Algorithm (IPA)

Input: Strategy set of all users: Q, ϵ.
Output: Request configuration: $\boldsymbol{\lambda}$.

1: *Initialization*: Each cloud user i ($i \in \mathcal{N}$) randomly choose a $\boldsymbol{\lambda}_i^{(0)} \in Q_i$ and set $\bar{\boldsymbol{\lambda}}_i \leftarrow \mathbf{0}$. Set $S_c \leftarrow \mathcal{N}$, $S_l \leftarrow \emptyset$, and $k \leftarrow 0$.
2: **while** ($S_c \neq S_l$) **do**
3: Set $S_l \leftarrow S_c$.
4: **while** ($\left\|\boldsymbol{\lambda}^{(k)} - \boldsymbol{\lambda}^{(k-1)}\right\| > \epsilon$) **do**
5: **for** (each cloud user $i \in S_c$) **do**
6: Receive $\boldsymbol{\lambda}_{\Sigma}^{(k)}$ from the cloud provider and compute $\boldsymbol{\lambda}_i^{(k)}$ as follows (by Algorithm 2):
7: $\boldsymbol{\lambda}_i^{(k+1)} \leftarrow \arg\min_{\boldsymbol{\lambda}_i \in Q_i}\left\{f_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{\Sigma}^{(k)}) + \frac{\tau}{2}\left\|\boldsymbol{\lambda}_i - \bar{\boldsymbol{\lambda}}_i\right\|^2\right\}$.
8: Send the updated strategy to the cloud provider.
9: **end for**
10: **if** (Nash equilibrium is reached) **then**
11: Each user i ($i \in S_c$) updates his/her centroid $\bar{\boldsymbol{\lambda}}_i \leftarrow \boldsymbol{\lambda}_i^{(k)}$.
12: **end if**
13: Set $k \leftarrow k+1$.
14: **end while**
15: **for** (each user $i \in S_c$) **do**
16: **if** $U_i(\boldsymbol{\lambda}_i^{(k)}, \boldsymbol{\lambda}_{\Sigma}^{(k)}) < v_i$ **then**
17: Set $\boldsymbol{\lambda}_i^{(k)} \leftarrow \mathbf{0}$, and $S_c \leftarrow S_c - \{i\}$.
18: **end for**
19: **end while**
20: **return** $\boldsymbol{\lambda}^{(k)}$.

Theorem 3.6. *There exists a constant τ_0 such that if $\tau > \tau_0$, then any sequence $\{\boldsymbol{\lambda}_i^{(k)}\}_{k=1}^{\infty}$ ($i \in S_c$) generated by the IPA algorithm converges to a Nash equilibrium.*

Proof. We may note that Algorithm 1 converges if the inner while loop (Steps 4-14) can be terminated. Therefore, if we can prove that Steps 4-14 converges, the result follows. In practice, Steps 4-14 in Algorithm 1 is a developed instance of the proximal decomposition algorithm, which is presented in Algorithm 4.2 [30] for the

variational inequality problem. Next, we rewrite the convergence conditions exploiting the equivalence between game theory and variational inequality ([30, chapter 4.2]). Given $f_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i})$ defined as in equation (13), Algorithm 1 convergences if the following two conditions are satisfied. (1) The Jacobian matrix of $\mathbf{F}$ is positive semidefinite ([30, Theorem 4.3]). We denote the Jacobian by $\mathrm{J}\mathbf{F}(\boldsymbol{\lambda}) = (\mathrm{J}_{\boldsymbol{\lambda}_j}\mathbf{F}_i(\boldsymbol{\lambda}))_{i,j=1}^{n}$, where $\mathrm{J}_{\boldsymbol{\lambda}_j}\mathbf{F}_i(\boldsymbol{\lambda}) = (\nabla_{\boldsymbol{\lambda}_j} f_i(\boldsymbol{\lambda}))_{j=1}^{n}$, which is the partial Jacobian matrix of $\mathbf{F}_i$ with respect to λ_j vector. (2) The $n \times n$ matrix $\boldsymbol{\Upsilon}_{\mathbf{F},\tau} = \boldsymbol{\Upsilon}_{\mathbf{F}} + \tau \mathbf{E}_n$ is a P-matrix ([30, Corollary 4.1]), where

$$[\boldsymbol{\Upsilon}_{\mathbf{F}}]_{ij} = \begin{cases} \alpha_i^{\min}, & \text{if } i = j; \\ -\beta_{ij}^{\max}, & \text{if } i \neq j; \end{cases}$$

with

$$\alpha_i^{\min} = \inf_{\boldsymbol{\lambda} \in Q} \eta_{\min}\big(\mathrm{J}_{\boldsymbol{\lambda}_i}\mathbf{F}_i(\boldsymbol{\lambda})\big),$$

and

$$\beta_{ij}^{\max} = \sup_{\boldsymbol{\lambda} \in Q} \eta_{\min}\Big(\mathrm{J}_{\boldsymbol{\lambda}_j}\mathbf{F}_i(\boldsymbol{\lambda})\Big),$$

and $\eta_{\min}(\mathbf{A})$ denoting the smallest eigenvalue of $\mathbf{A}$. After some algebraic manipulation, we can write the block elements of $\mathrm{J}\mathbf{F}(\boldsymbol{\lambda})$ as

$$\begin{aligned} \mathrm{J}_{\boldsymbol{\lambda}_i}\mathbf{F}_i(\boldsymbol{\lambda}) &= \nabla^2_{\boldsymbol{\lambda}_i} f_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{\Sigma}) \\ &= \operatorname{diag}\left\{ \left[2a\big(2\lambda_{\Sigma}^h + \lambda_i^h\big) + \frac{2w_i\delta^h}{\big(\mu - \lambda_{\Sigma}^h\big)^3} \right]_{h=1}^{H} \right\}, \end{aligned}$$

and

$$\begin{aligned} \mathrm{J}_{\boldsymbol{\lambda}_j}\mathbf{F}_i(\boldsymbol{\lambda}) &= \nabla^2_{\boldsymbol{\lambda}_i\boldsymbol{\lambda}_j} f_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{\Sigma}) \\ &= \operatorname{diag}\left\{ \left[2a\big(\lambda_{\Sigma}^h + \lambda_i^h\big) + \frac{2w_i\delta^h}{\big(\mu - \lambda_{\Sigma}^h\big)^3} \right]_{h=1}^{H} \right\}, \end{aligned}$$

for $i \neq j$ $(i, j \in \mathcal{N})$.

Next, we show that the above conditions (1) and (2) hold, respectively. By Theorem 3.2, we know that the vector function $\mathbf{F}(\boldsymbol{\lambda})$ is monotone on Q, which implies that $\mathrm{J}\mathbf{F}(\boldsymbol{\lambda})$ is semidefinite. On the other hand, considering $\mathrm{J}_{\boldsymbol{\lambda}_i}\mathbf{F}_i(\boldsymbol{\lambda})$, we have $\alpha_i^{\min} > 0$,

Let

$$L^h\big(\lambda_i^h, \boldsymbol{\lambda}_{-i}^h\big) = 2a\big(\lambda_{\Sigma}^h + \lambda_i^h\big) + \frac{2w_i\delta^h}{\big(\mu - \lambda_{\Sigma}^h\big)^3}.$$

Then, we have $\frac{\partial L^h}{\partial \lambda_i^h} > 0$. As mentioned before, λ_{Σ}^h $(\forall h \in \mathcal{H})$ does not exceed the total processing capacity of all servers μ. We assume that $\lambda_{\Sigma}^h \leq (1-\varepsilon)\mu$, where ε is a small positive constant. Then we can conclude that

$$L^h(\lambda_i^h, \boldsymbol{\lambda}_{-i}^h) \leq 4a(1-\varepsilon)\mu + \frac{2w_{\max}\delta^h}{(\varepsilon\mu)^3},$$

where $w_{\max} = \max_{i=1,\ldots,n}\{w_i\}$.

Hence, if

$$\tau_0 \geq (n-1)\left(4a(1-\varepsilon)\mu + \frac{2w_{\max}\delta^H}{(\varepsilon\mu)^3}\right),$$

then

$$\beta_{ij}^{\max} = \sup_{\boldsymbol{\lambda} \in Q}\Big\|\mathrm{J}_{\boldsymbol{\lambda}_j}\mathbf{F}_i(\boldsymbol{\lambda})\Big\| \leq \tau_0.$$

Then, it follows from [30, Properties 4.3] that, if τ is chosen as in Theorem 3.6, the matrix $\boldsymbol{\Upsilon}_{\mathbf{F},\tau}$ is a P-matrix, and the result follows. □

Next, we focus on the calculation for the optimization problem in (33). Let

$$L_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{\Sigma}) = f_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{\Sigma}) + \frac{\tau}{2}\big\|\boldsymbol{\lambda}_i - \bar{\boldsymbol{\lambda}}_i\big\|^2. \tag{34}$$

Then, we have to minimize $L_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{\Sigma})$. Note that the variable in (34) is only $\boldsymbol{\lambda}_i$, therefore, we can rewrite (34) as

$$L_i(\boldsymbol{\lambda}_i, \boldsymbol{\kappa}_{\Sigma}) = f_i(\boldsymbol{\lambda}_i, \boldsymbol{\kappa}_{\Sigma}) + \frac{\tau}{2}\big\|\boldsymbol{\lambda}_i - \bar{\boldsymbol{\lambda}}_i\big\|^2, \tag{35}$$

where $\boldsymbol{\kappa}_{\Sigma} = \boldsymbol{\lambda}_{\Sigma} - \boldsymbol{\lambda}_i$. We denote R_i the constraint of user i, i.e.,

$$R_i = \lambda_i^1 + \lambda_i^2 + \cdots + \lambda_i^H = \Lambda_i,$$

and try to minimize $L_i(\boldsymbol{\lambda}_i, \boldsymbol{\kappa}_{\Sigma})$ by using the method of Lagrange multiplier, namely,

$$\frac{\partial L_i}{\partial \lambda_i^h} = \phi \frac{\partial R_i}{\partial \lambda_i^h} = \phi,$$

for all $1 \leq h \leq H$, where ϕ is a Lagrange multiplier. Notice that

$$\frac{\partial P_i^h}{\partial \lambda_i^h} = a\big(2\big(\lambda_i^h + \kappa_{\Sigma}^h\big)\lambda_i^h + \big(\lambda_i^h + \kappa_{\Sigma}^h\big)^2\big) + b,$$

and

$$\frac{\partial \bar{T}^h}{\partial \lambda_i^h} = \frac{1}{\big(\mu - \kappa_{\Sigma}^h - \lambda_i^h\big)^2}.$$

We obtain

$$\begin{aligned} \frac{\partial L_i}{\partial \lambda_i^h} &= \frac{\partial P_i^h}{\partial \lambda_i^h} + w_i\delta^h \frac{\partial \bar{T}^h}{\partial \lambda_i^h} - r + \tau\big(\lambda_i^h - \bar{\lambda}_i^h\big) \\ &= a\big(2\big(\lambda_i^h + \kappa_{\Sigma}^h\big)\lambda_i^h + \big(\lambda_i^h + \kappa_{\Sigma}^h\big)^2\big) \\ &\quad + b + \frac{w_i\delta^h}{\big(\mu - \kappa_{\Sigma}^h - \lambda_i^h\big)^2} - r + \tau\big(\lambda_i^h - \bar{\lambda}_i^h\big) = \phi. \end{aligned} \tag{36}$$

Denote $Y_i^h(\lambda_i^h, \kappa_{\Sigma}^h)$ as the first order of $L_i(\boldsymbol{\lambda}_i, \boldsymbol{\kappa}_{\Sigma})$ on λ_i^h. Then, we have

$$\begin{aligned} Y_i^h\big(\lambda_i^h, \kappa_{\Sigma}^h\big) &= a\big(2\big(\lambda_i^h + \kappa_{\Sigma}^h\big)\lambda_i^h + \big(\lambda_i^h + \kappa_{\Sigma}^h\big)^2\big) + b \\ &\quad + \frac{w_i\delta^h}{\big(\mu - \kappa_{\Sigma}^h - \lambda_i^h\big)^2} - r + \tau\big(\lambda_i^h - \bar{\lambda}_i^h\big). \end{aligned} \tag{37}$$

Since the first order of $Y_i^h(\lambda_i^h, \kappa_\Sigma^h)$ is

$$\frac{\partial Y_i^h}{\partial \lambda_i^h} = \frac{\partial^2 L_i}{\partial \left(\lambda_i^h\right)^2} = 2a\left(3\lambda_i^h + 2\kappa_\Sigma^h\right) + \frac{2w_i\delta^h}{\left(\mu - \kappa_\Sigma^h - \lambda_i^h\right)^3} + \tau > 0, \tag{38}$$

we can conclude that $Y_i^h(\lambda_i^h, \kappa_\Sigma^h)$ is an increasing positive function on λ_i^h. Based on above derivations, we propose an algorithm to calculate $\boldsymbol{\lambda}_i$ ($i \in \mathcal{N}$), which is motivated by [28].

Algorithm 2. Calculate_$\boldsymbol{\lambda}_i(\varepsilon, \mu, a, b, r, \tau, \boldsymbol{\lambda}_i, \boldsymbol{\lambda}_\Sigma, \Lambda_i)$

Input: $\varepsilon, \mu, a, b, r, \tau, \boldsymbol{\lambda}_i, \boldsymbol{\lambda}_\Sigma, \Lambda_i$
Output: $\boldsymbol{\lambda}_i$.
1: *Initialization*: Let inc be a relative small positive constant. Set $\boldsymbol{\kappa}_\Sigma \leftarrow \boldsymbol{\lambda}_\Sigma - \boldsymbol{\lambda}_i$, $\boldsymbol{\lambda}_i \leftarrow \mathbf{0}$, and $\phi \leftarrow 0$.
2: **while** $(\lambda_i^1 + \lambda_i^2 + \cdots + \lambda_i^H < \Lambda_i)$ **do**
3: Set $mid \leftarrow \phi + inc$, and $\phi \leftarrow mid$.
4: **for** (each time slot $h \in \mathcal{H}$) **do**
5: $\lambda_i^h \leftarrow$ Calculate_$\lambda_i^h(\varepsilon, \mu, a, b, r, \tau, \kappa_\Sigma^h, \phi)$.
6: **end for**
7: Set $inc \leftarrow 2 \times inc$.
8: **end while**
9: Set $lb \leftarrow 0$ and $ub \leftarrow \phi$.
10: **while** $(ub - lb > \epsilon)$ **do**
11: Set $mid \leftarrow (ub + lb)/2$, and $\phi \leftarrow mid$.
12: **for** (each time slot $h \in \mathcal{H}$) **do**
13: $\lambda_i^h \leftarrow$ Calculate_$\lambda_i^h(\varepsilon, \mu, a, b, r, \tau, \kappa_\Sigma^h, \phi)$.
14: **if** $(\lambda_i^1 + \lambda_i^2 + \cdots + \lambda_i^H < \Lambda_i)$ **then**
15: Set $lb \leftarrow mid$.
16: **else**
17: Set $ub \leftarrow mid$.
18: **end if**
19: **end for**
20: **end while**
21: Set $\phi \leftarrow (ub + lb)/2$.
22: **for** (each time slot $h \in \mathcal{H}$) **do**
23: $\lambda_i^h \leftarrow$ Calculate_$\lambda_i^h(\varepsilon, \mu, a, b, r, \tau, \kappa_\Sigma^h, \phi)$.
24: **end for**
25: **return** $\boldsymbol{\lambda}_i$.

Algorithm 3. Calculate_$\lambda_i^h(\varepsilon, \mu, a, b, r, \tau, \kappa_\Sigma^h, \phi)$

Input: $\varepsilon, \mu, a, b, r, \tau, \kappa_\Sigma^h, \phi$.
Output: λ_i^h.
1: *Initialization*: Set $ub \leftarrow (1 - \varepsilon)\mu - \kappa_\Sigma^h$, and $lb \leftarrow 0$.
2: **while** $(ub - lb > \epsilon)$ **do**
3: Set $mid \leftarrow (ub + lb)/2$, and $\lambda_i^h \leftarrow mid$.
4: **if** $(Y_i^h(\lambda_i^h, \kappa_\Sigma^h) < \phi)$ **then**
5: Set $lb \leftarrow mid$.
6: **else**
7: Set $ub \leftarrow mid$.
8: **end if**
9: **end while**
10: Set $\lambda_i^h \leftarrow (ub + lb)/2$.
11: **return** λ_i^h.

Given $\varepsilon, \mu, a, b, r, \tau, \boldsymbol{\lambda}_i, \boldsymbol{\lambda}_\Sigma$, and Λ_i, our optimal request configuration algorithm to find $\boldsymbol{\lambda}_i$ is given in Algorithm 2. The algorithm uses another subalgorithm Calculate_λ_i^h described in Algorithm 3, which, given $\varepsilon, \mu, a, b, r, \tau, \kappa_\Sigma^h$, and ϕ, finds λ_i^h satisfies (36).

The key observation is that the left-hand side of (36), i.e., (37), is an increasing function of λ_i^h (see (38)). Therefore, given ϕ, we can find λ_i^h by using the binary search method in certain interval $[lb, ub]$ (Steps 2-9 in Algorithm 3). We set lb simply as 0. For ub, as mentioned in Theorem 3.6,

$$\lambda_i^h \leq (1 - \varepsilon)\mu,$$

where ε is a relative small positive constant. Therefore, in this paper, ub is set in Step 1 based on the above discussion. The value of ϕ can also be found by using the binary search method (Steps 10-20 in Algorithm 2). The search interval $[lb, ub]$ for ϕ is determined as follows. We set lb simply as 0. As for ub, we notice that the left-hand side of (36) is an increasing function of λ_i^h. Then, we set an increment variable inc, which is initialized as a relative small positive constant and repeatedly doubled (Step 7). The value of inc is added to ϕ to increase ϕ until the sum of λ_i^h ($h \in \mathcal{H}$) found by Calculate_λ_i^h is at least Λ_i (Steps 2-8). Once $[lb, ub]$ is decided, ϕ can be searched based on the fact that $Y_i^h(\lambda_i^h, \boldsymbol{\lambda}_{-i}^h)$ is an increasing function of λ_i^h. After ϕ is determined (Step 21), $\boldsymbol{\lambda}_i$ can be computed (Steps 22-24).

Finally, we can describe the proposed iterative proximal algorithm as follows. At the beginning, each cloud user i ($i \in \mathcal{N}$) sends his/her weight value (w_i) and total task request (Λ_i) to the cloud provider. Then the cloud provider computes τ as in Theorem 3.6 according to the aggregated information and chooses proper parameters a and b such that constraint (17) is satisfied. After this, the cloud provider puts the computed load billing parameters a and b into public information exchange module. Then, at each iteration k, the cloud provider broadcasts a synchronization signal and the current aggregated request profile $\boldsymbol{\lambda}_\Sigma^{(k)}$. Within iteration k, each user receives the aggregated profile $\boldsymbol{\lambda}_\Sigma^{(k)}$ and computes his/her strategy by solving its own optimization problem in (32), and then sends the newly updated strategy to the cloud provider. Last, as indicated in Steps 10-12 of Algorithm 1, the cloud provider checks whether the Nash equilibrium has been achieved and if so, it broadcasts a signal to inform all users to update their centroid $\bar{\boldsymbol{\lambda}}_i$. It also checks whether all cloud users' strategies are unchanged and if so, it informs all users to choose whether they still prefer to the cloud service due to their reserved values. This process continues until the set of the remaining cloud users and their corresponding strategies are kept fixed. In this paper, we assume that the strategies of all cloud users are unchanged if $\|\boldsymbol{\lambda}^{(k)} - \boldsymbol{\lambda}^{(k-1)}\| \leq \epsilon$, where $\boldsymbol{\lambda}^{(k)} = (\boldsymbol{\lambda}_i^{(k)})_{i=1}^n$ with $\boldsymbol{\lambda}_i^{(k)} = ((\lambda_i^h)^{(k)})_{h=1}^H$. The parameter ϵ is a pre-determined relatively small constant. We also denote S_c as the current set of remaining cloud users. Note that the individual strategies are not revealed among the users in any case, and only the aggregated request profile $\boldsymbol{\lambda}^{(k)}$, which is determined at the cloud provider adding the individual H-time slots ahead request profile, is communicated between the cloud provider and multiple cloud users.

4 Performance Evaluation of IPA

In this section, we provide some numerical results to validate our theoretical analyses and illustrate the performance of the IPA algorithm.

TABLE 1
System Parameters

System parameters	Value (Fixed)–[Varied range] (increment)
Aggregated task requests (Λ)	(500)–[100, 500] (50)
Number of cloud users (n)	(50)–[5, 50] (5)
Weight value (w_i)	[0, $1/n$]
Reservation value (v_i)	0
Other parameters ($\varepsilon, b, r, \delta$)	(0.01, 0, 50, 1.2)

In the following simulation results, we consider the scenario consisting of maximal 50 cloud users. Each time slot is set as one hour of a day and H is set as 24. As shown in Table 1, the aggregated request (Λ) is varied from 50 to 500 with increment 50. The number of cloud users (n) is varied from 5 to 50 with increment 5. Each cloud user i ($i \in \mathcal{N}$) chooses a weight value from 0 to $1/n$ to balance his/her time utility and net profit. For simplicity, the reservation value v_i for each user i ($i \in \mathcal{N}$) and billing parameter b are set to zero. Market benefit factor r is set to 50, deteriorating rate on time utility δ is equal to 1.2, and ε is set as 0.01. The total capacity of all servers μ is selected to satisfy constraint (24) and another billing parameter a is computed according to (17). In our simulation, the initial strategy configuration, i.e., before using IPA algorithm, is randomly generated from Q.

Fig. 2 presents the utility results for five different cloud users versus the number of iterations of the proposed IPA algorithm. Specifically, Fig. 2 presents the utility results of five randomly selected cloud users (users 1, 9, 23, 38, and 46) with a scenario consisting of 50 cloud users. We can observe that the utilities of all the users seem to increase and finally reach a relative stable state with the increase of iteration number. The reason behind lies in that the request strategies of all the users keep unchanged, i.e., reach a Nash equilibrium solution after several iterations. This trend also reflects the convergence process of our proposed IPA algorithm. It can be seen that the developed algorithm converges to a Nash equilibrium very quickly. Specifically, the utility of each user has already achieved a relatively stable state after about eight iterations, which verifies the validness of Theorem 3.6, as well as displays the high efficiency of the developed algorithm.

In Fig. 3, we compare the aggregated request profile of all cloud users with the situation before and after IPA algorithm. Specifically, Fig. 3 shows the aggregated requests in different time slots. The situation before IPA algorithm corresponds to a feasible strategy profile randomly generated in the initialization stage, while the situation after IPA algorithm corresponds to the result obtained by using our proposed IPA algorithm. Obviously, the proposed service reservation scheme encourages the cloud users to shift their task requests in peak time slots to non-peak time slots, resulting in a more balanced load shape and lower total load. We can also observe that the aggregated requests in different time slots are almost the same. To demonstrate this phenomenon, we further investigate the specific utilities of some users and their corresponding strategies in different time slots, which are presented in Figs. 4 and 5.

In Figs. 4 and 5, we plot the utility shape and the request profile of some cloud users for the developed IPA algorithm for a scenario of 10 users. Fig. 4 presents the utility shape under the developed algorithm over future 24 time slots. We randomly select six users (users 2, 3, 4, 7, 9, and 10). It can be seen that the utilities in different time slots of all users tend to decrease at different degrees. Specifically, the slot utilities of the users with higher weights have a clearly downward trend and tend to decrease sharply in later time slots (users 2, 3, 7, 9). On the other hand, the slot utilities of the users with lower weights decline slightly (users 4, 10). Fig. 5 exhibits the corresponding request strategies of the

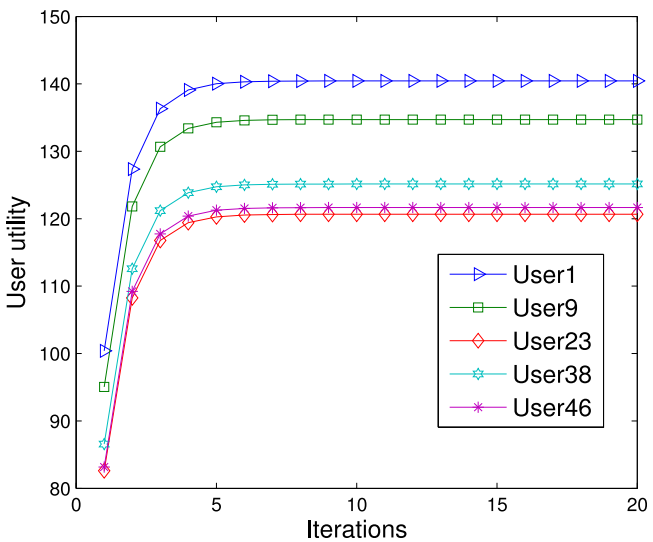


Fig. 2. Convergence process.

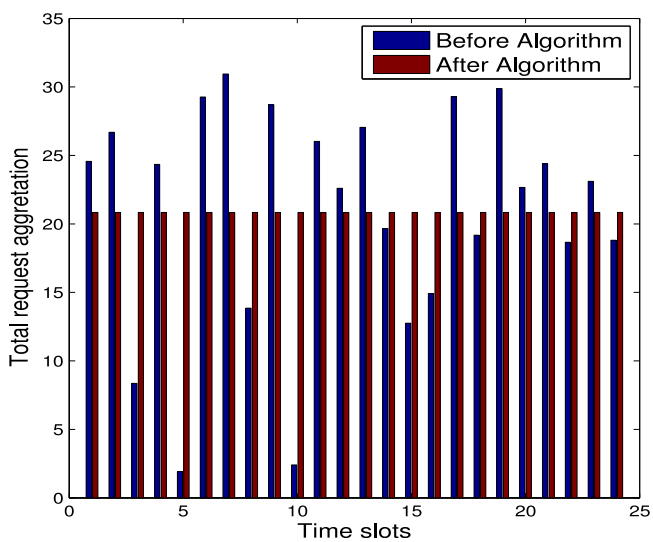


Fig. 3. Aggregation load.

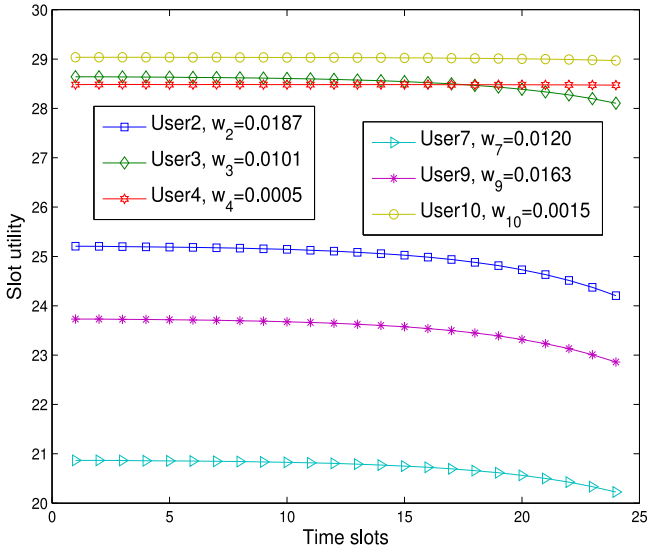


Fig. 4. Specific slot utility.

users shown in Fig. 4. We can observe that the slot utilities of the users with higher weights tend to decrease (users 2, 3, 7, 9) while those of the users with lower weights tend to increase (users 4, 10). Furthermore, the aggregated requests increase or decrease sharply in later time slots. The reason behind lies in the fact that in our proposed model, we take into the average response time into account and the deteriorating factor of the value grows exponentially, which also demonstrates the downward trends shown in Fig. 4. On the other hand, the weights are chosen randomly, there could be a balance between the increment and the decrement of the utilities. Hence, the aggregated requests in different time slots make little differences (Fig. 3).

Figs. 6 and 7 present the average utility versus the increase of request aggregation and the number of users, respectively. Fig. 6 illustrates the average utility results with the linear increment of request aggregation. We can observe that the average utility also linearly increases with the increase of request aggregation. No matter what the request aggregation is, the average utility obtained after our proposed IPA algorithm is better than that of the initial strategy profile. Moreover, the differences between the results before IPA algorithm and those after the algorithm are also increases. That is to say, our proposed IPA algorithm makes significant sense when the aggregated requests are somewhat large. Fig. 7 shows the impacts of number of users. It can be seen that both of the results after IPA algorithm and before algorithm are inversely proportional to the number of uses. The reason behind lies in that the variation of number of users makes little impact on the average utility value when the request aggregation is fixed. Moreover, similar to the results presented in Fig. 6, the average utility obtained after IPA algorithm is always better than that of the initial strategy profile.

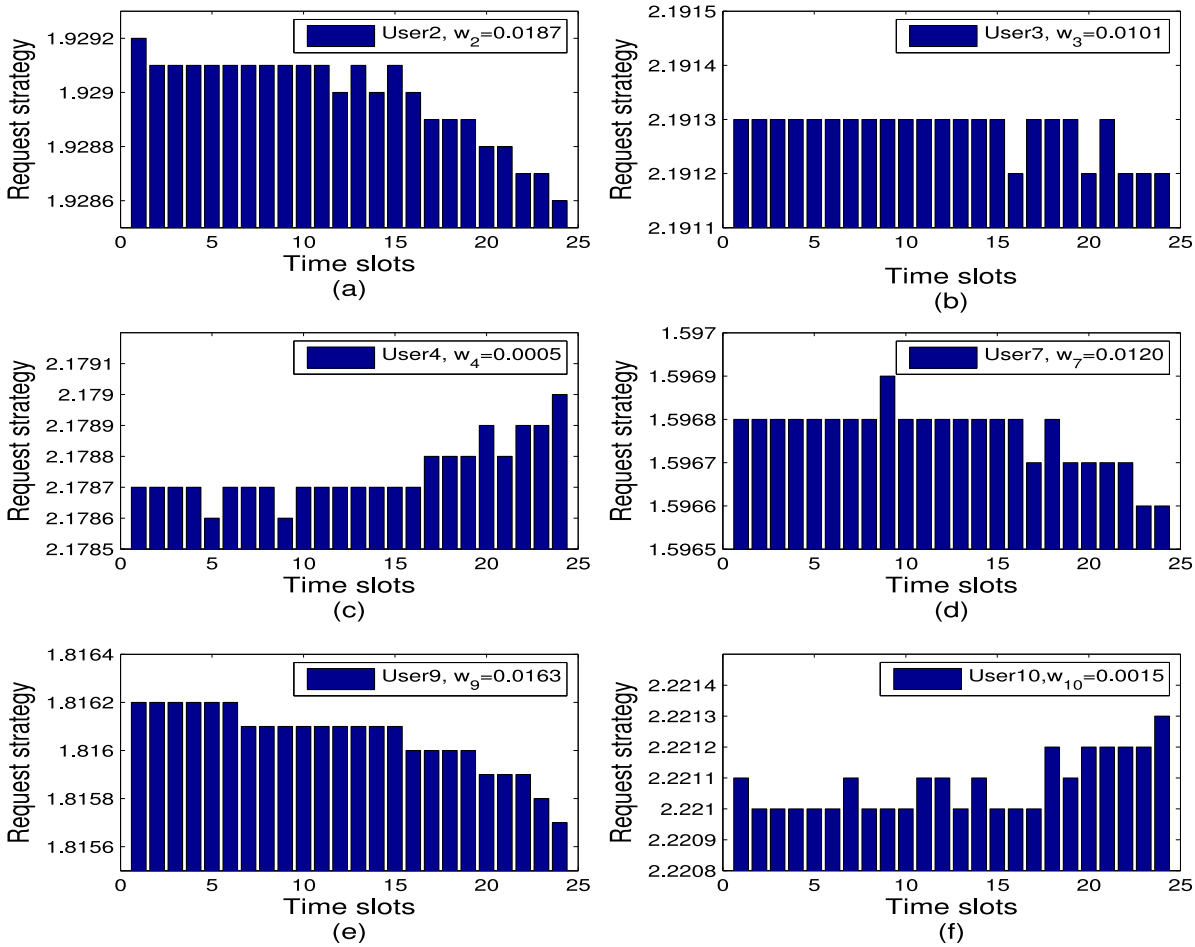


Fig. 5. Specific slot shifting.

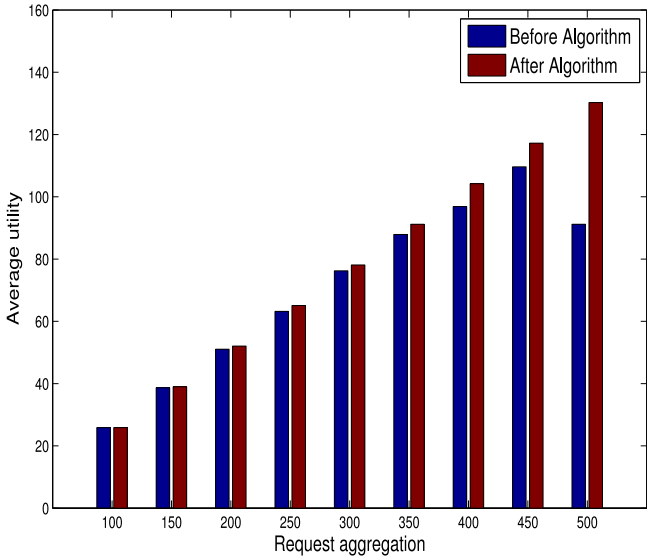


Fig. 6. Average utility versus aggregation.

5 Conclusions

With the popularization of cloud computing and its many advantages such as cost-effectiveness, flexibility, and scalability, more and more applications are moved from local to cloud. However, most cloud providers do not provide a mechanism in which the users can evaluate their costs and then decide whether to use the cloud service after obtaining a proper strategy. That is to say, the users cannot decide whether to use the service in advance and configure a most suitable strategies for their own. What's more, most cloud providers adopt the charge scheme that depends on the service time of the users' requests. However, we may note that the service time of a user is not only determined by the processing capacity of the cloud provider but also affected by the aggregated requests of other users. Therefore, it is not fair for the cloud users to certain extent when they use the service on peak-hours. To remedy these deficiencies, we focus on the strategy choices of multiple users to make cloud service reservation over several future time slots.

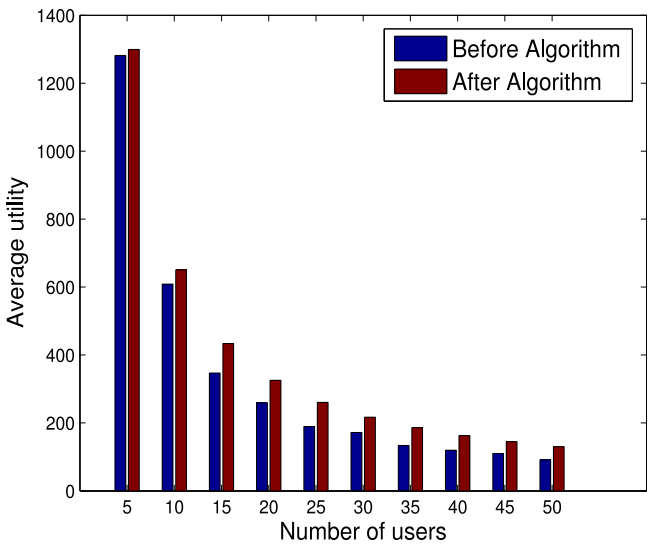


Fig. 7. Average utility versus number of users.

We first design a mechanism in which the cloud provider and its multiple users can establish negotiations, i.e., the users can decide whether to use the service and configure most suitable strategies for them. We consider the problem from a game theoretic perspective and formulate it into a non-cooperative game among the cloud users, in which each cloud user is informed with incomplete information of other users. For each user, we design a utility function which combines the profit with time efficiency and try to maximize its value. We solve the problem by employing variation inequality theory and prove that there exists a Nash equilibrium solution set for the formulated game. Then, we propose an iterative proximal algorithm , which is designed to compute a Nash equilibrium solution. The convergence of the IPA algorithm is also analyzed and we find that the proposed algorithm converges to a Nash equilibrium if several conditions are satisfied. Finally, we provide some numerical calculations to verify our theoretical analyses. The experimental results show that our proposed IPA algorithm converges to a stable state very quickly and improves the users' utilities to certain extent by configuring a proper request strategy.

As part of future directions, we will configure the multiple servers in cloud dynamically and study the relationship between the cloud provider and multiple users. Another direction is to study the cloud choice among multiple different cloud providers or determine a proper mixed choice strategy.

Acknowledgments

The authors would like to thank the three anonymous reviewers for their comments and suggestions to improve the manuscript. The research was partially funded by the Key Program of National Natural Science Foundation of China (Grant Nos. 61133005, 61432005), the National Natural Science Foundation of China (Grant Nos. 61370095, 61472124, 61202109, 61472126, 61402400), and the US National Science Foundation under grant CCF-1016966. Kenli Li is the corresponding author.

References

[1] Y. Feng, B. Li, and B. Li, "Price competition in an oligopoly market with multiple iaas cloud providers," *IEEE Trans. Comput.*, vol. 63, no. 1, pp. 59–73, Jan. 2014.

[2] R. Pal and P. Hui, "Economic models for cloud service markets: Pricing and capacity planning," *Theoretical Comput. Sci.*, vol. 496, pp. 113–124, 2013.

[3] P. D. Kaur and I. Chana, "A resource elasticity framework for QoS-aware execution of cloud applications," *Future Generation Comput. Syst.*, vol. 37, pp. 14–25, 2014.

[4] E. Simhon and D. Starobinski, "Game-theoretic analysis of advance reservation services," in *Proc. 48th Annu. Conf. Inf. Sci. Syst.*, 2014, pp. 1–6.

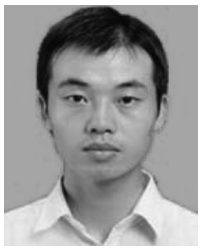
[5] R. Cohen, N. Fazlollahi, and D. Starobinski, "Path switching and grading algorithms for advance channel reservation architectures," *IEEE/ACM Trans. Netw.*, vol. 17, no. 5, pp. 1684–1695, Oct. 2009.

[6] S. Son and K. M. Sim, "A price-and-time-slot-negotiation mechanism for cloud service reservations," *IEEE Trans. Syst., Man, Cybern., Part B: Cybern.*, vol. 42, no. 3, pp. 713–728, Jun. 2012.

[7] J. Cao, K. Hwang, K. Li, and A. Y. Zomaya, "Optimal multiserver configuration for profit maximization in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1087–1096, Jun. 2013.

[8] A.-H. Mohsenian-Rad, V. W. Wong, J. Jatskevich, R. Schober, and A. Leon-Garcia, "Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid," *IEEE Trans. Smart Grid*, vol. 1, no. 3, pp. 320–331, Dec. 2010.

[9] H. Chen, Y. Li, R. H. Louie, and B. Vucetic, "Autonomous demand side management based on energy consumption scheduling and instantaneous load billing: An aggregative game approach," *IEEE Trans. Smart Grid*, vol. 5, no. 4, pp. 1744–1754, Jul. 2014.
[10] Z. Fadlullah, D. M. Quan, N. Kato, and I. Stojmenovic, "Gtes: An optimized game-theoretic demand-side management scheme for smart grid," *IEEE Syst. J.*, vol. 8, no. 2, pp. 588–597, Jun. 2014.
[11] H. Soliman and A. Leon-Garcia, "Game-theoretic demand-side management with storage devices for the future smart grid," *IEEE Trans. Smart Grid*, vol. 5, no. 3, pp. 1475–1485, May 2014.
[12] I. Atzeni, L. G. Ordóñez, G. Scutari, D. P. Palomar, and J. R. Fonollosa, "Noncooperative and cooperative optimization of distributed energy generation and storage in the demand-side of the smart grid," *IEEE Trans. Signal Process.*, vol. 61, no. 10, pp. 2454–2472, May 2013.
[13] G. Scutari, D. Palomar, F. Facchinei, and J.-S. Pang, "Convex optimization, game theory, and variational inequality theory," *IEEE Signal Process. Mag.*, vol. 27, no. 3, pp. 35–49, May 2010.
[14] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*. Cambridge, MA, USA: MIT Press, 1994.
[15] J.-P. Aubin, *Mathematical Methods of Game and Economic Theory*. New York, NY, USA: Dover, 2007.
[16] S. S. Aote and M. Kharat, "A game-theoretic model for dynamic load balancing in distributed systems," in *Proc. Int. Conf. Adv. Comput., Commun. Control*, 2009, pp. 235–238.
[17] N. Li and J. Marden, "Designing games for distributed optimization," in *Proc. 50th IEEE Conf. Decis. Control Eur. Control Conf.*, Dec 2011, pp. 2434–2440.
[18] E. Tsiropoulou, G. Katsinis, and S. Papavassiliou, "Distributed uplink power control in multiservice wireless networks via a game theoretic approach with convex pricing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 1, pp. 61–68, Jan. 2012.
[19] G. Scutari and J.-S. Pang, "Joint sensing and power allocation in nonconvex cognitive radio games: Nash equilibria and distributed algorithms," *IEEE Trans. Inf. Theory*, vol. 59, no. 7, pp. 4626–4661, Jul. 2013.
[20] N. Immorlica, L. E. Li, V. S. Mirrokni, and A. S. Schulz, "Coordination mechanisms for selfish scheduling," *Theoretical Comput. Sci.*, vol. 410, no. 17, pp. 1589–1598, 2009.
[21] S. Penmatsa and A. T. Chronopoulos, "Game-theoretic static load balancing for distributed systems," *J. Parallel Distrib. Comput.*, vol. 71, no. 4, pp. 537–555, 2011.
[22] G. Scutari, F. Facchinei, J.-S. Pang, and D. Palomar, "Real and complex monotone communication games," *IEEE Trans. Inf. Theory*, vol. 60, no. 7, pp. 4197–4231, Jul. 2014.
[23] N. Mandayam, G. Editor, S. Wicker, J. Walrand, T. Basar, J. Huang, and D. Palomar, "Game theory in communication systems [guest editorial]," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 7, pp. 1042–1046, Sep. 2008.
[24] E. Larsson, E. Jorswieck, J. Lindblom, and R. Mochaourab, "Game theory and the flat-fading gaussian interference channel," *IEEE Signal Process. Mag.*, vol. 26, no. 5, pp. 18–27, Sep. 2009.
[25] K. Li, C. Liu, and K. Li, "An approximation algorithm based on game theory for scheduling simple linear deteriorating jobs," *Theoretical Comput. Sci.*, vol. 543, no. 0, pp. 46–51, 2014.
[26] P. Samadi, H. Mohsenian-Rad, R. Schober, and V. Wong, "Advanced demand side management for the future smart grid using mechanism design," *IEEE Trans. Smart Grid*, vol. 3, no. 3, pp. 1170–1180, Sep. 2012.
[27] I. Atzeni, L. Ordonez, G. Scutari, D. Palomar, and J. Fonollosa, "Demand-side management via distributed energy generation and storage optimization," *IEEE Trans. Smart Grid*, vol. 4, no. 2, pp. 866–876, Jun. 2013.
[28] J. Cao, K. Li, and I. Stojmenovic, "Optimal power allocation and load distribution for multiple heterogeneous multicore server processors across clouds and data centers," *IEEE Trans. Comput.*, vol. 63, no. 1, pp. 45–58, Jan. 2014.
[29] S. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge university press, 2009.
[30] G. Scutari, D. Palomar, F. Facchinei, and J.-S. Pang, " Monotone games for cognitive radio systems," in *Proc. Distrib. Decision Making Control*, 2012, pp. 83–112.
[31] E. Altman, T. Basar, T. Jimenez, and N. Shimkin, "Competitive routing in networks with polynomial costs," *IEEE Trans. Autom. Control*, vol. 47, no. 1, pp. 92–96, Jan. 2002.



Chubo Liu is currently working towards the PhD degree at Hunan University, China. His research interests mainly include modeling and scheduling of distributed computing systems, approximation and randomized algorithms, game theory, and grid and cloud computing.

Kenli Li received the PhD degree in computer science from the Huazhong University of Science and Technology, China, in 2003. He was a visiting scholar at the University of Illinois at Urbana-Champaign from 2004 to 2005. He is currently a full professor of computer science and technology at Hunan University and deputy director of the National Supercomputing Center in Changsha. His major research areas include parallel computing, high-performance computing, and grid and cloud computing. He has published more than 100 research papers in international conferences and journals such as *IEEE Transactions on Computers*, *IEEE Transactions on Parallel and Distributed Systems*, *Journal of Parallel and Distributed Computing*, ICPP, CCGrid. He is an outstanding member of CCF. He is a member of the IEEE and serves on the editorial board of the *IEEE Transactions on Computers*.

Chengzhong Xu received the PhD degree from the University of Hong Kong in 1993. He is currently a professor of the Department of Electrical and Computer Engineering of Wayne State University. He also holds an adjunct appointment with the Shenzhen Institute of Advanced Technology of Chinese Academy of Science as the director of the Institute of Advanced Computing and Data Engineering. His research interests are in parallel and distributed systems and cloud computing. He has published more than 200 papers in journals and conferences. He was the Best Paper Nominee of 2013 IEEE High Performance Computer Architecture (HPCA), and the Best Paper Nominee of 2013 ACM High Performance Distributed Computing (HPDC). He serves on a number of journal editorial boards, including the *IEEE Transactions on Computers*, *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Cloud Computing*, *Journal of Parallel and Distributed Computing*, and *China Science Information Sciences*. He was a recipient of the Faculty Research Award, Career Development Chair Award, and the Presidents Award for Excellence in Teaching of WSU. He was also a recipient of the Outstanding Oversea Scholar award of NSFC. He is a senior member of the IEEE.

Keqin Li is a SUNY distinguished professor of computer science. His current research interests include parallel computing and high-performance computing, distributed computing, energy-efficient computing and communication, heterogeneous computing systems, cloud computing, big data computing, CPU-GPU hybrid and cooperative computing, multicore computing, storage and file systems, wireless communication networks, sensor networks, peer-to-peer file sharing systems, mobile computing, service computing, Internet of things, and cyber-physical systems. He has published more than 320 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He is currently or has served on the editorial boards of the *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *IEEE Transactions on Cloud Computing*, and the *Journal of Parallel and Distributed Computing*. He is a fellow of the IEEE.