This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2014.2322355, IEEE Transactions on Cloud Computing

A Novel Model for Competition and Cooperation Among Cloud Providers

Tram Truong-Huu, *Member, IEEE*, and Chen-Khong Tham, *Member, IEEE*

Abstract—Having received significant attention in the industry, the cloud market is nowadays fiercely competitive with many cloud providers. On one hand, cloud providers compete against each other for both existing and new cloud users. To keep existing users and attract newcomers, it is crucial for each provider to offer an optimal price policy which maximizes the final revenue and improves the competitive advantage. The competition among providers leads to the evolution of the market and dynamic resource prices over time. On the other hand, cloud providers may cooperate with each other to improve their final revenue. Based on a Service Level Agreement, a provider can outsource its users' resource requests to its partner to reduce the operation cost and thereby improve the final revenue. This leads to the problem of determining the cooperating parties in a cooperative environment. This paper tackles these two issues of the current cloud market. First, we solve the problem of competition among providers and propose a dynamic price policy. We employ a discrete choice model to describe the user's choice behavior based on his obtained benefit value. The choice model is used to derive the probability of a user choosing to be served by a certain provider. The competition among providers is formulated as a non-cooperative stochastic game where the players are providers who act by proposing the price policy simultaneously. The game is modelled as a Markov Decision Process whose solution is a Markov Perfect Equilibrium. Then, we address the cooperation among providers by presenting a novel algorithm for determining a cooperation strategy that tells providers whether to satisfy users' resource requests locally or outsource them to a certain provider. The algorithm yields the optimal cooperation structure from which no provider unilaterally deviates to gain more revenue. Numerical simulations are carried out to evaluate the performance of the proposed models.

Index Terms—Cloud computing, dynamic pricing, cooperation, Markov Decision Process, Markov Perfect Equilibrium, game theory

1 Introduction

During the past few years, cloud computing has received significant investments in the industry. Many cloud providers are participating in the market, forming a competitive environment which is referred to as *multi-user and multi-provider cloud market*. Hereafter, we will use the terms "providers" and "users" to refer to the cloud actors. Since the amount of resources in a user's request is much smaller than the capacity of a provider, the user's request can be satisfied by any provider. A rational user will choose the provider whose resources best satisfy his computational needs, and the resource usage cost does not exceed his budgetary constraint. The user's satisfaction can be evaluated through a *utility* measure which depends not only on the resource properties but also on the user's preference to choose certain providers, i.e., two providers with the same resource capacities and usage price may be considered different for a user due to the user's choice behavior and loyalty. Furthermore, the task of optimally pricing cloud resources to attract users and improve revenue is very challenging [1]. They need to take into account a wide range of factors including the preferences of users, resource capacities and potential competition from other providers. A provider naturally wishes to set a higher price to get a higher revenue; however, in doing so, it also bears the risk of discouraging demand in the future. On the other hand, they also look for the means to cooperate with other providers to reduce the operation cost and therefore improve their final revenue. In this paper, we study both problems of the current cloud market: competition and cooperation among providers.

The competition among providers leads to the dynamics of cloud resource pricing. Modelling this competition involves the description of the user's choice behaviour and the formulation of the dynamic pricing strategies of providers to adapt to the market state. To describe the user's choice behavior, we employ a widely used discrete choice model, the multi-nomial logit model [2], which is defined as a utility function whose value is obtained by using resources requested from providers. From the utility function, we derive the probability of a user choosing to be served by a certain provider. The choice probability is then used by providers to determine the optimal price policy. The fundamental question is how to determine the optimal price policy. When a provider joins the market, it implicitly participates in a competitive game established by existing providers. Thus, optimally playing this game helps providers to not only survive in the market, but also improve their revenues. To give providers a means to solve this problem, we formulate the competition as a non-cooperative stochastic game [3]. The game is modelled as a Markov Decision Process (MDP) [4] whose state space is finite and computed by the distribution of users among providers. At each step of the game, providers simultaneously propose new price policies with respect to the current policies of other competitors such that their revenues are maximized. Based on those price policies, users will decide which provider they will select to request resources. This also determines whether the market will move to a new state or not. The solution of the game is a Markov Perfect Equilibrium (MPE) such that none of providers can improve their revenues by unilaterally deviating from the equilibrium in the long run.

• *The authors are with the Department of Electrical & Computer Engineering, National University of Singapore. Postal address: Block E4, Level 6, Room 12, Engineering Drive 3, Singapore 117583.*
E-mail: {eletht,eletck}@nus.edu.sg

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2014.2322355, IEEE Transactions on Cloud Computing

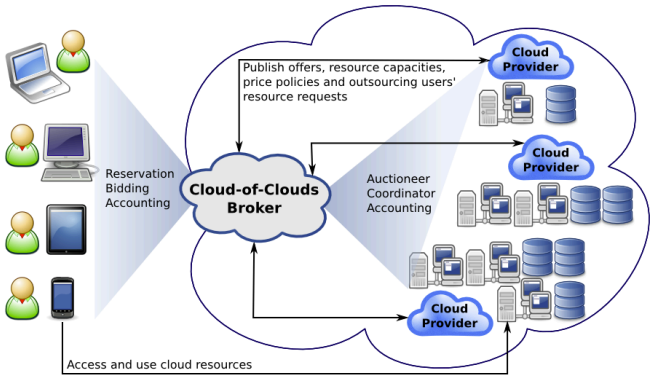


Fig. 1: Overall architecture of a Cloud-of-Clouds system.

On the second problem, the cooperation based on a financial option allows providers to enhance revenue and acquire the needed resources at any given time [5]. The revenue depends on the total operation cost which includes a cost to satisfy users' resource requests (i.e., cost for active resources) and another cost for maintaining data center services (i.e., cost for idle resources). We address the problem of cooperation among providers by first employing the learning curve [6] to model the operation cost of providers and then introducing a novel algorithm that determines the cooperation structure. The cooperation decision algorithm uses the operation cost computed based on the learning curve model and price policies obtained from the competition part as parameters to calculate the final revenue when outsourcing or locally satisfying users' resource requests. The cooperation among providers makes the cloud market become a united cloud environment, called *Cloud-of-Clouds* environment as illustrated in Fig. 1. In this architecture, the Cloud-of-Clouds Broker is responsible for coordinating the cooperation among providers, receiving users' resource requests and also doing accounting management.

The rest of this paper is organized as follows. After discussing related work in Section 2, we introduce in Section 3 our assumption and the relevant models for users' resource requests, providers' operation costs and revenues. We describe in Section 4 the game formulation for competition among providers. In Section 5, we present the method for solving the stochastic game which results in the MPE. Section 6 presents the model and algorithm for cooperation among providers. Section 7 presents the numerical simulations and analysis of results that assess the validity of the proposed model. Finally, we conclude the paper in Section 8.

2 Related Work

2.1 Dynamic pricing and competition

Dynamic pricing in the cloud has gained considerable attention from both industry and academia. Amazon EC2 has introduced a "spot pricing" feature for its resource instances where the spot price is dynamically adjusted to reflect the equilibrium prices that arises from resource demand and supply. Analogously, a statistical model of spot instance prices in public cloud environments has been presented in [7], which fits Amazons spot instances prices well with a good degree of accuracy. To capture the realistic value of the cloud resources, the authors of [8] employ a financial option theory and treat the cloud resources as real assets. The cloud resources are then priced by solving the finance model. Also based on financial option theory, in [5], a cloud resource pricing model has been proposed to address the resource trading among members of a federated cloud environment. The model allows providers to avoid the resource over-provisioning and under-provisioning problems. But there is an underlying assumption is that there are always providers willing to sell call options.

In [9] and [10], the authors presented their research results on dynamic pricing for cloud resources. While [9] studied the case of a single provider operating an IaaS cloud with a fixed capacity, [10] focussed on the case of an oligopoly market with multiple providers. However, both [9] and [10] make the assumption that the user's resource request is a concave function with respect to resource prices. The amount of resources requested will decrease when prices increase. This assumption is not practical when users have a processing deadline or architectural requirements for their execution platform. Users therefore have to request the required amount of resources no matter what the prices are. In [11] and [12], the authors presented auction-based mechanisms to determine optimal resource prices, taking into account the user's budgetary and deadline constraints. However, they considered the pricing model of only one provider. In contrast, we consider the realistic case of the current cloud market with multiple providers. In addition, users may have their preferences in choosing to be served by particular providers.

2.2 Game theory in utility computing

Game theory has been widely applied in economic studies for dynamic pricing competition [13], [14], [15]. In utility computing, game theory has been applied to study different issues: scheduling and resource allocation [16], [17], dynamic pricing [18] and revenue optimization [19]. In [16], a game-theoretic resource allocation algorithm has been proposed to minimize the energy consumption while guaranteeing the processing deadline and architectural requirement. In [17], a user-oriented job allocation scheme has been formulated as a non-cooperative game to minimize the expected cost of executing users' tasks. The solution is a Nash equilibrium which is obtained using a distributed algorithm. However, none of these works considered the user's choice behavior, although some of them assume that resources are owned by different resource owners.

2.3 Cooperation among providers

Cooperation among providers in cloud computing has been extensively studied with two research approaches: cloud federation and coalitional formation based on coalitional game theory.

The idea of federating systems was originally presented for grid computing. For instance, in [20] and [21], the authors used the federation approach to get more computing resources to execute large scale applications

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2014.2322355, IEEE Transactions on Cloud Computing

in a distributed grid environment. The application of the federation approach in the cloud was initially proposed within the RESERVOIR project [22]. Nevertheless, the aforementioned works focused only on aggregating as much resources as possible to satisfy users' resource requests. They did not consider the economic issue which is one of the intrinsic characteristics of cloud computing. In [23], the authors presented an economic model along with a federated scheduler which allow a provider, operating in a federated cloud, to increase the final revenue by saving capital and operation costs.

Based on coalitional game theory, [24] studied the problem of motivating self-interested providers to join a determined horizontal dynamic cloud federation platform and the problem of deciding the amount of resources to be allocated to the federation. The authors of [19] used the coalitional game approach to form cloud federations and share the obtained revenue among coalition members fairly. However, this work did not consider the operation cost of providers which is an important factor in the economic model. In this paper, we consider the realistic case of the current cloud market where providers may have different operation costs. Cooperation among providers may reduce the operation cost and therefore improve the final revenue.

3 System formulation

3.1 Assumptions

In this paper, we consider providers which offer Infrastructure as a Service (IaaS) from which users can request a number of cloud resource instances and deploy their own platform for executing their applications. Table 1 presents all the notations used throughout this paper.

There are a total of N providers on the cloud market. These providers offer a number of resource types denoted by M, e.g., Amazon EC2 offers 4 types of resources called VM instances: small, medium, large and extra large[1]. Depending on the capacity of each resource type, providers define a per unit price to charge users for resource usage. We denote vector $\mathbf{p}_i = (p_{i1}, p_{i2}, \ldots, p_{iM})$ as the price vector of provider i where p_{ij} with $j \in [1, \ldots, M]$ is the per unit price of resource type j (US$/hour). For each resource type of each provider, we additionally define the per unit benefit λ_{ij}, i.e., per unit benefit of resource type j offered by provider i, to reflect the relative capacity of the resource in satisfying the user's computational needs. A higher benefit resource may be more expensive than a lower benefit one.

3.2 Users' resource requests

We assume that, in total, K users distribute their resource requests among N providers. User k places a request for a bundle of resources rather than for individual items which is the usual case in the cloud environment. Therefore, the resource request of user k is represented by a vector $\mathbf{r}_k = (r_{k1}, r_{k2}, \ldots, r_{kM})$ where r_{kj} with $j \in [1, \ldots, M]$ is the required number of instances of resource type j. For user k, we additionally define b_k as his budgetary constraint. Given the price policies of all providers, the cost that user k needs to pay for using resources offerred by provider i is defined as $c_{ki} = \sum_{j=1}^{M} r_{kj} p_{ij}$, subject to $c_{ki} \leqslant b_k, \forall i \in [1, \ldots, N]$, and $\forall k \in [1, \ldots, K]$.

1. Amazon EC2 Instances: http://aws.amazon.com/ec2/pricing

TABLE 1: Mathematical notations

Notation	Description
N	Number of providers on the cloud market
M	Number of resource types of each provider
K	Number of users on the cloud market
i	Index of cloud providers
j	Index of resource types
k	Index of cloud users
b_k	Budgetary constraint of user k
φ_i	Learning factor of provider i
ψ_{ij}	Maximum number of instances of resource type j offered by provider i
λ_{ij}	Per unit benefit of resource type j offered by provider i
p_{ij}	Per unit price of resource type j charged by provider i
c_{ij}^{o}	Operation cost of the first active instance of resource type j owned by provider i
$c_{ij}^{\bar{o}}$	Operation cost of the first idle instance of resource type j owned by provider i
Ω	State space of stochastic game/the cloud market
$\boldsymbol{\omega}$	State of the market ($\boldsymbol{\omega} \in \Omega$)
$\boldsymbol{\beta}_i$	Individual state of provider i
γ	Discount factor of money
η_{ki}	Preference of user k to provider i
r_{kj}	Number of instances of resource type j requested by user k
c_{ki}	Cost that user k needs to pay when requesting resources from provider i
U_{ki}	Utility of user k being served by provider i
P_{ki}	Probability of user k choosing provider i
C_i^{o}	Operation cost of provider i for active resources
$C_i^{\bar{o}}$	Operation cost of provider i for idle resources
C_i	Total operation cost of provider i
$C_{ii'}^{\text{outsource}}$	Outsourcing cost of provider i when outsourcing to provider i'
R_i^{local}	Revenue of provider i when satisfying all users' resource requests locally
$R_{ii'}^{\text{outsource}}$	Revenue of provider i when outsourcing all users' resource requests to provider i'
R_i^{hosting}	Revenue of provider i when hosting users' resource requests from other providers
$\hat{V}_i$	Discounted sum of future revenue of provider i

3.3 Market state

The state of the market is given by $\boldsymbol{\omega} = (\omega_1, \omega_2, \ldots, \omega_K)$ where $\omega_k \in \{1, \ldots, N\}$ is the identification/index of the provider to which user k sends his resource request, i.e., user k chooses to be served by provider ω_k. Let Ω denote the state space which is the set of possible states of the market. We define a function $\delta : \Omega \times \{1, \ldots, N\} \rightarrow \{0, 1\}^K$ to convert the market state to the individual state of a specific provider. Let $\boldsymbol{\beta}_i$ be the individual state of provider i corresponding to the market state $\boldsymbol{\omega}$, we have

$$\boldsymbol{\beta}_i = \delta(\boldsymbol{\omega}, i) = (\beta_{i1}, \beta_{i2}, \ldots, \beta_{iK}), \beta_{ik} = \begin{cases} 1 & \text{if } \omega_k = i, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2014.2322355, IEEE Transactions on Cloud Computing

3.4 Providers' operation costs

In the current cloud market, providers may have different resource capacities and operation management processes. This leads to different operation costs among providers. The learning curve model [6] assumes that as the number of production units are doubled, the marginal cost of production decreases by a fixed factor. One minus this factor is called learning factor. The provider with a higher learning factor has to pay a higher operation cost than that of the provider with a lower learning factor when running the same number of production units. Mathematically, given the individual state $\boldsymbol{\beta}_i$ of provider i, we denote vector $\mathbf{a}_i = (a_{i1}, a_{i2}, \ldots, a_{iM})$ as the list of numbers of active instances of each resource type operated by provider i, i.e., these instances are in production for satisfying users' resource requests. Hence, a_{ij} is computed as $\sum_{k=1}^{K} \beta_{ik} r_{kj}$. The total cost of operating active instances is then

$$C_i^o(\boldsymbol{\beta}_i) = \sum_{j=1}^{M} \frac{c_{ij}^o a_{ij}^{1+\log_2 \varphi_i}}{1+\log_2 \varphi_i} \tag{2}$$

where c_{ij}^o is the operation cost of the first active instance of resource type j operated by provider i, and φ_i is the learning factor of provider i. It is to be noted that the learning factor is the same for all resource types operated by the same provider.

In addition to active resource instances, providers also need to maintain the basic services for idle resource instances. Let vector $\boldsymbol{\psi}_i = (\psi_{i1}, \psi_{i2}, \ldots, \psi_{iM})$ denote the resource capacities of provider i, where ψ_{ij} is maximum number of instances of resource type j offered by provider i, the number of idle instances of each resource type is represented by vector $(\psi_{i1} - a_{i1}, \psi_{i2} - a_{i2}, \ldots, \psi_{iM} - a_{iM})$. The total operation cost for these idle resource instances is therefore

$$C_i^{\bar{o}}(\boldsymbol{\beta}_i) = \sum_{j=1}^{M} \frac{c_{ij}^{\bar{o}} (\psi_{ij} - a_{ij})^{1+\log_2 \varphi_i}}{1+\log_2 \varphi_i} \tag{3}$$

where $c_{ij}^{\bar{o}}$ is the operation cost of the first idle instance of resource type j operated by provider i.

The total operation cost of provider i is therefore

$$\begin{aligned} C_i(\boldsymbol{\beta}_i) &= C_i^o(\boldsymbol{\beta}_i) + C_i^{\bar{o}}(\boldsymbol{\beta}_i) \\ &= \sum_{j=1}^{M} \frac{c_{ij}^o a_{ij}^{1+\log_2 \varphi_i}}{1+\log_2 \varphi_i} + \sum_{j=1}^{M} \frac{c_{ij}^{\bar{o}} (\psi_{ij} - a_{ij})^{1+\log_2 \varphi_i}}{1+\log_2 \varphi_i}. \end{aligned} \tag{4}$$

3.5 Providers' final revenues

Given the price policy $\mathbf{p}_i = (p_{i1}, p_{i2}, \ldots, p_{iM})$ and the individual state $\boldsymbol{\beta}_i$ of provider i, the total gross revenue of provider i is defined as follows:

$$\begin{aligned} R_i^{\text{gross}}(\boldsymbol{\beta}_i, \mathbf{p}_i) &= R_i^{\text{gross}}(\delta(\boldsymbol{\omega}, i), \mathbf{p}_i) \\ &= \sum_{k=1}^{K} \beta_{ik} c_{ki} = \sum_{k=1}^{K} \left(\beta_{ik} \sum_{j=1}^{M} r_{kj} p_{ij} \right). \end{aligned} \tag{5}$$

Combining the gross revenue with the total operation cost, we obtain the final revenue of provider i as follows:

$$\begin{aligned} R_i(\boldsymbol{\beta}_i, \mathbf{p}_i) = R_i^{\text{gross}}(\boldsymbol{\beta}_i, \mathbf{p}_i) - C_i(\boldsymbol{\beta}_i) = \sum_{k=1}^{K} \beta_{ik} c_{ki} \\ - \sum_{j=1}^{M} \frac{c_{ij}^o a_{ij}^{1+\log_2 \varphi_i}}{1+\log_2 \varphi_i} - \sum_{j=1}^{M} \frac{c_{ij}^{\bar{o}} (\psi_{ij} - a_{ij})^{1+\log_2 \varphi_i}}{1+\log_2 \varphi_i}. \end{aligned} \tag{6}$$

4 Dynamic Resource Pricing and Competition among Providers

4.1 Stochastic game formulation for competition

We formulate the competition among providers as a stochastic game [25]. The game is then modelled as a Markov Decision Process (MDP) which is defined by a four-tuple $(\Omega, \mathbb{A}, R, P)$. Ω is the state space of the market. At any time, the state of the market is given by $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_K)$ where $\omega_k \in \{1, \ldots, N\}$ is the identification or index of the provider by which user k chooses to be served. The size of state space Ω is $|\Omega| = N^K$. $\mathbb{A}$ is the action space. Providers are the players of the game who act by proposing the price policy simultaneously. R is the reward function which is the total final revenue of each provider and P is the transition probability. For instance, $P(\boldsymbol{\omega'}|\boldsymbol{\omega}, \mathbf{p}_1, \ldots, \mathbf{p}_N)$ is the probability of transitioning to state $\boldsymbol{\omega'}$ given that the market is at state $\boldsymbol{\omega}$ and providers propose price policies $(\mathbf{p}_1, \ldots, \mathbf{p}_N)$. More precisely, at state $\boldsymbol{\omega} \in \Omega$, provider i proposes a price policy $\mathbf{p}_i = (p_{i1}, \ldots, p_{iM})$ to attract users sending resource requests to it in the next period. The objective of each provider is therefore to find an optimal price policy $\hat{\mathbf{p}}_i = (\hat{p}_{i1}, \ldots, \hat{p}_{iM})$ at all states $\boldsymbol{\omega}$ that maximizes the discounted sum of future revenue:

$$\hat{V}_i(\boldsymbol{\omega}, \hat{\mathbf{p}}_1, \ldots, \hat{\mathbf{p}}_N) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}\left[R_i^t | \hat{\mathbf{p}}_1, \ldots, \hat{\mathbf{p}}_N, \boldsymbol{\omega}_0 = \boldsymbol{\omega} \right], \tag{7}$$

where R_i^t is the revenue received at period t as defined in (6). The game proceeds as follows:

Step 1. Assuming that the game is currently at the end of period $t-1$. The state of the market is $\boldsymbol{\omega}$ in which the individual state of provider i is $\boldsymbol{\beta}_i$ and state of all providers other than i is $\boldsymbol{\beta}_{-i}$. $\boldsymbol{\beta}_{-i}$ is observable to provider i.

Step 2. Given this information, all providers in the market simultaneously propose a new price policy. They can increase or decrease the price to increase the chance that users will send resource requests to them in the next period t.

Step 3. Depending on the providers' price policies, users then decide which provider they will be served by. The market moves to state $\boldsymbol{\omega}'$ at period t.

Step 4. In the period t, provider i will receive a total revenue of R_i^t as defined in (6). The game then repeats again from Step 1 to Step 4.

Solving this game results in the optimal price policies for all providers. The optimal solution is an MPE which assures the stability of the solution: no provider can achieve a better revenue by deviating unilaterally from his or her Markov strategy as long as the other providers continue to use their Markov strategies [4]. It is well-known that the use of Markov strategies in a non-cooperative game may prevent the attainment of efficient

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2014.2322355, IEEE Transactions on Cloud Computing

outcomes which are the Pareto Optima [26]. However, since providers pursue their own self-interest, they do not have incentives to play according to Pareto Optima in a non-cooperative game. For more discussion on MPE and Pareto Optima, we refer the reader to [27]. Yet, when a new provider joins the market, the game changes in the number of players and providers have to change their price policies to take into account the potential competitive advantage of the new provider. Thus, the game needs to be re-solved once the market changes.

4.2 User utility function and choice probability

In the current cloud market, users can easily compare resource prices of all providers and calculate the obtained utility before deciding to be served by a certain provider. Understanding the user's choice behavior can help providers to strengthen their competitive advantage. Discrete choice models have been widely used to describe the user's choice behavior and derive the choice probability from the principle of utility-maximization [15]. In this paper, we employ the multinomial logit (MNL) model based on its broad adoption [2]. For more discussion on the MNL model used in marketing science literature, we refer the reader to [28].

Naturally, with the same amount of resources requested from a provider, the user's utility is inversely proportional to the cost he pays to the provider. We define the utility function U_{ki} of user k to provider i as follows:

$$U_{ki} = \alpha_{ki} - c_{ki} + \eta_{ki} = v_{ki} + \eta_{ki}, \tag{8}$$

where $\alpha_{ki} = \sum_{j=1}^{M} r_{kj}\lambda_{ij}$ is the total benefit received by using resources of provider i, $c_{ki} = \sum_{j=1}^{M} r_{kj}p_{ij}$ is the cost that user k pays to provider i, and η_{ki} is the preference of user k to provider i. For ease of presentation, we define an additional variable $v_{ki} = \alpha_{ki} - c_{ki}$.

As shown in (8), the user's utility is decomposed into two parts: the first part, labeled as v_{ki}, that is observable by providers, and the second part denoted by η_{ki} which is the user's preference to a provider to whom it is unknown. This unknown part is therefore treated as random variable. We assume that random variable η_{ki} is an i.i.d extreme value, i.e., the distribution is also called the Gumbel distribution and type I extreme value [29]. The probability density of η_{ki} is $f(\eta_{ki}) = e^{-\eta_{ki}}e^{-e^{-\eta_{ki}}}$, and the cumulative distribution is $F(\eta_{ki}) = e^{-e^{-\eta_{ki}}}$.

We can now derive the logit choice probability, denoted as P_{ki}, that user k chooses to be served by provider i as follows:

$$\begin{aligned} P_{ki} &= \text{Prob}(U_{ki} > U_{ki'}, \forall i' \neq i) \\ &= \text{Prob}(v_{ki} + \eta_{ki} > v_{ki'} + \eta_{ki'}, \forall i' \neq i) \\ &= \text{Prob}(\eta_{ki'} < \eta_{ki} + v_{ki} - v_{ki'}, \forall i' \neq i). \end{aligned} \tag{9}$$

Based on the density function and the cumulative distribution, some algebraic manipulation of Eq. (9) results in a succinct, closed-form expression:

$$P_{ki} = \frac{e^{v_{ki}}}{\sum_{i'} e^{v_{ki'}}}, \tag{10}$$

which is the probability of user k choosing provider i to request resources. For the detailed algebra that leads to (10), we refer the reader to [2] (chapter 3).

4.3 State transition probability

Given the choice probability of user k choosing to be served by provider i, $P_{ki} = e^{v_{ki}} / \sum_{i'} e^{v_{ki'}}$, the probability of user k not choosing provider i is then $1 - P_{ki}$. Given the current state of the market, $\boldsymbol{\omega}$, we now compute the probability of a transition from state $\boldsymbol{\omega}$ to $\boldsymbol{\omega}'$. According to Eq. (1), the corresponding current individual state of provider i is $\boldsymbol{\beta}_i = \delta(\boldsymbol{\omega}, i)$, and the future individual state is $\boldsymbol{\beta}'_i = \delta(\boldsymbol{\omega}', i)$. The weighted probability of a transition from the current state $\boldsymbol{\beta}_i$ to the future state $\boldsymbol{\beta}'_i$ of provider i is defined as follows:

$$q_i(\boldsymbol{\beta}'_i|\boldsymbol{\beta}_i, \mathbf{p}_i) = \sigma_i(\boldsymbol{\beta}_i|\boldsymbol{\beta}'_i) \prod_{k=1}^{K} \left(\beta'_{ik}P_{ki} + (1-\beta'_{ik})(1-P_{ki})\right), \tag{11}$$

where $\sigma_i(\boldsymbol{\beta}_i|\boldsymbol{\beta}'_i) = \exp\left(-\sum_{k=1}^{K}(\beta_{ik} - \beta'_{ik})^2\right)$ is a weighting factor that is small when the sum of squared distance between state $\boldsymbol{\beta}_i$ and $\boldsymbol{\beta}'_i$ is large. The weighted probability of the market to transit from state $\boldsymbol{\omega}$ to state $\boldsymbol{\omega}'$ is

$$q(\boldsymbol{\omega}'|\boldsymbol{\omega}, \mathbf{p}_1, \ldots, \mathbf{p}_N) = \prod_{i=1}^{N} q_i(\boldsymbol{\beta}'_i|\boldsymbol{\beta}_i, \mathbf{p}_i), \tag{12}$$

where $\boldsymbol{\beta}_i = \delta(\boldsymbol{\omega}, i)$ and $\boldsymbol{\beta}'_i = \delta(\boldsymbol{\omega}', i)$. We now have the normalized state transition probability:

$$P(\boldsymbol{\omega}'|\boldsymbol{\omega}, \mathbf{p}_1, \ldots, \mathbf{p}_N) = \frac{q(\boldsymbol{\omega}'|\boldsymbol{\omega}, \mathbf{p}_1, \ldots, \mathbf{p}_N)}{\sum_{\boldsymbol{\theta}\in\Omega} q(\boldsymbol{\theta}|\boldsymbol{\omega}, \mathbf{p}_1, \ldots, \mathbf{p}_N)}. \tag{13}$$

5 Solving the Stochastic Game

We now present the algorithm for solving the game. The algorithm's output is the optimal price policies of all providers. We also analyze a limitation of the algorithm and introduce a method to overcome this limitation.

5.1 Algorithm for finding optimal price policies $\hat{\mathbf{p}}$

Since the game modelled as an MDP satisfies Bellman's Principle of Optimality which indicates that "an optimal policy has the property that whatever the initial state and initial action are, the remaining actions must constitute an optimal policy with regard to the state resulting from the first action", it can be solved via dynamic programming. The algorithm is based on the Bellman equation which is derived by summarizing the rewards of the current period and that of all future periods associated with a discount factor [30]. At the end of period $t-1$, when the market is at state $\boldsymbol{\omega}$, the corresponding Bellman equation is

$$\hat{V}_i(\boldsymbol{\omega}) = \max_{\mathbf{p}_i \in \mathbb{R}_+^M} \left\{ R_i(\delta(\boldsymbol{\omega}, i), \mathbf{p}_i) + \gamma \mathbb{E}_{\boldsymbol{\omega}'} \left[\hat{V}_i(\boldsymbol{\omega}') | \boldsymbol{\omega}, \mathbf{p}_i, \mathbf{p}_{-i} \right] \right\}, \tag{14}$$

where $\mathbf{p}_i$ is the price policy of provider i at state $\boldsymbol{\omega}$, $\mathbf{p}_{-i}$ is the price policies of providers other than i, $R_i(\delta(\boldsymbol{\omega}, i), \mathbf{p}_i)$ is the final revenue of provider i received at state $\boldsymbol{\omega}$ with price policy $\mathbf{p}_i$ as defined in (6), and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2014.2322355, IEEE Transactions on Cloud Computing

$\gamma\mathbb{E}_{\boldsymbol{\omega'}}\left[\hat{V}_i(\boldsymbol{\omega'})|\boldsymbol{\omega}, \mathbf{p}_i, \mathbf{p}_{-i}\right]$ is the discounted expected future revenue of provider i where the expectation is taken over the successor state $\boldsymbol{\omega'}$. Solving (14) results in the maximal revenue and the optimal price policy of provider i at state $\boldsymbol{\omega}$ as follows

$$\hat{\mathbf{p}}_i(\boldsymbol{\omega}) = \underset{\mathbf{p}_i \in \mathbb{R}_+^M}{\arg\max} \left\{ R_i(\delta(\boldsymbol{\omega}, i), \mathbf{p}_i) + \gamma\mathbb{E}_{\boldsymbol{\omega'}}\left[\hat{V}_i(\boldsymbol{\omega'})|\boldsymbol{\omega}, \mathbf{p}_i, \mathbf{p}_{-i}\right] \right\}. \tag{15}$$

By making explicit the expectation operator in (14) and (15), we can derive a new formula for the expected revenue in (16) and the optimal price policy in (17).

$$\hat{V}_i(\boldsymbol{\omega}) = \max_{\mathbf{p}_i \in \mathbb{R}_+^M} \left\{ R_i(\delta(\boldsymbol{\omega}, i), \mathbf{p}_i) + \gamma \sum_{\boldsymbol{\omega'} \in \Omega} P(\boldsymbol{\omega'}|\boldsymbol{\omega}, \mathbf{p}_i, \mathbf{p}_{-i}) \hat{V}_i(\boldsymbol{\omega'}) \right\} \tag{16}$$

$$\hat{\mathbf{p}}_i(\boldsymbol{\omega}) = \underset{\mathbf{p}_i \in \mathbb{R}_+^M}{\arg\max} \left\{ R_i(\delta(\boldsymbol{\omega}, i), \mathbf{p}_i) + \gamma \sum_{\boldsymbol{\omega'} \in \Omega} P(\boldsymbol{\omega'}|\boldsymbol{\omega}, \mathbf{p}_i, \mathbf{p}_{-i}) \hat{V}_i(\boldsymbol{\omega'}) \right\} \tag{17}$$

$$\hat{V}_i^t(\boldsymbol{\omega}) = \max_{\mathbf{p}_i \in \mathbb{R}_+^M} \left\{ R_i^t(\delta(\boldsymbol{\omega}, i), \mathbf{p}_i^t) + \gamma \sum_{\boldsymbol{\omega'} \in \Omega} P(\boldsymbol{\omega'}|\boldsymbol{\omega}, \mathbf{p}_i^t, \mathbf{p}_{-i}^{t-1}) \hat{V}_i^{t-1}(\boldsymbol{\omega'}) \right\} \tag{18}$$

$$\hat{\mathbf{p}}_i^t(\boldsymbol{\omega}) = \underset{\mathbf{p}_i \in \mathbb{R}_+^M}{\arg\max} \left\{ R_i^t(\delta(\boldsymbol{\omega}, i), \mathbf{p}_i^t) + \gamma \sum_{\boldsymbol{\omega'} \in \Omega} P(\boldsymbol{\omega'}|\boldsymbol{\omega}, \mathbf{p}_i^t, \mathbf{p}_{-i}^{t-1}) \hat{V}_i^{t-1}(\boldsymbol{\omega'}) \right\} \tag{19}$$

Each provider has its own version of Eqs. (16) and (17). This implies that for each provider and for each initial state, maximization is performed with respect to a non-linear equation with M variables p_{ij}, with $\forall i = 1, \ldots, N$ and $\forall j = 1, \ldots, M$. In total, there are $N \times |\Omega| = N \times N^K = N^{K+1}$ non-linear equations, where N is the number of providers, K is the number of users and N^K is the size of state space. The best response of all these non-linear equations must satisfy the users' budgetary constraints, i.e., $c_{ki} \leqslant b_k, \forall i \in [1, \ldots, N]$, and $\forall k \in [1, \ldots, K]$.

Two well-known algorithms to solve the maximization of systems of non-linear equations are Gauss-Jacobi and Gauss-Seidel [31], [32] which are both iterative-based methods. The main difference between them is the way to update the value and price policy in each iteration. We refer the reader to [33] for a more detailed discussion on both methods. In this paper, we advocate the Gauss-Seidel method as it has the advantage that information is used as soon as it becomes available. The full algorithm is shown in Algorithm 1. It is noted that, during the execution of the *while* loop (lines 4–12) whenever new values $\hat{V}_i^t(\boldsymbol{\omega})$ and $\mathbf{p}_i^t(\boldsymbol{\omega})$ are available, they are immediately used to replace the old values $\hat{V}_i^{t-1}(\boldsymbol{\omega})$, $\mathbf{p}_i^{t-1}(\boldsymbol{\omega})$ within the same iteration. In lines 6 and 7, the cc variable is defined as the convergence criterion; it is compared to a small number ϵ for stopping the algorithm.

5.2 Avoiding the curse of dimensionality

The key difficulty of the model is computing the expectation over all successor states in (16) and (17). As the size of the state space is N^K where N is the number of providers and K is the number of users, it will increase exponentially when N or K increases. Thus, the number of successor states also increases. The problem becomes very severe when applying this model to today's cloud market with many thousands of users. To cope with the explosion in the number of successor states, we assume that the changes in the providers' price policies cause the state transitions that are restricted to going one level up, one level down or staying the same, i.e., provider will gain, lose one user or stay with the same set of users. Thus, the number of successor states is only $K+1$. With a total of N providers, the total number of successor states is $N(K+1)$. Fig. 2 shows an example of the new set of successor states generated from a current state in the market with 2 providers and 4 users.

Applying this method to Algorithm 1, the *while* loop needs to be slightly modified such that from the current state $\boldsymbol{\omega} \in \Omega$, the algorithm first generates the new set of successor states and then uses this set to update the value function $\hat{V}_i^t(\boldsymbol{\omega})$ and the price policy $\mathbf{p}_i^t(\boldsymbol{\omega})$ for each provider according to Eqs. (18) and (19), respectively.

Algorithm 1 Finding optimal price policies $\hat{\mathbf{p}}$

Input: Users' resource requests and budgetary constraints: r_k and b_k, $k = 1, \ldots, K$. Information about all providers: φ_i, ψ_{ij}, λ_{ij}, c_{ij}^o and $c_{ij}^{\bar{o}}$, $i = 1, \ldots, N, j = 1, \ldots, M$. Discount factor: γ.

Output: Optimal price policies: $\hat{\mathbf{p}}$

1: Make initial guesses for the value function $\hat{V}_i^0(\boldsymbol{\omega}) \in \mathbb{R}$ and the price policy $\mathbf{p}_i^0(\boldsymbol{\omega}) \in \mathbb{R}_+^M$ for each provider $i = 1, \ldots, N$ in each state $\boldsymbol{\omega} \in \Omega$. Pick a random state to be the initial state of the market;
2: $stop \leftarrow 0$; /*stop condition*/; $t \leftarrow 0$ /*iteration*/
3: **while** $stop \neq 1$ **do**
4: Update the value function $\hat{V}_i^t(\boldsymbol{\omega})$ and the price policy $\mathbf{p}_i^t(\boldsymbol{\omega})$ for all providers according to Eqs. (18) and (19), respectively. In (18) and (19), $\mathbf{p}_{-i}^{t-1}$ refers to the price policies of providers other than provider i at iteration $t-1$;
5: $cc \leftarrow \max_{\boldsymbol{\omega} \in \Omega} \left| \left(\hat{V}^t(\boldsymbol{\omega}) - \hat{V}^{t-1}(\boldsymbol{\omega}) \right) / \left(1 + \hat{V}^t(\boldsymbol{\omega}) \right) \right|$;
6: **if** $cc < \epsilon$ **then** /*satisfied by all providers*/
7: $stop \leftarrow 1$;
8: **else**
9: Compute the next state of the market;
10: $t \leftarrow t + 1$;
11: **end if**
12: **end while**

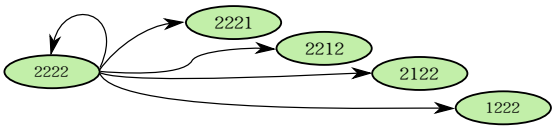


Fig. 2: Generation of successor states: the current state is 2222, which means all users choose to be served by Provider 2, there will be 5 successor states in which users can stay unchanged or switch to Provider 1.

6 Cooperation among Providers

In this section, we first present an overview on a Cloud-of-Clouds system. Then, we describe the cooperation

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2014.2322355, IEEE Transactions on Cloud Computing

method among providers based on the mathematical formulations of the operation cost and revenue. Finally, we present the algorithm for making cooperation decisions.

6.1 Overview of Cloud-of-Clouds system

Increasing resource demands with different requirements from users raise new challenges which a single provider may not be able to satisfy, given that the resilience of cloud services and the availability of data stored in the cloud are the most important issues. Scaling up the infrastructure might be a solution for each provider, but it costs a lot to do so, and the infrastructure may be under-utilized when demand is low. A multiple cloud approach, which is referred to as *Cloud-of-Clouds*, is a promising solution in which several providers cooperate to build up a Cloud-of-Clouds system for allocating resources to users. The Cloud-of-Clouds system can facilitate expense reduction (i.e., savings on the operation cost), avoiding adverse business impacts and offering cooperative or portable cloud services to users [34]. The architecture of a Cloud-of-Clouds system is depicted in Fig. 1 in which a dedicated broker is responsible for coordinating the cooperation among providers. The broker has all information about the resource capacities and price policies of all providers. Based on the users' resource requests, the broker will run a cooperation decision algorithm to decide with whom a particular provider should cooperate. The broker can be cloned on each provider's infrastructure and the cooperation decision algorithm will be executed when required by its owner. However, since price policies and resource capacities of providers change over time [8], keeping the consistency of this information for each version of the broker may not be easy. Therefore, we consider the case of a centralized algorithm run on the centralized Cloud-of-Clouds Broker to yield a global optimal solution.

The focused cooperation problem is the agreement among providers for outsourcing users' resource requests. Although providers are competing to attract users and improve their revenues, between any two providers, an outsourcing agreement may be established such that one provider can outsource its users' resource requests to its cooperator (or helper), i.e., satisfying users' resource requests by using the cooperator's infrastructure. However, how is the outsourcing cost calculated? Since providers are rational, the cooperation should result in a *win-win* situation where the provider who outsources its users' resource requests may pay a lower cost than satisfying them locally, and the provider who hosts outsourcing requests will receive the final revenue at least as much as that without cooperation.

It is to be noted that under the Cloud-of-Clouds model, many intertwined issues need to be considered before the system can operate efficiently. First, interoperability is one of the major issues. Every provider has its own way on how users or applications interact with the cloud infrastructure, leading to *cloud API propagation* [35]. This prevents the growth of the cloud ecosystem and limits cloud choice because of provider lock-in, lack of portability and the inability to use the services offered by multiple providers. An interoperability standard is therefore needed to enable users' applications on the Cloud-of-Clouds to be interoperable. Second, cooperation among providers requires a common Service Level Agreement (SLA) governing expected quality of service, resource usage and operation cost. Defining a "good faith" SLA allows the Cloud-of-Clouds to minimize conflicts which may occur during the negotiation among providers. One reason for the occurrence of these conflicts is that each provider must agree with the resources contributed by other providers against a set of its own policies. Another reason is the incurrence of high cooperation costs (e.g., network establishment, information transmission, capital flow) by the providers as they do not know with whom they should cooperate [34]. Last but not least, the network latency among providers' infrastructures also needs to be taken into account. It can be added as a constraint in the cooperation model to guarantee the service availability and satisfy the special requirements of users, and thus, may affect the optimal cooperation structure. In this paper, we only focus on the cooperation agreement among providers, and leave the study of the other issues mentioned above for future work. Hereafter, the term "hosting" provider is used to refer to the provider who satisfies its own users' requests (i.e., "stays local") and accepts outsourcing requests from other providers, and the term "outsourcing" provider is used to refer to the provider who outsources its users' resource requests to another provider and becomes idle.

6.2 Assumptions for the cooperation method

Several assumptions are needed to make our model simple but still reflect the real behaviors of providers. First, we assume that a provider can outsource its users' resource requests to only one provider at one time. All its users' resource requests will be satisfied by the selected provider and its local resources become idle. Satisfying a partial number of users' resource requests at the local site or sending to multiple providers may not be the best choice since the overhead of the operation cost at the local site plus the cooperation costs at the partners' sites may increase.

Second, a provider can accept as many as outsourcing requests without facing the limitation of resource capacities. This reflects our aforementioned assumption that a provider has enough resource capacity to satisfy all users' resource requests since supply is much higher than demand in the current cloud market. Additionally, with the quasi-unlimited capacities, the order of accepting outsourcing requests becomes unimportant. This assumption reflects the case of Amazon EC2 which has quasi-unlimited capacities and wants to have as many resource requests as possible to gain higher revenue [36].

Finally, a provider can refuse outsourcing requests if its total final revenue when hosting resource requests from others is less than that when outsourcing its own users' resource requests to another provider. In this case, outsourcing providers will choose the next highest provider if it exists. If not, they have to satisfy their users' resource requests locally.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2014.2322355, IEEE Transactions on Cloud Computing

6.3 Outsourcing cost and final revenue of providers

We now present the mathematical models for the outsourcing cost and the final revenue of both outsourcing provider and hosting provider. Recall that the final revenue of provider i satisfying its users' resource requests locally is defined as

$$R_i^{\text{local}}(\boldsymbol{\beta}_i, \mathbf{p}_i) = \sum_{k=1}^{K} \beta_{ik} c_{ki} - \sum_{j=1}^{M} \frac{c_{ij}^{o} a_{ij}^{1+\log_2 \varphi_i}}{1+\log_2 \varphi_i} - \sum_{j=1}^{M} \frac{c_{ij}^{\bar{o}} \left(\psi_{ij} - a_{ij}\right)^{1+\log_2 \varphi_i}}{1+\log_2 \varphi_i} \tag{20}$$

where the first term is the gross revenue received from users. The second and third terms are the total operation costs for active and idle resources, respectively.

When two providers cooperate, the resource usage cost, that a provider has to pay when outsourcing to the other, is cheaper than that paid by users for the same requests. We define the outsourcing cost of provider i which outsources its users' resource requests to provider i' as the difference in the operation cost when the amount of resources of provider i' allocated to provider i is active and idle. Mathematically, this cost is defined as follows:

$$\begin{aligned} C_{ii'}^{\text{outsource}}(\boldsymbol{\beta}_i, i') &= \sum_{j=1}^{M} \frac{c_{i'j}^{o} a_{ij}^{1+\log_2 \varphi_{i'}}}{1+\log_2 \varphi_{i'}} - \sum_{j=1}^{M} \frac{c_{i'j}^{\bar{o}} a_{ij}^{1+\log_2 \varphi_{i'}}}{1+\log_2 \varphi_{i'}} \\ &= \sum_{j=1}^{M} \frac{\left(c_{i'j}^{o} - c_{i'j}^{\bar{o}}\right) a_{ij}^{1+\log_2 \varphi_{i'}}}{1+\log_2 \varphi_{i'}} \end{aligned} \tag{21}$$

where a_{ij} is the number of instances of resource type j defined as $a_{ij} = \sum_{k=1}^{K} \beta_{ik} r_{kj}$, that are outsourced to provider i', and $\varphi_{i'}$ is the learning factor of provider i'. The final revenue of provider i when outsourcing its users' resource requests to provider i' is then

$$R_{ii'}^{\text{outsource}}(\boldsymbol{\beta}_i, \mathbf{p}_i, i') = \sum_{k=1}^{K} \beta_{ik} c_{ki} - \sum_{j=1}^{M} \frac{c_{ij}^{\bar{o}} \psi_{ij}^{1+\log_2 \varphi_i}}{1+\log_2 \varphi_i} - \sum_{j=1}^{M} \frac{\left(c_{i'j}^{o} - c_{i'j}^{\bar{o}}\right) a_{ij}^{1+\log_2 \varphi_{i'}}}{1+\log_2 \varphi_{i'}} \tag{22}$$

where the first term is the gross revenue that provider i receives from users, the second term is the operation cost for all local idle resources and the third term is the outsourcing cost that provider i pays to provider i'.

Finally, the total final revenue of provider i', which stays local to satisfy its own users' resource requests and accept outsourcing requests from other providers, can be formulated by combining Eqs. (4), (5) and (21). In detail, Let $\mathbb{L}_{i'}$ denote the set of all providers which outsource their users' resource requests to provider i'. We extend Eq. (4) to formulate the operation cost of provider i':

$$\begin{aligned} C_{i'}(\boldsymbol{\beta}_{i'}, \mathbb{L}_{i'}) = &\sum_{j=1}^{M} \frac{c_{i'j}^{o} \left(a_{i'j} + \sum_{l \in \mathbb{L}_{i'}} a_{lj}\right)^{1+\log_2 \varphi_{i'}}}{1+\log_2 \varphi_{i'}} \\ &+ \sum_{j=1}^{M} \frac{c_{i'j}^{\bar{o}} \left(\psi_{i'j} - a_{i'j} - \sum_{l \in \mathbb{L}_{i'}} a_{lj}\right)^{1+\log_2 \varphi_{i'}}}{1+\log_2 \varphi_{i'}}. \end{aligned} \tag{23}$$

The final revenue of provider i' when accepting outsourcing requests is then

$$\begin{aligned} R_{i'}^{\text{hosting}}(\boldsymbol{\beta}_{i'}, \mathbf{p}_{i'}, \mathbb{L}_{i'}) = R_{i'}^{\text{gross}}(\boldsymbol{\beta}_{i'}, \mathbf{p}_{i'}) + \sum_{l \in \mathbb{L}_{i'}} C_{li'}^{\text{outsource}}(\boldsymbol{\beta}_l, i') \\ - C_{i'}(\boldsymbol{\beta}_{i'}, \mathbb{L}_{i'}) \end{aligned} \tag{24}$$

where the first term is defined in (5), the second term is defined in (21) and the third term is defined in (23).

Provider i outsources its users' resource requests to provider i' if the final revenue of provider i when outsourcing is greater than that when satisfying all users' resource requests locally, i.e., $R_{ii'}^{\text{outsource}}(\boldsymbol{\beta}_i, \mathbf{p}_i, i') > R_i^{\text{local}}(\boldsymbol{\beta}_i, \mathbf{p}_i)$. Provider i' accepts outsourcing requests from other providers if it does not have a higher gain when outsourcing its own users' resource requests.

6.4 Algorithm for making cooperation decisions

In this section, we present an algorithm for determining the cooperation structure among providers. Since the full algorithm is quite long and complex, we first provide an overview of the algorithm in Algorithm 2 and then present a detailed analysis of the algorithm.

The algorithm starts by computing the final revenue of each provider when outsourcing its users' resource requests to each of the other $N-1$ providers, $R_{ii'}^{\text{outsource}}(\boldsymbol{\beta}_i, \mathbf{p}_i, i'), \forall i$, defined in (22). These $R_{ii'}^{\text{outsource}}(\boldsymbol{\beta}_i, \mathbf{p}_i, i')$ values are used later on in the algorithm to determine the best provider i' for provider i to outsource to, and also to compare with the sum of revenues when hosting local users' resource requests, i.e., "staying local", and when hosting users' resource requests from other provider's users, $R_{i'}^{\text{hosting}}(\boldsymbol{\beta}_{i'}, \mathbf{p}_{i'}, \mathbb{L}_{i'})$ defined in (24). This procedure is presented in lines 2–8.

The critical part of the algorithm is in the *while* loop, lines 10–38, where it has to make the decision for a provider who wants to outsource its users' resource requests, but may also have outsourcing requests from other providers, or its best provider to outsource to has not decided to become a hosting provider yet. The *while* loop stops when there is no more provider which can be a hosting provider, i.e., it has outsourcing requests from other providers. The *stop* flag is used to exit the *while* loop. In each iteration, a provider will be removed from $\mathbb{P}$, the set of all providers. This provider may be put into $\mathbb{P}_1$, the set of all hosting providers, or $\mathbb{P}_2$, the set of all outsourcing providers.

In the *for* loop, lines 12–17, the algorithm determines $\mathbb{L}_{i'}$, the set of providers who maximize their revenues when outsourcing their users' resource requests to provider i', by evaluating $R_{ii'}^{\text{outsource}}(\boldsymbol{\beta}_i, \mathbf{p}_i, i')$. Provider i belongs to $\mathbb{L}_{i'}$ if the condition $R_{ii'}^{\text{outsource}}(\boldsymbol{\beta}_i, \mathbf{p}_i, i') > R_{ii''}^{\text{outsource}}(\boldsymbol{\beta}_i, \mathbf{p}_i, i''), \forall i'' \neq i'$ holds. For example, Provider 1 can outsource its users' resource requests to Provider 2 or Provider 3 if the revenues when outsourcing to Provider 2 or 3 are higher than that when staying local, i.e., $R_{12}^{\text{outsource}}(\boldsymbol{\beta}_1, \mathbf{p}_1, 2) > R_1^{\text{local}}(\boldsymbol{\beta}_1, \mathbf{p}_1)$ and $R_{13}^{\text{outsource}}(\boldsymbol{\beta}_1, \mathbf{p}_1, 3) > R_1^{\text{local}}(\boldsymbol{\beta}_1, \mathbf{p}_1)$. Depending on the values of $R_{12}^{\text{outsource}}(\boldsymbol{\beta}_1, \mathbf{p}_1, 2)$ and $R_{13}^{\text{outsource}}(\boldsymbol{\beta}_1, \mathbf{p}_1, 3)$, Provider 1 will belong to the set $\mathbb{L}_2$ or $\mathbb{L}_3$. If $R_{12}^{\text{outsource}}(\boldsymbol{\beta}_1, \mathbf{p}_1, 2) > R_{13}^{\text{outsource}}(\boldsymbol{\beta}_1, \mathbf{p}_1, 3)$, then Provider

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2014.2322355, IEEE Transactions on Cloud Computing

Algorithm 2 Determination of the cooperation structure

Input: Users' resource requests: $r_k, k = 1, \ldots, K$. Information about all providers: φ_i, ψ_{ij}, c^o_{ij} and $c^{\bar{o}}_{ij}$, $i = 1, \ldots, N, j = 1, \ldots, M$. Price policies of all providers $\hat{\mathbf{p}}$.

Output: Cooperation structure.

1: $\mathbb{P} \leftarrow \{1, 2, \ldots, N\}$; /*set of all providers*/
2: **for** $i = 1 \rightarrow N$ **do**
3: **for** $i' = 1 \rightarrow N$ **do**
4: **if** $i \neq i'$ **then**
5: Compute $R^{\text{outsource}}_{ii'}(\boldsymbol{\beta}_i, \mathbf{p}_i, i')$ based on (22);
6: **end if**
7: **end for**
8: **end for**
9: $\mathbb{P}_1 \leftarrow \emptyset; \mathbb{P}_2 \leftarrow \emptyset; stop \leftarrow$ FALSE;
10: **while** $stop \neq$ TRUE **do**
11: $stop \leftarrow$ TRUE;
12: **for** $i' \in \mathbb{P} \setminus \{\mathbb{P}_1 \cup \mathbb{P}_2\}$ **do**
13: Find $\mathbb{L}_{i'}$; /*set of providers outsourcing to i'*/
14: **if** $\mathbb{L}_{i'} \neq \emptyset$ **then**
15: $stop \leftarrow$ FALSE;
16: **end if**
17: **end for**
18: $i^* \leftarrow 0; i^{*'} \leftarrow 0; maxval \leftarrow -realmax$;
19: **for** $i' \in \mathbb{P} \setminus \{\mathbb{P}_1 \cup \mathbb{P}_2\}$ and $\mathbb{L}_{i'} \neq \emptyset$ **do**
20: Compute $R^{\text{hosting}}_{i'}(\boldsymbol{\beta}_{i'}, \mathbf{p}_{i'}, \mathbb{L}_{i'})$ defined in (24);
21: Determine the best provider i'' for provider i' to outsource to by evaluating $R^{\text{outsource}}_{i',i''}$;
22: **if** $i'' \neq 0$ **then**
23: $R^{\text{diff}}_{i'} \leftarrow R^{\text{hosting}}_{i'} - R^{\text{outsource}}_{i',i''}$;
24: **else**
25: $R^{\text{diff}}_{i'} \leftarrow R^{\text{hosting}}_{i'}$;
26: **end if**
27: **if** $maxval < R^{\text{diff}}_{i'}$ **then**
28: $maxval \leftarrow R^{\text{diff}}_{i'}; i^* \leftarrow i'; i^{*'} \leftarrow i''$;
29: **end if**
30: **end for**
31: **if** $i^* \neq 0$ **then**
32: **if** $maxval \geqslant 0$ or $i^{*'} \notin \mathbb{P}_1$ **then**
33: $\mathbb{P}_1 \leftarrow \mathbb{P}_1 \cup \{i^*\}$; /*hosting provider*/
34: **else**
35: $\mathbb{P}_2 \leftarrow \mathbb{P}_2 \cup \{i^*\}$; /*outsourcing provider*/
36: **end if**
37: **end if**
38: **end while**
39: **if** $\mathbb{P} \setminus \{\mathbb{P}_1 \cup \mathbb{P}_2\} \neq \emptyset$ **then**
40: **for** $i \in \mathbb{P} \setminus \{\mathbb{P}_1 \cup \mathbb{P}_2\}$ **do**
41: Determine the best provider i' for provider i to outsource to;
42: **if** $i' \neq 0$ and $i' \in \mathbb{P}_1$ **then**
43: $\mathbb{P}_2 \leftarrow \mathbb{P}_2 \cup \{i\}$; /*outsourcing provider*/
44: **else**
45: $\mathbb{P}_1 \leftarrow \mathbb{P}_1 \cup \{i\}$; /*stay local*/
46: **end if**
47: **end for**
48: **end if**

1 belongs to set $\mathbb{L}_2$ since Provider 1 gains better revenue when outsourcing its users' resource requests to Provider 2 rather than Provider 3. Otherwise, Provider 1 belongs to set $\mathbb{L}_3$. If $R^{\text{local}}_i(\boldsymbol{\beta}_i, \mathbf{p}_i) \geqslant R^{\text{outsource}}_{ii'}(\boldsymbol{\beta}_i, \mathbf{p}_i, i'), \forall i'$, provider i will not belong to any $\mathbb{L}_{i'}$. Set $\mathbb{L}_{i'}$ is updated in each iteration of the *while* loop as the best provider i' for provider i to outsource to may decide to become an outsourcing provider rather than a hosting provider. In this case, provider i will choose the next highest provider i' by re-evaluating $R^{\text{outsource}}_{ii'}(\boldsymbol{\beta}_i, \mathbf{p}_i, i')$.

Lines 18–30 determine the next provider who will make a decision. For provider i' which has not yet decided and which may have several outsourcing requests from providers in $\mathbb{L}_{i'}$, the algorithm computes its total revenue when staying local and accepting all outsourcing requests from all providers in $\mathbb{L}_{i'}$, $R^{\text{hosting}}_{i'}(\boldsymbol{\beta}_{i'}, \mathbf{p}_{i'}, \mathbb{L}_{i'})$, defined in (24). The algorithm also determines the best provider i'' to which provider i' can outsource its own users' resource requests by evaluating $R^{\text{outsource}}_{i',i''}$. Let $R^{\text{diff}}_{i'}$ denote the difference in the revenue of provider i' between staying local and accepting outsourcing requests from other providers in $\mathbb{L}_{i'}$, versus outsourcing its own users' resource requests to its best provider i'', $R^{\text{diff}}_{i'} = R^{\text{hosting}}_{i'} - R^{\text{outsource}}_{i',i''}$. If provider i' gains the highest revenue when staying local even without hosting outsourcing requests, i.e., there is no provider i'' for provider i' to outsource to, $R^{\text{diff}}_{i'}$ is set to $R^{\text{hosting}}_{i'}$. The provider which has the highest value of $R^{\text{diff}}_{i'}$ and its best hosting provider are marked by i^* and $i^{*'}$, respectively.

Provider i^* is selected to make the final decision by executing lines 31–37. If $maxval$ (i.e., $R^{\text{diff}}_{i^*}$) is greater than or equal to 0, i.e., provider i^* gains a higher revenue when accepting outsourcing requests from other providers than that when it outsources its own users' resource requests to provider $i^{*'}$, the final decision of provider i^* is to stay local and to accept outsourcing requests. Provider i^* is put into set $\mathbb{P}_1$ as a hosting provider. If $maxval$ is negative, provider i^* prefers to outsource its users' resource requests to its best provider $i^{*'}$. The final decision of provider i^* depends on the decision of provider $i^{*'}$. Two cases can happen. First, if provider $i^{*'}$ is a hosting provider, the decision of provider i^* is to outsource its users' resource requests to provider $i^{*'}$. When that happens, provider i^* is put into set $\mathbb{P}_2$. Second, if provider $i^{*'}$ has not made its decision yet, then provider i^* must stay local and satisfy its users' resource requests using local resources. This is the final decision of provider i^* since provider $i^{*'}$ later on will also become an outsourcing provider as $R^{\text{diff}}_{i^{*'}} < R^{\text{diff}}_{i^*} < 0$. When that happens, provider i^* is put into set $\mathbb{P}_1$.

After exiting the *while* loop, all providers which can be a hosting provider have been processed. For the remaining providers in set $\mathbb{P}$, which have not yet made a decision, each will become either an outsourcing provider if its best provider to outsource to is a hosting provider (i.e., in set $\mathbb{P}_1$), this provider enters set $\mathbb{P}_2$. Or, it will stay local to satisfy only its own users' resource requests, this provider enters set $\mathbb{P}_1$ (lines 39–48).

It is noted that during the execution of the *while* loop, it can happen that provider i wants to outsource to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2014.2322355, IEEE Transactions on Cloud Computing

provider i' and provider i' also wants to outsource to provider i (i.e., "cyclic cooperation"). By choosing the provider which has the maximum value of R_i^{diff}, we give the privilege to the provider, who best improves the final revenue, to make the cooperation decision first and thereby solve the cyclic cooperation problem.

It is also worth noting that once providers are placed in set $\mathbb{P}_1$ or $\mathbb{P}_2$, they will not change the decision anymore. Indeed, if providers are in set $\mathbb{P}_1$, their final revenue when being a hosting provider and accepting outsourcing requests is higher than that when outsourcing its own users' resource requests. Hence, they will not change to become an outsourcing provider. Furthermore, if there is any outsourcing request from the others, the providers in set $\mathbb{P}_1$ will accept and gain even higher revenue according to the second assumption presented in Section 6.2. Similarly, a provider in set $\mathbb{P}_2$ does not have any outsourcing request from other providers and the revenue when outsourcing its own users' resource requests is higher than that when staying local. The matching process of provider i outsourcing its users' resource requests to its best hosting provider i' is done when provider i is put into set $\mathbb{P}_2$ (lines 35 and 43).

6.5 Optimality of the algorithm

With the assumptions presented in Section 6.2, we claim that the proposed algorithm yields the optimal cooperation structure in which the cooperation decision results in the highest revenue for each provider. Thus, no provider can unilaterally deviate to gain more revenue.

Proposition 6.1. *The cooperation structure found by the proposed algorithm is the optimal solution for all providers in the Cloud-of-Clouds based on the assumptions presented in Section 6.2.*

Proof: The proof of the above proposition is self-explanatory from the principle of revenue maximization. Indeed, as explained in Section 6.4, once provider i makes the decision which is its best solution based on the principle of revenue maximization, it will not change its decision anymore. The decision made by other providers later on will not change the decision of provider i as it gains more revenue when it is a hosting provider, or it gains nothing when it is an outsourcing provider. Due to the limit of space, we refer the reader to [37] for an example illustrating the optimality of the algorithm. □

It is worth noting that without the assumptions presented in Section 6.2, the problem will be more complex and require a more advanced formulation that is solved, e.g., by a combinatorial optimization method. For instance, the first assumption is said that a provider can outsource to only one provider, and when it decides to outsource, all its users' resource requests will be satisfied by the hosting provider. Without this assumption, outsourcing providers have to solve the problem of determining which user's resource request will be outsourced to which hosting provider. Furthermore, if we ignore the second assumption which relates to quasi-unlimited capacities of providers, then hosting providers have to determine which outsourcing requests will be accepted to maximize the revenue. They also have to consider the arrival order of outsourcing requests since it has an impact on their final revenues.

7 Numerical Simulations and Results

7.1 Parameter settings of simulations

For all simulations, we set the discount factor $\gamma = 0.95$ which corresponds to a 5% interest rate, the convergence constraint $\epsilon = 1e^{-4}$. The users' preferences which follow the Gumbel distribution are generated with the location parameter $\mu = 3$ and the scale parameter $\beta = 4$. We set the number of resource types offered by each provider $M = 4$ which is similar to that offered by Amazon EC2. We also use the per unit prices charged by Amazon to calculate the budgetary constraint for all users. The initial value for price policies of all providers is set to zero. Users' resource requests are classified into three classes: *small*, *medium* and *high* demand classes which require a tuple of resources $r_j = (6, 5, 4, 2)$, $r_j = (25, 20, 15, 8)$ and $r_j = (60, 50, 40, 20)$, respectively. This reflects the real user behavior that a user in the small class often requests a few number of resource instances for testing and experimental purpose while a user in the high demand class needs more resources for running their application on the production level. The number of users is set to 1024, except when explicitly indicated.

In the first group of simulations performed for validating the dynamic pricing and competition approach, the number of providers is set to $N = 2$ whose resources generate per unit benefit $\boldsymbol{\lambda}_1 = (0.1, 0.3, 0.4, 0.8)$ and $\boldsymbol{\lambda}_2 = (0.15, 0.25, 0.5, 0.7)$, respectively. By choosing $N = 2$, we simplify the simulation without loss of generality and retaining the competitive characteristics of the cloud market. Three simulations were examined:

1) The objective of the first simulation is to show the difference between the monopoly market with a sole provider and the oligopoly market with 2 providers;
2) The second simulation is to illustrate the evolution of the market to the equilibrium state; and
3) The last simulation is to show the convergence of the price policies and revenue of providers, and to assess the performance of the algorithm.

In the second group of simulations for validating the cooperation approach, we performed a large-scale simulation to illustrate the step-by-step execution of the cooperation decision algorithm.

Finally, we performed an intensive simulation to show the scalability of our model to a realistic market size and measured the running time of the proposed algorithms.

7.2 Analysis of results on dynamic resource pricing and competition

7.2.1 Monopoly versus oligopoly

With the same number of users and the same resource requests, it is expected that the revenue of the sole provider in a monopoly market is much higher than that in an oligopoly market since users do not have any choice. However, when there is competition in the market, users' resource requests are distributed among

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2014.2322355, IEEE Transactions on Cloud Computing

TABLE 2: Price policies and total revenue of providers when market is a monopoly and oligopoly

	Price policies				Total Revenue
	Type 1	Type 2	Type 3	Type 4	
Monopoly	0.32	0.40	0.48	0.77	30087.68
Oligopoly	0.32	0.40	0.47	0.78	09395.20
	0.32	0.39	0.47	0.77	20695.04

TABLE 3: Users' resource requests, budgetary constraints and their preferences to cloud providers

User index	Request class	Budget constraint	User's preference (η_{ki}) to	
			Provider 1	Provider 2
1	small	6.82	6.0570	1.7210
2	small	8.24	0.9966	1.0970
3	medium	28.21	8.5080	−0.9336
4	medium	31.73	10.7100	5.0810
5	high	100.60	3.4213	3.3910
6	small	8.49	0.2286	7.9860
7	medium	39.53	8.2780	4.3190
8	medium	27.09	2.4630	3.1840

the providers. Thus, the final revenues of providers decrease. In any case, the price policies are optimal since Algorithm 1 finds the price policies that maximize the revenue of providers while still assuring that usage costs do not exceed the users' budgetary constraints. As we can observe in Table 2, the sum of revenues of all providers in the oligopoly market is almost equal to the revenue of the sole provider in the monopoly market. However, depending on the users' resource requests, price policies are set accordingly. For instance, from Table 2, it can be observed that in the *Monopoly* simulation, the first three resource types (Types 1–3) are slightly more expensive than in the *Oligopoly* simulation while the fourth resource type (Type 4) is slightly cheaper.

7.2.2 Evolution of the market to the equilibrium

We performed this simulation with $K = 8$ users. This allows us to depict the evolution of the market as shown in Fig. 3. The users' resource requests and their preferences to providers are presented in Table 3. The results show that there always exists a unique MPE no matter what initial state is set. Indeed, we pick randomly any initial state among the 256 states in the state space, the game always converges to the equilibrium state $\boldsymbol{\omega} = (2, 2, 1, 1, 2, 2, 1, 1)$ as illustrated in Fig. 3. At the equilibrium state, the optimal price policies of the two providers are $\mathbf{p}_1 = (0.27, 0.35, 0.46, 0.82)$ and $\mathbf{p}_2 = (0.29, 0.31, 0.48, 0.82)$ with the optimal revenues $R_1 = \$88.09$ and $R_2 = \$108.94$, respectively. As illustrated by dotted arrows in Fig. 3, from the equilibrium state, providers have no incentive to unilaterally deviate from their optimal price policies to improve the revenue.

This simulation also shows that if users are indifferent to providers, they will be sensitive to prices. As shown in Table 3, User 5 slightly prefers Provider 1 to Provider 2, i.e., the preference of User 5 is 3.4213 to Provider 1 versus 3.3910 to Provider 2. However, the price policy of Provider 2 makes the utility of User 5 attain the maximal value. Thus, User 5 changes his provider to Provider 2 instead of Provider 1 at the equilibrium state. Yet, this simulation shows that although providers do not know the users' preferences and have to treat these variables as

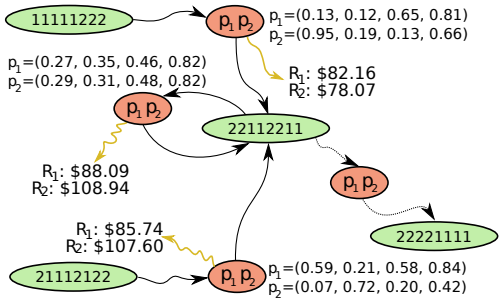


Fig. 3: Market's evolution with 2 providers and 8 users.

random, the logit discrete choice model allows providers to derive the accurate probability of being chosen by users. Adjusting price policies can increase or decrease the providers' chances of being chosen. Except for users who may be confused among providers and have to change their providers (i.e., User 5), all users are more satisfied with their preferred provider than with others.

7.2.3 Convergence of price policies and revenue

We present the convergence of the price policy of Provider 1 in Fig. 4a, and for Provider 2 in Fig. 4b, respectively. Fig. 4c shows the convergence of the discounted sum of future revenue of both providers. By comparing the prices of the same resource type offered by the two providers, it is expected that the price of resource types with higher benefit will be more expensive. In this simulation, all resource types 1, 2 and 3 follows this affirmation. However, if users are more familiar with the provider having the lower benefit resource, this provider can increase the price of this resource type to improve his revenue. In this simulation, resource type 4 of Provider 1 has a higher benefit than resource type 1 of Provider 2, i.e., $\lambda_{14} = 0.8$ versus $\lambda_{24} = 0.7$, but it is cheaper, i.e., $p_{14} = 0.80$ versus $p_{24} = 0.82$.

As shown in Figs. 4a and 4b, the price policies quickly converge to the equilibrium after 10 iterations. The number of iterations depends on the speed of convergence of the optimization solver used for solving the problem. However, our intensive simulations show that the algorithm always converges to the optimal solution. Furthermore, to avoid the convergence to the local optimum, we use the stopping condition which is based on the discounted sum of future revenue as depicted in Fig. 4c.

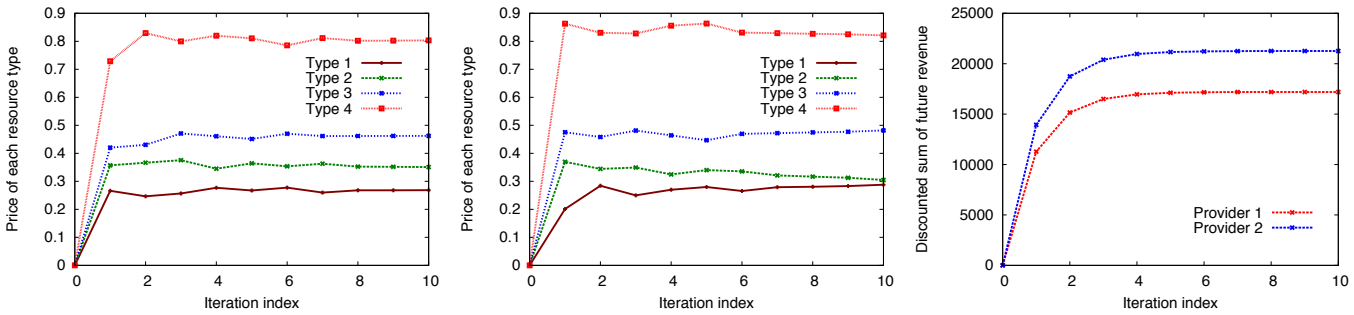
7.3 Analysis of results on cooperation

In this section, we present an evaluation of the cooperation model. We consider a Cloud-of-Clouds with $N = 8$ providers since it is large enough for a provider to choose its cooperation partner. The number of users' resource requests is set to $K = 512$. With the large number of users' resource requests submitted to a provider, the operation cost and revenue will be highly affected when the provider outsources its users' resource requests to its partner. We first present a simulation to illustrate the execution of the algorithm step-by-step and then analyze the dynamic changes of the cooperation decisions.

7.3.1 Algorithm validation

With the large number of users, we will not present all details of their resource requests. For providers, we

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2014.2322355, IEEE Transactions on Cloud Computing



(a) Resource prices of Provider 1. (b) Resource prices of Provider 2. (c) Discounted sum of future revenues.

Fig. 4: Convergence of price policies and discounted sum of future revenues of all providers.

TABLE 4: Simulation input data for illustrating the cooperation decision algorithm: infrastructure size and operation cost of the first active and idle instance of each resource type provided by each provider: the index is that of providers

Index	Infrastructure size				Operation cost of the first active instance				Operation cost of the first idle instance				Learning factor
	Type 1	Type 2	Type 3	Type 4	Type 1	Type 2	Type 3	Type 4	Type 1	Type 2	Type 3	Type 4	
1	2408	2479	2211	2340	0.06	0.07	0.09	0.10	0.03	0.03	0.05	0.05	0.79
2	2453	2483	2458	2379	0.06	0.07	0.08	0.10	0.02	0.03	0.04	0.05	0.75
3	2063	2078	2396	2372	0.07	0.08	0.08	0.10	0.02	0.04	0.04	0.05	0.76
4	2457	2486	2480	2196	0.07	0.07	0.08	0.09	0.03	0.03	0.05	0.06	0.88
5	2316	2479	2328	2328	0.07	0.07	0.08	0.09	0.02	0.03	0.04	0.05	0.86
6	2048	2243	2017	2085	0.07	0.07	0.08	0.10	0.02	0.04	0.05	0.05	0.80
7	2139	2400	2425	2353	0.07	0.08	0.08	0.09	0.02	0.03	0.04	0.05	0.90
8	2273	2071	2467	2015	0.06	0.07	0.09	0.09	0.02	0.04	0.04	0.05	0.75

TABLE 5: Price policies, users distributions and final revenues after competition

Provider Index	Price policies				#Users	Total Revenue
	Type 1	Type 2	Type 3	Type 4		
1	0.17	0.28	0.53	1.05	27	837.90
2	0.18	0.26	0.56	1.05	32	1105.80
3	0.17	0.29	0.53	1.06	47	1592.10
4	0.12	0.27	0.54	1.11	69	2197.40
5	0.16	0.24	0.57	1.07	55	1761.80
6	0.17	0.28	0.53	1.05	81	2737.20
7	0.13	0.25	0.56	1.11	95	3180.80
8	0.12	0.27	0.54	1.11	106	3830.80

present in Table 4 the infrastructure size which is the maximum number of resource instances offered by each provider, operation costs for the first active and idle resource instance, and the learning factor. With these input data, after running Algorithm 1 described in Section 5, we obtained the price policies and the distribution of users among providers as shown in Table 5.

Following the first part of Algorithm 2 (lines 2–8), the algorithm computes the final revenue of each provider when it outsources its users' resource requests to each of the other 7 providers. The values of $R_{i,i'}^{\text{outsource}}$ are shown in Table 6 where row provider represents i and column provider represents i'. When $i = i'$, $R_{i,i'}^{\text{outsource}}$, which is shown in bold text, is the revenue of provider i when satisfying users' resource requests locally. $R_{i,i'}^{\text{outsource}}$, which is in italics, is the higher revenue of provider i when outsourcing its users' resource requests to provider i' compared to the revenue when staying local. It is observed that Providers 4, 5 and 7 improve the final revenue when outsourcing to Providers 1, 2 and 3 since they have the higher learning factor which leads to the higher operation cost. Similarly, Provider 6 also improves the revenue when outsourcing to Provider 1. However, Providers 1, 2 and 3 also want to outsource their users' resource requests to their preferred provider.

In the first iteration of the *while* loop, the algorithm first determines the set of outsourcing providers for each provider by evaluating $R_{i,i'}^{\text{outsource}}$. Based on Table 6, we can see that the best provider for Providers 2, 3, 4, 5, 6 and 7 to outsource to is Provider 1. Thus, $\mathbb{L}_1 = \{2, \ldots, 7\}$. Without outsourcing requests from other providers, Provider 1 may want to outsource to Provider 2. Thus, $\mathbb{L}_2 = \{1\}$. For Providers 4, 5 and 7, though they can outsource to Provider 2 or 3, the best provider to which they can outsource and gain the highest revenue is Provider 1. Provider 8 does not belong to any set as it does not gain a higher revenue when outsourcing. Thus, we will have $\mathbb{L}_3 = \ldots = \mathbb{L}_8 = \emptyset$. The algorithm then executes lines 18–30 to determine which provider will make its decision first, i.e., choosing one provider among the providers who have outsourcing requests from the others. There are only Providers 1 and 2 which have outsourcing requests. The algorithm computes the total revenue of Provider 1 when hosting all outsourcing requests from providers in set $\mathbb{L}_1$ and does similarly for Provider 2. The total hosting revenue of Providers 1 and 2 is compared to their revenue when outsourcing. In this case, Provider 1 prefers to outsources to Provider 2 and Provider 2 wants to outsource to Provider 1. Table 7 presents the R_1^{diff} and R_2^{diff}values. As Provider 1 has the best improvement, i.e., $R_1^{\text{diff}} = 60.36$ versus $R_2^{\text{diff}} = 2.05$, Provider 1 is selected to make the decision. It is then put into set $\mathbb{P}_1$ as a hosting provider (lines 31–37).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2014.2322355, IEEE Transactions on Cloud Computing

TABLE 6: Outsourcing revenue of each provider to other providers in the market: the index is that of providers

Index	1	2	3	4	5	6	7	8
1	**837.9**	*850.0*	839.2	812.4	786.7	771.8	735.7	726.0
2	*1107.9*	**1105.8**	1095.5	1063.4	1032.8	1015.4	971.9	960.8
3	*1595.3*	1581.4	**1592.1**	1538.6	1498.3	1476.3	1417.6	1404.0
4	*2244.5*	*2227.2*	*2208.0*	**2197.4**	2154.2	2125.9	2045.4	2028.5
5	*1797.4*	*1782.3*	*1765.7*	1717.8	**1761.8**	1693.7	1627.6	1612.9
6	*2749.5*	2730.3	2709.0	2642.0	2580.1	**2737.2**	2486.4	2467.7
7	*3268.3*	*3247.1*	*3223.4*	3146.6	3076.0	3040.2	**3180.8**	3019.6
8	3822.1	3799.2	3773.7	3688.3	3610.2	3571.1	3450.9	**3830.8**

TABLE 7: Hosting revenue versus outsourcing revenue

Provider (i)	R_i^{hosting}	$R_{i,i'}^{\text{outsource}}$	R_i^{diff}
1	910.36	850.00 ($i'=2$)	60.36
2	1109.95	1107.90 ($i'=1$)	2.05

TABLE 8: Competition and cooperation revenue and improvement percentage: the index is that of providers

Index	Competition revenue	Cooperation revenue		Improvement percentage
1	850.00	910.36	(hosting)	7.10%
2	1105.80	1107.90	($i^*=1$)	0.19%
3	1592.10	1595.33	($i^*=1$)	0.20%
4	2197.42	2244.52	($i^*=1$)	2.14%
5	1761.84	1797.44	($i^*=1$)	2.02%
6	2737.21	2749.53	($i^*=1$)	0.45%
7	3180.81	3268.29	($i^*=1$)	2.75%
8	3830.80	3830.80	(locally)	0%

Starting the second iteration of the *while* loop, the algorithm updates the set of outsourcing providers. Only $\mathbb{L}_2$ is updated since Provider 1 has decided to be a hosting provider. $\mathbb{L}_2$ is then an empty set. As all Providers 2–8 do not have outsourcing requests, i.e., $\mathbb{L}_2 = \cdots = \mathbb{L}_8 = \emptyset$, the algorithm exits the *while* loop and executes lines 39–48. For Providers 2–7, their best provider to outsource to is Provider 1 which is in set $\mathbb{P}_1$. Hence, their cooperation decision is to outsource their users' resource requests to Provider 1 and become idle. Since Provider 8 does not gain a higher revenue when outsourcing, it decides to stay local to satisfy its own users' resource requests without any outsourcing request from other providers. The optimal cooperation decision and revenue's improvement of each provider are presented in Table 8. The revenue improvement is up to 7.10% due to cooperation, except for Provider 8.

7.3.2 Dynamic decision for cooperation

Depending on the market situation where users' resource requests and price policies of providers change dynamically, the cooperation decision will be adjusted accordingly. Since the cooperation model depends on many parameters, we could not perform simulations to determine the main parameter affecting the cooperation decision of providers, and to show different cooperation decisions in different market states. However, it can be expected that the providers, which have low users' demands, high operation cost and learning factor, tend to outsource their users' resource requests to a provider which has a lower operation cost and learning factor. This reflects the case from real life where a person can use his car or bus to travel. If he uses the bus, he will pay for the bus ticket and also the car park cost. If he uses his own car, he will pay for petrol but have a lower car park cost. Hence, the final decision depends on the route distance that he has to travel, per unit cost of petrol, car park cost and the bus ticket price. Comparing the total cost of travelling by bus and that of using his own car will give the final decision. Therefore, Algorithm 2 should be executed once the market state changes.

7.4 Scalability of the model

The proposed model is able to scale to a realistic size of the market within an acceptable time limit. On a Dell Optiplex 780 with 2 processors Core 2 Duo running at 3.16GHz with 4GB RAM, we run the simulation with 8 providers which corresponds to the number of biggest public providers in the current market. We exponentially increase the number of users up to 1024. For each value of the number of users, we execute the algorithm 10 times with different users' preferences and resource requests. It is expected that the execution time of the algorithm increases when the number of users increases. Despite the exponential increase of the state space size of the market, the proposed method of avoiding the curse of dimensionality reduces significantly the number of successor states that the algorithm has to compute the expectation. At the highest number of users, the improved algorithm generates the results after 1306.534 ± 222.657 seconds, while the simulation could not even run with the original algorithm. The standard deviation of the execution time is relatively high due to the convergence time of the Bellman equation. The scalability of our model would be even better if Algorithm 1 is implemented as a parallel program where each execution thread processes the expressions for each provider in the game.

8 Conclusion

In the current fiercely competitive cloud market, many providers are facing two major challenges: finding the optimal prices for resources to attract a common pool of potential users while maximizing their revenue in the presence of other competitors, and deciding whether to cooperate with their competitors to gain higher revenue after receiving their own users' resource requests. We presented a game-theoretic approach to address the former challenge. We integrated the discrete choice model, which describes the user's choice behavior based on the user's utility, to allow providers to derive the probability of being chosen by a user. By modelling the stochastic game as an MDP, our numerical results prove the existence of an MPE from which providers cannot unilaterally deviate to improve their revenue. Our algorithm,

which computes the equilibrium prices, is shown to converge quickly. Next, we introduced a novel approach for the cooperation among providers. The cooperation algorithm results in a win-win situation where both cooperation partners can improve their final revenue. The cooperation structure found by the algorithm is the optimal one for each provider. Thus, no provider has any incentive to unilaterally deviate to gain more revenue. The simulation results show that our approach is scalable and can be adopted by actual cloud providers.

REFERENCES

[1] H. Xu and B. Li, "Dynamic Cloud Pricing for Revenue Maximization," *IEEE Trans. Cloud Comput.*, vol. 1, no. 2, pp. 158–171, 2013.

[2] K. E. Train, "Discrete Choice Methods with Simulation," *Identity*, vol. 18, no. 3, pp. 273–383, 2003.

[3] M. J. Osborne, *An Introduction to Game Theory*. Oxford University Press, 2004.

[4] R. Bellman, "A Markovian Decision Process," *Indiana Univ. Math. J.*, vol. 6, no. 4, pp. 679–684, 1957.

[5] A. N. Toosi, R. K. Thulasiram, and R. Buyya, "Financial Option Market Model for Federated Cloud Environments," in *UCC 2012*, Chicago, Illinois, USA, Dec. 2012, pp. 3–12.

[6] A. Gera and C. H. Xia, "Learning Curves and Stochastic Models for Pricing and Provisioning Cloud Computing Services," *Service Science*, vol. 3, no. 1, pp. 99–109, Mar. 2011.

[7] B. Javadi, R. K. Thulasiram, and R. Buyya, "Characterizing spot price dynamics in public cloud environments," *Future Generation Computer Systems*, vol. 29, no. 4, pp. 988–999, Jun. 2013.

[8] B. Sharma, R. K. Thulasiram, P. Thulasiraman, S. K. Garg, and R. Buyya, "Pricing Cloud Compute Commodities: A Novel Financial Economic Model," in *CCGrid 2012*, Ottawa, Canada, May 2012, pp. 451–457.

[9] D. Niu, C. Feng, and B. Li, "Pricing Cloud Bandwidth Reservations under Demand Uncertainty," in *SIGMETRICS'12*, London, UK, June 2012, pp. 151–162.

[10] H. Xu and B. Li, "Maximizing Revenue with Dynamic Cloud Pricing: The Infinite Horizon Case," in *IEEE ICC 2012*, Ottawa, Canada, June 2012, pp. 2929–2933.

[11] F. Teng and F. Magoulès, "Resource Pricing and Equilibrium Allocation Policy in Cloud Computing," in *CIT 2010*, Bradford, UK, June 2010, pp. 195–202.

[12] M. Mihailescu and Y. M. Teo, "On Economic and Computational-Efficient Resource Pricing in Large Distributed Systems," in *CCGrid 2010*, Melbourne, Australia, May 2010, pp. 838–843.

[13] G. Allon and I. Gurvich, "Pricing and Dimensioning Competing Large-Scale Service Providers," *Manufacturing Service Operations Management*, vol. 12, no. 3, pp. 449–469, 2010.

[14] D. Bergemann and J. Välimäki, "Dynamic price competition," *Journal of Economic Theory*, vol. 127, no. 1, pp. 232–263, 2006.

[15] K. Y. Lin and S. Y. Sibdari, "Dynamic price competition with discrete customer choices," *European Journal Of Operational Research*, vol. 197, no. 3, pp. 969–980, 2009.

[16] S. U. Khan and I. Ahmad, "A Cooperative Game Theoretical Technique for Joint Optimization of Energy Consumption and Response Time in Computational Grids," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 3, pp. 346–360, 2009.

[17] S. Penmatsa and A. T. Chronopoulos, "Price-based User-optimal Job Allocation Scheme for Grid Systems," in *IEEE IPDPS 2006*, Rhodes Island, Greece, April 2006, pp. 336–343.

[18] Z. Kong, B. Tuffin, Y.-K. Kwok, and J. Wang, "Analysis of Duopoly Price Competition Between WLAN Providers," in *IEEE ICC 2009*, Dresden, Germany, June 2009, pp. 1–5.

[19] D. Niyato, A. V. Vasilakos, and Z. Kun, "Resource and Revenue Sharing with Coalition Formation of Cloud Providers: Game Theoretic Approach," in *CCGrid 2011*, Newport Beach, USA, May 2011, pp. 215–224.

[20] B. Boghosian, P. Coveney, S. Dong, L. Finn, S. Jha, G. Karniadakis, and N. Karonis, "NEKTAR, SPICE and Vortonics: using federated grids for large scale scientific applications," *Cluster Computing*, vol. 10, no. 3, pp. 351–364, 2007.

[21] M. Sobolewski and R. M. Kolonay, "Federated grid computing with interactive service-oriented programing," *Concurrent Engineering*, vol. 14, no. 1, pp. 55–66, 2006.

[22] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, M. Ben-Yehuda, W. Emmerich, and F. Galan, "The RESERVOIR Model and Architecture for Open Federated Cloud Computing," *IBM J Res Dev*, vol. 53, no. 4, pp. 535–545, 2009.

[23] I. Goiri, J. Guitart, and J. Torres, "Economic model of a Cloud provider operating in a federated Cloud," *Information Systems Frontiers*, vol. 14, no. 4, pp. 827–843, Sept. 2011.

[24] M. M. Hassan, M. S. Houssan, A. M. J. Sarkar, and E.-n. Huh, "Cooperative game-based distributed resource allocation in horizontal dynamic cloud federation platform," *Information Systems Frontiers*, pp. 1–20, June 2012.

[25] J. Filar and K. Vrieze, *Competitive Markov Decision Processes*. Springer-Verlag New York, Inc., 1996.

[26] P. Dubey, "Inefficiency of Nash Equilibria," *Mathematics of Operations Research*, vol. 11, no. 1, pp. 1–8, 1986.

[27] G. Martín-Herrán and J. Rincón-Zapatero, "Efficient Markov perfect Nash equilibria: theory and application to dynamic fishery games," *J. Econ. Dynam. Control*, vol. 29, no. 6, 2005.

[28] G. L. Lilien, P. Kotler, and K. S. Moorthy, *Marketing Models*. Prentice Hall, 1992.

[29] E. J. Gumbel, "Multivariate extremal distributions," *Bull. Inst. Internat. Statist.*, vol. 39, no. 2, pp. 471–475, 1962.

[30] R. A. Howard, *Dynamic Programming and Markov Processes*. The MIT Press, 1960.

[31] U. Doraszelski and K. L. Judd, "Avoiding the curse of dimensionality in dynamic stochastic games," *Quantitative Economics*, vol. 3, no. 1, pp. 53–93, 2012.

[32] U. Doraszelski and A. Pakes, "A Framework for Applied Dynamic Analysis in IO," in *Handbook of Industrial Organization*, ser. Handbooks in Economics, M. Armstrong and R. Porter, Eds. Elsevier, 2007, ch. 30, pp. 1887–1966.

[33] K. L. Judd, *Numerical Methods in Economics*. The MIT Press, 1998.

[34] M. Hassan, B. Song, and E.-N. Huh, "A market-oriented dynamic collaborative cloud services platform," *Ann. Telecommun.*, vol. 65, no. 11-12, pp. 669–688, 2010.

[35] A. V. Parameswaran and A. Chaddha, "Cloud Interoperability and Standardization," *SETLabs Briefings*, vol. 7, no. 7, 2009.

[36] J. Vargas and C. Sherwood, "Cloud Computing Capacity Planning: Maximizing Cloud Value," IBM, Tech. Rep., Nov. 2010.

[37] T. Truong-Huu and C.-K. Tham, "Competition and Cooperation Among Providers in a Cloud-of-Clouds Environment," National University of Singapore, Tech. Rep., Jan. 2014. [Online]. Available: http://137.132.153.61/techreport.pdf

Tram Truong-Huu received the MS from the Francophone Institute for Computer Science, Hanoi, Vietnam and the PhD degree in computer science from the University of Nice Sophia Antipolis, France in Oct. 2007 and Dec. 2010, respectively. He held a postdoctoral fellowship at the French National Center for Scientific Research (CNRS) from Jan. 2011 to Jun. 2012. From Jul. 2012, he is a research fellow at the Department of Electrical & Computer Engineering (ECE) of the National University of Singapore (NUS). His research interests include scientific workflow, grid, cloud and mobile cloud computing involving resource management, and competition and cooperation among cloud providers. He won the Best Presentation Recognition at IEEE/ACM UCC 2013.

Chen-Khong Tham received the MA and PhD degrees in electrical and information sciences engineering from the University of Cambridge, United Kingdom. He is an associate professor at the Department of Electrical & Computer Engineering (ECE) of the National University of Singapore (NUS). His current research focuses on cyberphysical systems involving wireless sensor networks and cloud computing, and cooperative and coordinated networks and distributed systems. His co-authors and he won the Best Paper awards at IEEE ICUWB 2008 and ACM MSWiM 2007. From 2007 to 2010, he was on secondment at the Institute for Infocomm Research (I2R) Singapore and served as principal scientist and department head of the Networking Protocols Dept and programme manager of the Personalization, Optimization & Intelligence through Services (POISe) Programme. He held a 2004/2005 Edward Clarence Dyason University Fellowship at the University of Melbourne, Australia. He is in the Editorial Board of the IEEE Internet of Things Journal and the International Journal of Network Management, and is/was the general chairman of IEEE SECON 2014, IEEE AINA 2011 and IEEE APSCC 2009.