Dynamic Pricing of Applications in Cloud Marketplaces using Game Theory

Safiye Ghasemi[1*], Mohammad Reza Meybodi[2], Mehdi Dehghan Takht Fooladi[2], and Amir Masoud Rahmani[1,3]

[1] Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran.
[2] Computer Engineering and Information Technology, Amirkabir University of Technology, Tehran, Iran. mmeybody@aut.ac.ir
[3] Computer Science, University of Human Development, Sulaimanyah, Iraq.

Abstract: Cloud marketplaces are known as new concerns in delivery of services; they have been recently attracted many researches. The competitive nature of such environments makes the pricing policies of services a crucial task for firms. We address this concern by designing a normal-form game between providers. A committee is considered in which providers register for improving their competition-based pricing policies. The functionality of game theory is applied to design dynamic pricing policies. The usage of the committee makes the game a complete-information one, in which each player is aware of every others' payoff functions. The players enhance their pricing policies to maximize their profits. The contribution of this paper is the quantitative modeling of Cloud marketplaces in form of a game in order to providing novel dynamic pricing strategies; the model is validated by proving the existence and the uniqueness of Nash equilibrium of the game.

Index Terms: Application, Cloud computing marketplace, competition-based pricing, game theory, Nash equilibrium.

1. Introduction

From 2007, Cloud computing has been emerged as one of the most attractive technologies in IT industry (Buyya, 2009; Hurwitz, 2010; Szabo, 2014; Zhang, 2014). An increasing number of companies are taking advantage of services provided by Cloud computing. The services are in terms of Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS), which are offered by providers. IaaS providers prepare computing resources and storage resources (Hurwitz, 2010; Di Valerio, 2013) in form of virtual machines (VMs). These computing resources are requested by PaaS/SaaS providers or industrial/academic organizations for their applications to be run without necessity to maintain underlying infrastructures (Di Valerio, 2013; Truong-Huu, 2014). Nowadays, many Cloud computing providers compete with each other in maximizing their profit. The competition between SaaS providers has been emerged as a new challenge in this area. SaaS providers offer software applications and the related services (Buyya, 2009). The competition of Cloud computing providers may make the market and dynamic prices be evolved over time. The pricing strategies of these providers must be economically efficient according to the competition between the providers (Di Valerio, 2013). There are a lot of researches that focus on pricing and strategic behavior in the Internet markets and Cloud computing (Anselmi, 2014). Different pricing researches are studied in (El-Fattah, 1976; Kauffman, 2013; Nan, 2013; Narahari, 2005; Yaïche, 2000). A price optimization approach in a free competitive market is investigated in (El-Fattah, 1976); the proposed approach, (Kauffman, 2013), maximizes users' profit by transferring their application to other providers or continuing using the current provider. Furthermore, (Nan, 2013) considered dynamic pricing mechanisms of providers with different levels of services; users select a proper level based on some parameters such as response time, security and storage capacity. Communication of competitive providers and users in form of a game model is presented in (Narahari, 2005); users tend to choose the services with the best quality (QoS) while several service providers cooperate with each other in an oligopoly market for attracting more and more users and increase the profits. However, just a few works exist on SaaS competition increasing the users of their applications. It is to be noted that the users usually prefer a provider with the minimum acceptable price. Thus, the offered prices have an essential impact on the number of the users. Besides pricing strategy has a significant influence on the profit-maximizing strategies of companies (Lehmann, 2009). Pricing of applications is computed by considering some properties, such as price formation, structure of payment flow, price discrimination, and assessment base (Narahari, 2005; Lehmann, 2009; Mathew, 2010).

Most researches, which have used game theory to model the interactions of providers or users, have mainly focused on optimal resource allocation in Cloud providers, but they did not study the competition-based pricing models especially for applications. In current paper, such pricing model is studied, which considers the pricing concerns of an application. We design a game of which each player is fully aware as well as the payoff function, known as a complete information one (Fudenberg, 1996). The players are SaaS providers, who tend to attract Cloud users to maximize their final profit; they try to compute a proper price for the current request. The strategies of players are the pricing policies. The unique features of our work are as follows. First, this work has an analytical insight into Cloud markets and provides a quantitative modeling of these markets in form of a game between providers. Goal of the considered game's players is to attract users by offering proper prices; and based on the game model, the equilibrium is computed using Nash equilibrium primitives. This paper lies in trying to capture the strategic dynamics of providers as a strategic form game and the competition-based pricing model, which considers the main parameters of pricing applications. Second, the work covers some novel considerations in mentioned pricing policy of applications (Narahari, 2005), which makes pricing more flexible.

The rest of this paper is organized as follows. In Section2, we discuss the preliminaries of SaaS providers' interactions in Cloud computing and introduce the game theory concepts. The proposed distributed algorithm, which leads to find the optimal solution for the application pricing game is formulated in Section3. The experimental results are reported in Section4. Finally, the paper is summarized with some concluding remarks in Section5.

2. Problem Statement and Notations

Cloud computing has a powerful paradigm in request processing for delivery of applications through provisioning of virtualized resources (Buyya, 2009). We suppose a Cloud computing marketplace which delivers application to the users; the overall architecture of the proposed Cloud market is depicted in Fig.1. Services of Cloud computing are consumed over the Internet; they can be accessed by users either via web browsers, directly or by the application programming interfaces (APIs), indirectly (Stanoevska, 2009).

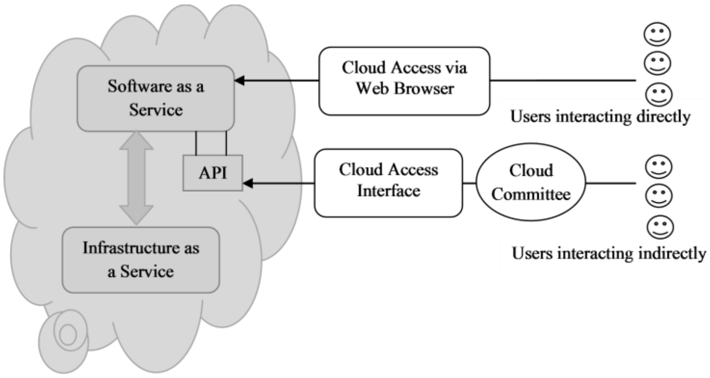


Fig.1 Cloud computing marketplace interacting with users via API (indirect) or via a web browser (direct)

A unit, named *Cloud Committee*, is considered in which users can join and demand applications via the API; it is a central coordinator between users and providers. The coordinator receives users' requests; then, it transfers a request to the registered providers. Such central unit cannot become a bottleneck as a request pool manages the entrance of requests to the unit.

After receiving the request, providers offer an optimal price for providing the requested application, and users are notified of prices by the committee. Finally, users contact to a SaaS provider whose service is the best based on its offered price and performance. It is to be noted that requests can be processed directly or indirectly. If a user demands services via the web browser of a SaaS provider (directly), the SaaS provider offers a price to the user without considering the offered prices of other available SaaS providers for the same services. Otherwise, in indirect access, providers, who have registered in the committee, perform dynamic comparative pricing. In the former case, the user demands required services from several SaaS providers separately, then, compares their offers, and chooses the best. In the latter case, the committee sends the demand from the user to all registered SaaS providers, and afterwards they offer their prices based on a game-theoretic model. Then the committee chooses the least one and notifies the user. The latter case is discussed in this paper.

2.1 User requests

As mentioned previously, a user demands applications from SaaS providers via *Cloud Committee*. A request consists of parameters such as application identification, required configuration, and time period of the request. The configuration is introduced in form of parameters of VMs such as type, memory, and price. These properties comprise a VM model named VMM as {*Size*, *Memory*, *Core*, *Storage*, *HostOS*, *HourCost*}. Per unit prices are determined to charge users for using the resources according to the *Size* of each virtual resource. The parameters *Memory*, *Core*, *Storage* and *HostOS* are used for finding the proper resources to host the demanded application; *HourCost* is applied for computing the operational cost of resources in a provider.

Table 1 System parameters

Notation	Declaration
Req_r	Request *r* of the Cloud market
q	Number of requests sent to providers of *Cloud Committee*
VMM	The considered model of virtual machine
τ_r	The duration of time the application is needed in Req_r
R_i	Number of VMs that provider *i* bought from IaaS provider
L_i	Number of applications that provider *i* bought from software developers
N	Number of SaaS providers or players of game DPG
α_i	Per unit benefit of virtual resources of provider *i*
β_i	Per unit benefit applications of provider *i*
S	Possible strategies of DPG
S_i	Possible strategies of SaaS provider *i* (player *i*)
s	Strategy profile of players in DPG
s_i	Selected strategy of player *i*
μ_i	Number of services of the requested application in Req_r
P_i	Offered price of SaaS provider *i* (player *i*) for Req_r
c_{ij}	Infrastructural cost that a provider needs to pay when providing resources for App_{ij}
C_i	Cost of providing Req_r for SaaS provider *i* (player *i*)
θ_{ij}	Price of application *j* in provider *i*
ω_i	The predefined parameter by provider *i*
$u_i(s)$	Payoff function for player *i* while playing strategy profile *s*

The requests are commonly gathered in the request pool in form of a vector named $\boldsymbol{REQ}$=<Req_1, Req_2, …, Req_q> by *Cloud Committee*, where Req_r demonstrates request *r* in $\boldsymbol{REQ}$ as { $AppID_r$, W_r, τ_r, Pay_r, Prf_r}. $AppID_r$ presents the requested application of Req_r, W_r is the willingness of the user to pay and it is the maximum amount that the user is willing to pay to use the application; it is somehow the budgetary constraint of the user for Req_r, τ_r represents the duration of time that the application $AppID_r$ is demanded, Pay_r is the payment flow of the user which may be single payment or regularly recurring payment, and Prf_r shows whether the user requires a certain performance level guarantees for a determined price or not; it is to be noted that such guarantees increase the price of the request. A more detailed discussion on the parameters is presented in Section2.3.

The adopted notations in this research are summarized in Table1.

2.2 Market state

The received requests are sent to the available SaaS providers simultaneously by *Cloud Committee*. Let SaaS provider *i* bought R_i numbers of VMs of different types; for each VM *k* that is offered by provider *i*, a per unit benefit is defined (Truong-Huu, 2014) as α_{ik} in $\boldsymbol{\alpha_i}$=<α_{i1}, α_{i2},…, α_{iRi}>.

Furthermore, provider *i* have L_i numbers of instances of the applications; each application *j* has an individual per unit benefit for the provider as β_{ij} in $\boldsymbol{\beta_i}$=<β_{i1}, β_{i2},…, β_{iLi}>. The applications can be multi-tenant; a single instance of a multi-tenant application serves multiple users. Although multi-tenant applications are more expensive, providers try to increase these applications as multi-tenancy can be economical; they have greater benefit for the providers as software development and maintenance costs are shared. The instances list of applications which provider *i* owns is $\boldsymbol{App_i}$=<App_{i1}, App_{i2},…, App_{iLi}>; application App_{ij}={$AppID_{ij}$, μ_{ij}, $\boldsymbol{Srv_{ij}}$, MT_{ij}, θ_{ij}}; where, $AppID_{ij}$ presents the application *j*'s identification in provider *i*; this application is assumed to consist of μ_{ij} services, $\boldsymbol{Srv_{ij}}$ is the list of services in the application in form of <VMM_{j1}, VMM_{j2}, …, $VMM_{j\mu ij}$>. Each service demands an individual VM type. It is to be noted that μ_{ij} and $\boldsymbol{Srv_{ij}}$ are dependent on provider which hosts the application; therefore, they are determined by each provider independently. MT_{ij} denotes the number of users able to own the application simultaneously; it is to be noted that MT_{ij} >1 for multi-tenant applications. θ_{ij} is the initial price of App_{ij} which is determined by its developer.

Suppose Req_r demands for $AppID_{ij}$; the requested application is App_{ij}. The cost that SaaS provider *i* has to pay IaaS provider for hosting App_{ij} is computed as

$$c_{ij} = \tau_r \times \sum_{k=1}^{\mu_{ij}} VMM_{jk}.HourCost, \qquad \forall k \in [1, \dots, \mu_{ij}]. \tag{1}$$

After receiving a request, the provider computes the price of processing the request. In the competition between SaaS providers, one may win due to its offered price while others lose.

2.3 Pricing models for the applications

Software products and requests have different properties, which affect the pricing strategies; these properties are extracted from available researches (Narahari, 2005; Lehmann, 2009; Mathew, 2010) as the following:

- **Initial cost**: the amount of money that the service provider spends for buying the software; this factor consists of the costs of the components of a SaaS based service.
- **Resource appropriation**: the efficient allocation of resources will help in reducing the wastage of resources and help keep the service as lean as possible. Eq.1 computes the resource appropriation of an application.
- **Multi-tenancy**: Number of users accessing the application simultaneously, which helps in lowering costs for the users and providers; SaaS providers can fully exploit the underlying technology.
- **User willingness to pay**: users determine an amount of money that they intend to pay for the application according to its realized value; providers do not have any knowledge of this factor, therefore, users determine it in the request.
- **Performance**: the SaaS provider guarantees a certain performance level for a determined price and pays a penalty in the case that the performance is not achieved.
- **Structure of payment flow**: users can make a single payment and thus obtain perpetual rights of use for the service, or can make a regularly recurring payment.

In this research, depending on the range of these factors, different levels of services are determined; the levels affect the pricing strategy introduced in Section3. For detailed parameters that lead to dynamic pricing, we refer reader to (Narahari, 2005; Lehmann, 2009; Mathew, 2010). Different values of these parameters comprise different states, which are introduced as service levels; Table2 depicts some of the states. Users can view the details of each level while requesting in the web page of *Cloud Committee*. The values of each parameter are assigned as follows. The value of 1 for Utilization parameter denotes that the resource appropriation of current application is the same as its requirements; false value for Multi-tenancy parameter indicates that the application is not a multi-tenant one, and true value shows a multi-tenant application. As mentioned previously, a multi-tenant application has a higher initial price, but as the deployment and maintenance costs are shared, users have a lower final price; Performance parameter is true, when the provider guarantees a certain performance level and determines a penalty of violation for the service, otherwise, it is false.

It is to be noted that when Utilization is less than 1, Performance cannot be guaranteed and it is false. Finally, Payment flow values can be single or recurring.

Services have different parameters which provide different levels of service (see Table2). Users can determine which level to access while requesting; for instance, if a user demands a typical application (not multi-tenant) with performance guarantees and single payment, then the level of service is one.

These introduced service levels are used in pricing strategies of providers introduced in Section3.3 The initial cost and resource appropriation parameters directly affect the offered price of requests; the remaining parameters affect the price by determining different service levels. Each level individually influences the offered prices. In addition to these factors, a service provider must consider the offered prices of other service providers as well. Competition-based pricing, which sets the prices according to other competitors' prices, is a potential dynamic pricing model. A dynamic pricing model of applications is proposed within this research by the aim of the game theory as follows.

Table 2 The information of service levels and the corresponding parameters of service

service level	Utilization	Multi-tenancy	Performance	payment flow	ω_i
Level 1	≥ 1	False	True	Single	[0.1,0.15,...,0.4]
Level 2	≥ 1	False	True	Recurring	[0.04,0.05,...,0.09]
Level 3	<1	True	False	Recurring	[0.006,0.009,..., 0.03]
Level 4	<1	True	False	Single	[0.001,0.002,..., 0.005]

3. Dynamic Pricing of Application Requests in a Competition-based Cloud Marketplace

This section firstly studies the formulation of our proposed approach for SaaS providers' pricing model with the aim of optimizing their profit. Then, in order to establish a competition-based pricing model the setup of a game between SaaS providers is described.

3.1 Proposed Architecture

The overall architecture of the considered Cloud computing market (Fig.1), with several SaaS providers, is depicted in Fig.2. The *Request Interface*, which is placed on top of the structure, is an interface unit for cumulated received requests, ***REQ***; it maps request *r* into the introduced form in Section2.1 as Req_r. The next unit is *Request Handler*, which consists of two modules: *Provisioning* and *Pricing*. These modules have main role in SaaS providers' processes; *Provisioning* module allocates the proper available VMs, stored in *Virtual Resources* unit of the provider; *Pricing* module determines a dynamic price for the current request, Req_r. *API* sends Req_r to SaaS providers, who are registered in the considered committee. The registered SaaS providers compete for serving Req_r. They perform allocation of resources and compute a pricing process; finally, SaaS provider *i* sends its offer to *Market Manager* in form of A_i.

Market Manager receives offers of SaaS providers, and stores them in a vector named ***s***. The offers are sent to the user of Req_r, in order to the most proper offer be chosen. *Market Manager* resends the overall information of the offers to the providers to inform the winner provider of the competition; this information is sent as a vector named $\boldsymbol{Rep_r}$={*winner_id*, ***s***}. As mentioned before, application *j* in provider *i*, App_{ij}, consists of μ_{ij} variant services. The requirements of service *m* are specified in terms of VM parameters as VMM_{jm} in $\boldsymbol{Srv_{ij}}$. In our model, the goal of a SaaS provider is to find the most proper price, while satisfying the requirements of the application. In next section, the formulation of a SaaS provider for achieving its goal is studied.

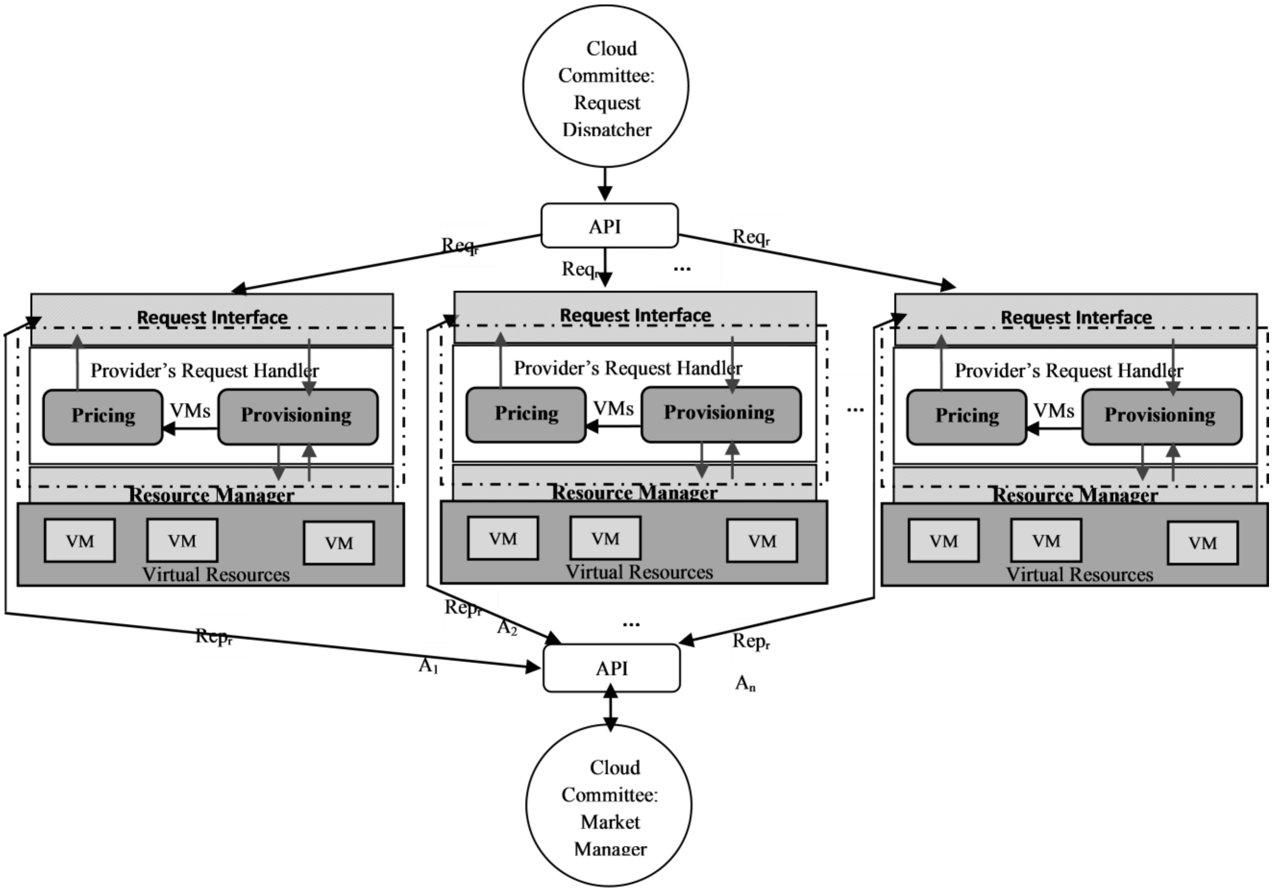


Fig. 2 The target Cloud computing marketplace model with the considered structure of a SaaS provider

3.2 Formulation of providers' strategies optimization

We assume that SaaS providers face an optimization problem for maximizing their profits, while satisfying users. Profit is one of the most important aspects that firms are fond of. The profit of a SaaS provider is the difference between its earned revenue of processing users' requests and its paid cost for providing applications and deploying them on virtual resources. The optimization problem of SaaS provider *i* is formulated as following.

$$
\begin{aligned}
& max\ u_i = max(P_i - C_i) = max\ P_i - min\, C_i \\
& s.t \qquad P_i \leq W_r \\
& \qquad\quad\ P_i \leq c_{ij} + \theta_{ij} \\
& \qquad\quad\ P_i > 0, C_i > 0,
\end{aligned}
\tag{2}
$$

where u_i is the profit of SaaS provider *i*; P_i and C_i are the revenue and the cost of SaaS provider *i*, respectively while provisioning Req_r; θ_{ij} denotes the initial cost that provider *i* has paid for owning the requested application, i.e. $AppID_{ij}$'s development cost; c_{ij} is the resource appropriation cost of the requested application *j* in provider *i*, introduced in Eq.1.

The constraints in Eq.2 are considered to guarantee some features as follows. The first constraint is that the offered price (P_i) should not exceed the users' willingness to pay. It is assumed that the users' willingness to pay cannot exceed the initial cost of the application *j* in provider i (θ_{ij}) and its deploying cost (c_{ij}) (Narahari, 2005), i.e., $W_r \leq c_{ij}+\theta_{ij}$. This assumption is applied by the committee to prevent users having low W_r. Second constraint is that the offered price does not exceed the sum of c_{ij} and θ_{ij}; otherwise users prefer to buy the required application of Req_r and its infrastructural requirements individually. The first two constraints result $P_i \leq W_r \leq c_{ij}+\theta_{ij}$. The last constraint denotes both the revenue and the cost are positive values.

A recommended solution to reach the goal is to maximize P_i while minimizing C_i, in such a way that the constraints are maintained; the ultimate value of P_i, which satisfies the constraints, is achieved by some parameters, which will be discussed later.

3.3. A game-theoretic setup

The interaction of SaaS providers can be modeled in form of a game. Hereafter, we formulate the game model for application pricing problem in the considered Cloud computing environment.

Definition1: Let DPG=(N,S,u) be a non-cooperative finite dynamic pricing game with complete information. N is a finite set of n SaaS providers in Cloud marketplace indexed by i; $S=S_1\times\cdots\times S_n$, where S_i is a finite set of strategies of provider i, which presents its pricing policies; $u=(u_1,...,u_n)$, where u_i is the payoff function of provider i. Let $\mathbf{s}=(s_1, ..., s_n)\epsilon S$ as the strategy profile, where $s_i \epsilon S_i$ is the strategy of the player i; s_i is chosen in a way that maximizes $u_i(\mathbf{s})$.

Players of the game are a set of SaaS providers of Cloud computing which registered in *Cloud Committee*. There are a rapidly growing number of SaaS providers; since there are a finite number of providers in Cloud environment (Hurwitz, 2010), we have a finite set of players, which is a necessity in a finite game. Users can easily find the latest list of SaaS providers that offer software solutions in their interested area. SaaS providers who register in *Cloud Committee* have a common database which makes DPG (dynamic pricing game) a complete information game.

SaaS provider can discriminate the prices (Narahari, 2005; Lehmann, 2009) according to the per unit benefits that each application and VM have; the price discrimination offers a same application to different users at different prices, depending on introduced factors in Section2.3.

We use a competition-based pricing model, realized by designing the introduced non-cooperative game, which has also benefited from price discrimination. S_i denotes the strategy set of provider i as,

$$S_i = \sqrt{\omega_i}\left(1+\gamma\sqrt{\omega_i}\right)\left(\theta_{ij}+c_{ij}\right), \tag{3}$$

where, ω_i is a parameter which is determined by provider i, γ is a constant value determined by the committee, less than 1.

Let ρ_r denote Utilization parameter for Req_r, which demands for $AppID_{ij}$, as

$$\rho_r = \frac{R_r}{R_i}. \tag{4}$$

Where R_r is the required infrastructures of requested application in Req_r, R_i denotes the provided resources for the request by provider i. The greater values of ρ_r guarantee a certain performance level for a determined price, otherwise, for less than 1 values of ρ_r ($R_r< R_i$) the offered price is discounted.

The other factors that influence ω_i are as follows; multi-tenancy state of the requested application, the user's interest for performance guarantees and payment flow, which are discussed in Section2.3. These factors determine different service levels. We determine different range of ω_i according to each service level. Let C_ρ, C_{MT}, C_{Pay} and C_{Prf} denote Utilization, Multi-tenancy, Payment flow, and Performance, respectively. The service levels are determined according to these parameters (see Table2); ω_i has different ranges of values based on each service level. Lower service levels have higher prices; i.e., for initial levels of service, high values of ω_i (e.g. [0.1,0.15,...,0.4]) are considered in order to not decrease the price of the provisioned request, these levels would have smaller discounted prices, and vice versa; the ranges of ω_i are discussed in Section5.1.

Service provisioning is a cost-prone process; however, there is a trade-off between revenues and costs. Formally, the payoff function, which introduces the profit of a provider, consists of both the properties of users' demand in Req_r, and the corresponding properties of the provided service.

Definition2: For the strategy profile $\boldsymbol{s} \in S$, $u_i: S \rightarrow \mathbb{R}$ is the payoff function, which assigns numerical values to each member of the strategy set S. For all $x, y \in S$, strategy x is preferred over strategy y iff $u_i(x) > u_i(y)$.

A payoff function is originated from software pricing principles discussed in Section2.1 (Narahari, 2005; Lehmann, 2009; Mathew, 2010), such as initial cost and multi-tenancy of application, resource appropriation costs, payment flow, utilization and performance of provided service. Payoff function of player i, for Req_r, is

$$u_i(\boldsymbol{s}) = D_i(S_i - C_i), \tag{5}$$

where S_i and C_i are the strategy of provider i for providing the request, respectively. $\boldsymbol{D}=\{D_1, D_2, \ldots, D_n\}$ denotes the demand vector of SaaS providers; $D_i=1$ if and only if $argmin(\boldsymbol{s})=i$, i.e. i is the index of the minimum of profile $\boldsymbol{s}$ and the strategy of player i has the least value in profile $\boldsymbol{s}$; therefore player i wins the game, otherwise it is zero. $u_i(\boldsymbol{s})$ not only depends on the strategy of provider i but also depends on all others', i.e. $\boldsymbol{s}=(s_1, s_2, \ldots, s_n)$. Users usually choose the least price for a satisfactory performance; therefore, the payoff is zero unless for the provider with the least price. C_i is computed as,

$$C_i = \omega_i\big(\alpha_j c_{ij} + \beta_{ij}\theta_{ij}\big). \tag{6}$$

ω_i is determined by each provider individually, it is applied to ensure the positivity of u_i, β_{ij} and α_j are per unit benefits of application j for provider i and per unit benefits of virtual resources that host Req_r, respectively; $\alpha_j = \sum_{k=1}^{\mu_{ij}} \alpha_{ik}$, where, α_{ik} denotes the per unit benefits of VM k for provider i.

Finally, the payoff function represented in Eq.5, is simplified as

$$u_i = D_i\left(\sqrt{\omega_i}\big(1+\gamma\sqrt{\omega_i}\big)\big(\theta_{ij} + c_{ij}\big) - \omega_i\big((\textstyle\sum_{k=1}^{\mu_{ij}} \alpha_{jk})c_{ij} + \beta_{ij}\theta_{ij}\big)\right). \tag{7}$$

The strategy profiles must converge to a desired profile, which is known as the solution concept of the game. This solution is named as Nash equilibrium in a normal form game; next section investigates the equilibrium.

Algorithm1 presents the considered game between SaaS providers for obtaining the best profit.

Algorithm 1 resource provisioning Game algorithm

The algorithm is run by each SaaS provider of the committee as CurrentPrv in a distributed manner.

Input:

Request of applications, Req_r; Information of SaaS providers: list of VMs and list of applications, their benefits $\boldsymbol{\alpha}$,

$\boldsymbol{\beta}$, γ, $\boldsymbol{D}$=0;

Output:

Optimal list of prices, BidList

1 Req_r = CloudCommittee.Dispatch();
2 **Do**
3 **foreach** Service in associated *AppID* of Req_r **do**
4 SelectedVMList[Service] = Select a proper VM for Service;
5 ω= Selectω(Req_r);
6 BidList[CurrentPrv] = $S_{\text{CurrentPrv}}$ (Req_r, θ_{AppID}, SelectedVMList, ω, γ);
7 Winner = MarketMgr(BidList);
8 **If** CurrentPrv **matches** Winner
9 Cost = $C_{\text{CurrentPrv}}$(Req_r, SelectedVMList, $\alpha_{\text{CurrentPrv}}$, $\beta_{\text{CurrentPrv}}$);
10 $D_{CurrentPrv}$=1;
11 $u_{\text{CurrentPrv}}$ = BidList[CurrentPrv] - Cost;
12 **If** MarketMgr() matches NE then
13 **Return** BidList**;**
14 **While** (1)

Algorithm1 presents the algorithm for game DPG. The output is the list of providers' pricing offers in Nash equilibrium. Firstly, the provider receives a request from *Request Dispatcher* of *Cloud Committee* in line1. Then, the game starts and it proceeds until the equilibrium achieved. Each provider provisions some virtual resources to deploy the requested application in lines3 and 4. After that, each provider specifies a value to ω via line5 (Table2); the price offer is computed in line6, according to Eq.3. The offered values of all providers are saved in *BidList*, which is a distributed memory between providers and the *Market Manager* of *Cloud Committee*. Line7 selects the winner of the game. The payoffs are computed through Eq.7 in lines8-11. Finally, achieving Nash equilibrium is checked in lines12 and 13; the process checks, whether any of the providers can get more benefit by changing the current offer, while other providers do not change their strategies. The complexity of this function is $n\times|S|$, where n is the number of available SaaS providers and $|S|$ is the size of the strategy set of the provider.

Next section investigates the properties of Nash equilibrium for the game.

4. Market Equilibrium

Cloud computing is a complex and heterogeneous distributed environment, in which management of the interactions between entities is a challenging task and needs automated and integrated intelligent strategies. States of SaaS providers in Cloud computing environment are unpredictable; therefore, predicting the behavior of providers accurately, would be a costly task. For this reason, we applied game theory to simplify the problem of dynamic pricing of applications in Cloud computing market; in this section, the properties, existence, and uniqueness of the solution of the game are studied.

4.1 Nash equilibrium conditions for the game

Unfortunately, the problem of finding Nash equilibrium of a general-sum game with n players cannot be formulated as a linear program (Shoham, 2008); thus, we cannot state the problem as an optimization problem as presented in Eq.2.

In our supposed DPG, providers determine pricing strategies, which satisfy them with their expected payoff, known as Nash equilibrium (Fudenberg, 1996). So, Nash equilibrium is an optimal criterion for DPG, which none of the SaaS providers can get more benefits by changing the selected strategy unilaterally; in Nash equilibrium, the assumption is that the other providers do not change their strategies.

As mentioned before, S_i, as the strategy of players, is a linear function of parameters related to the request and application, e.g., initial price of application, resources appropriation, and their benefit list. Each SaaS provider chooses the value of ω_i from a predefined range of finite values determined based on level of provided service (Table2). As presented in Algorithm1, they will continue choosing these values until the equilibrium condition is satisfied.

Hereafter, the existence of at least one equilibrium and its uniqueness will be studied.

4.2 Nash equilibrium existence and uniqueness

The termination condition of Algorithm1 in Section3.3 is to achieve Nash equilibrium; in Theorem1 it is proven and discussed.

Theorem1 There is at least one Nash equilibrium for DPG.

Proof Shoham (Shoham, 2008) has proven that every game with a finite number of players and a finite number of strategies has at least one Nash equilibrium. DPG has a finite number of players, which are SaaS providers in Cloud environment (Buyya, 2009). Besides, both strategy profile and payoff function of the game are finite, since their parameters have finite values. Therefore, Shoham's theorem verifies the existence of Nash equilibrium in this game. If the values of these parameters were chosen from a continuous value set, then catching Nash equilibrium in Algorithm1 (line12) would have the complexity of n^n; therefore, some other intelligent strategies would be needed. □

Finally, in order to guarantee the termination of the game the uniqueness of Nash equilibrium is proved as follows.

Theorem2 DPG has a unique Nash equilibrium.

Proof According to Theorem1 and the well-known Weierstrass theorem (De Branges, 1959), u_i is a closed function as it is a finite function (Rudin, 1964). The Weierstrass theorem guarantees that every function defined on a closed interval can be uniformly approximated by a polynomial function. This polynomial function can be assumed a linear function. According to these facts, $u_i(\boldsymbol{s})$ consists of several polynomial terms, which are linear. The concavity of $u_i(\boldsymbol{s})$ can be easily proved by studying its linear terms; as $ax+b$ can be supposed as a concave function, S_i is a concave one as well. On the other hand, C_i is an affine too, and it is concave. Consequently u_i, which is S_i-C_i is a concave function on convex set ω_i. It is to be noted that $u_i(\boldsymbol{s})$ is a second-order differentiable and concave function of its parameters (Chen, 2011), which guarantees the convergence of DPG.

Based on the concavity of $u_i(\boldsymbol{s})$, an equilibrium point of such game with a concave payoff function can be $\boldsymbol{s}^o$ as the following.

$$u_i(\boldsymbol{s}^o) = max_{y_i}\{u_i(\boldsymbol{s}_1^o, \cdots, y_i, \cdots, \boldsymbol{s}_n^o) | (\boldsymbol{s}_1^o, \cdots, y_i, \cdots, \boldsymbol{s}_n^o) \epsilon S\} \quad (i = 1, \cdots, n) \qquad (8)$$

At point $\boldsymbol{s}^o$ every provider stays in its best state and never changes the strategy while other strategies are unchanged. After considering the fact that the game is a concave one with n players, the uniqueness of Nash equilibrium, which is named $\boldsymbol{s}^o$ in Eq.8, is proved by using standard techniques according to (Rosen, 1965).

□

Based on Theorems1 and 2, DPG would have an individual Nash which is known as a solution concept. Thus, the game finds a solution for providers to reach the most available profit; in next section, this solution in a duopoly is discussed and the strategy of players in Nash is presented in form of a closed-form in duopoly.

4.3 Closed-form expression of the pricing strategy

In this section, the convergence of players to Nash equilibrium in a duopoly is studied; this is to be noted that the proof of a duopoly can be generalized to a scenario having more than two SaaS providers. Nash equilibrium price can be obtained through the best response function of each player in the non-cooperative game (Chen, 2011), i.e. $\boldsymbol{s}^*$ is considered as Nash equilibrium if s^*_i is the best response of provider *i*:

$$\begin{aligned} u_1(\boldsymbol{s}^*) &= u_1(s_1^*, s_2^*) \geq u_1(s_1, s_2^*) \quad ,\forall s_1 \epsilon S_1 \\ u_2(\boldsymbol{s}^*) &= u_2(s_1^*, s_2^*) \geq u_2(s_1^*, s_2) \quad ,\forall s_2 \epsilon S_2 \end{aligned} \tag{9}$$

The optimal $\boldsymbol{s}$ corresponding to maximal $u_i(\boldsymbol{s})$, which is the best response of provider *i*, is computed by differentiating $u_i(\boldsymbol{s})$ with respect to $\boldsymbol{s}$, then, it is set to zero, as

$$\frac{\partial u_i}{\partial s} = D_i\left(\frac{1}{2\sqrt{\omega_i}} + \gamma\right)\left(\theta_{ij} + c_{ij}\right) - D_i\left(\left(\sum_{k=1}^{\mu_{ij}} \alpha_{jk}\right)c_{ij} + \beta_{ij}\theta_{ij}\right). \tag{10}$$

Jointly solving the expression $\frac{\partial u_i}{\partial s} = 0$, the optimal pricing policy of provider *i* in a duopoly can be obtained as the closed-form expression of pricing policies. With a view to the parameters of s_i consists of γ, θ_{ij}, c_{ij}, and ω_i, γ is a constant coefficient determined by *Cloud Committee* marketplace, θ_{ij} is a value defined by the developer of the application, c_{ij} is computed based on resource appropriations, and ω_i, which is determined by provider *i*, equals to the following value in equilibrium point,

$$\omega_i{}^* = \left(\frac{\theta_{ij} + c_{ij}}{2\left(\left(\left(\sum_{k=1}^{\mu_{ij}} \alpha_{jk}\right)c_{ij} + \beta_{ij}\theta_{ij}\right) - \gamma\left(\theta_{ij} + c_{ij}\right)\right)}\right)^2. \tag{12}$$

The best response of each player in the considered duopoly is $\boldsymbol{s} = \left(\sqrt{\omega_1{}^*}\left(1 + \gamma\sqrt{\omega_1{}^*}\right)\left(\theta_{1j} + c_{1j}\right), \sqrt{\omega_2{}^*}\left(1 + \gamma\sqrt{\omega_2{}^*}\right)\left(\theta_{2j} + c_{2j}\right)\right)$. Consequently, closed-form expression of pricing policies of provider *i* in a duopoly is as $\sqrt{\omega_i{}^*}\left(1 + \gamma\sqrt{\omega_i{}^*}\right)\left(\theta_{ij} + c_{ij}\right)$ which is known as the solution of the duopoly market in DGP.

Table 3 Pricing defined by IaaS provider

Attr. / Size	VCPU	Memory (GB)	Storage (GB)	Price per VM/$
t2.small	1	2	1x 4 SSD	$0.026/Hour
t2.medium	2	4	1x 4 SSD	$0.052/Hour
m3.medium	1	3.75	1x 4 SSD	$0.070/Hour
c3.large	2	3.75	2x 16 SSD	$0.105/Hour
m3.large	2	7.5	1x 32 SSD	$0.140/Hour
R3.large	2	15	1x 32 SSD	$0.175/Hour

5. Performance Evaluation

In this section, we develop some experiments to analyze our model of SaaS providers' competition in Cloud computing marketplace. Firstly, we study parameter settings and performance metrics; then, the simulation configurations are explained, and finally the results are presented.

5.1 Experimental Setup

In this section, the parameters and the configuration of the model are clarified. The experiments are run on a semi Cloud computing marketplace, CloudSim toolkit 3.0.2, which is described as follows.

5.1.1 Parameters setting

The considered marketplace consists of multiple SaaS Cloud providers. SaaS providers initially owned random number of different type VMs. Three methods are added to implement the process of SaaS providers. The first method is used to determine whether the provider is capable of providing the received request according to its virtual resources or not. This method investigates the properties of available VMs by the aim of providers' *Resource Manager* and the requirements of each request by the aim of *Request Interface*. For simplicity, a single service is deployed on each VM. The second method finds the most proper VMs for deploying the requested application. This method chooses a VM, which is capable of deploying the service in a low per hour cost, for all services in the application. The third method specifies a price for supporting the request.

The parameters of VMs such as size, memory, and price which are considered according to what Amazon EC2 has defined (in December 2015). The parameters and the prices of considered VMs are listed in Table3. In our experiments, we assume that each VM hosts just one service. We extend *Vm* class of CloudSim to support the mentioned properties of *VMM* in Section2.1, according to Table3.

For all of performed simulations, γ is set to 0.95 (Truong-Huu, 2014); it corresponds to a 0.05 interest rate. The parameter ω_i is chosen from a finite set based on each service level. In the experiments, the values depicted in Table2. The probability distribution of values of ω_i is initialized as uniform distribution.

5.1.2 Simulation Configuration

Cloud environment is modeled in the form of an IaaS provider, several SaaS providers and some users. In our simulations, we assumed N=2, or 10, SaaS Cloud providers with a single IaaS provider and different provided VMs.

The requests of users are modeled as requests of applications, Req_r. These requests include execution-related requirements of applications such as memory, CPU usage, etc.

Moreover, each SaaS provider in our supposed Cloud computing marketplace owns multiple applications, and each application may consist of several services; the same list of ERP (Enterprise Resource Planning) applications is supposed in SaaS providers. Different ERP applications are provided from different SaaS providers; CRM is an ERP application, which has three main instances: Essential, Basic, and Professional. Some instances of Microsoft CRM applications and their potential costs are presented in Table4. Cloud applications' costs vary based on commercial fees [22]. Providers are monthly billed per user for online provisioning; the licensing prices are determined based on the instances, for on-premise provisioning. The prices of our applications are derived from ERP providers, such as Actionstep, iCIMS, Plex Systems and Host Analytics Inc.; the assumed values of parameters of the simulation are considered like (Truong-Huu, 2014) [22, 23].

Table 4 Considered applications offered by SaaS providers with their costs [22]

Type of Provision / Application's License	On-Premise	Online (per user per month)
CRM Server 2013	$4922	$150
CRM Professional User CAL	$983	$65
CRM Professional Device CAL	$787	$65
CRM Basic User CAL	$342	$30
CRM Basic Device CAL	$236	$30
CRM Essential CAL	$79	$15

5.2 Equilibrium Efficiency

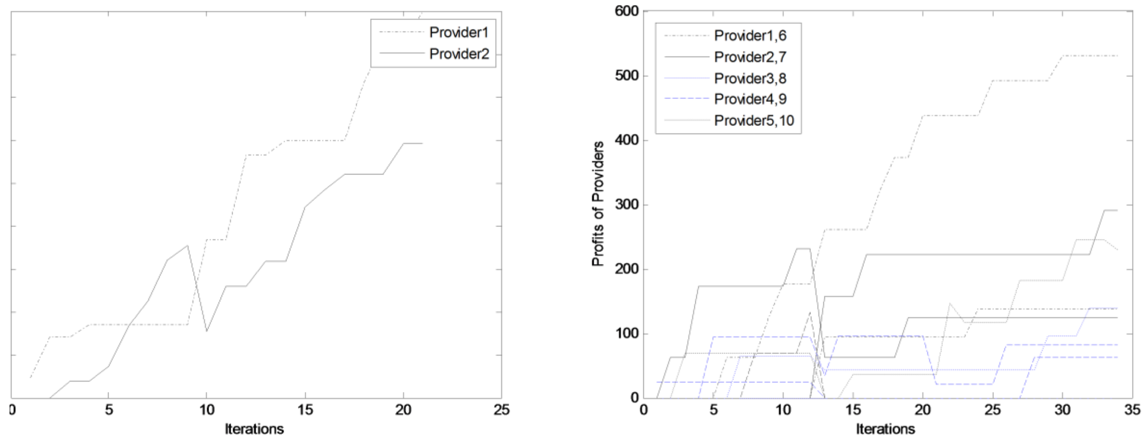
To validate the correctness of our proposed competition-based pricing approach, we run the experiments in a duopoly market with two SaaS providers, which is a two-player game scenario. In the following experiments the unit of profit and prices are $ and iteration denotes the number of repetitions which has not any unit. By such an assumption, we simplified the experiments, while retaining the competitive characteristics of the considered market. Fig.3a shows the profit of two providers while receiving different requests. As illustrated in this figure, while the game proceeds, both providers almost obtain an increasing profit.

Then, the experiments are performed for more than two providers to validate the approach in an oligopoly Cloud market. Fig.3b shows the profit of all participated providers with N=10 SaaS providers. From Fig.3b, it can be observed that while the game proceeds, the profits mostly increase as well. These experiments assess the performance of the competitive pricing mechanism. As it can be observed in Fig.3, the sum of profits of providers in duopoly and oligopoly with the same conditions are approximately equal to each other. One reason for large differences of profits of providers as depicted in Fig.3b is that these SaaS providers have different $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, which denote per unit benefits of resources and applications, respectively.

Fig.4 depicts the evolution of providers' offered prices, which is inserted as bids, in duopoly and oligopoly Cloud markets. Provider i chooses pricing parameter, ω_i, randomly, that turns out different prices. In each iteration, Nash equilibrium circumstances, introduced in Section4.1, are checked. As shown in Fig.4a, in 17th iteration, both providers reach Nash equilibrium, where their profit is better than any of other their offers. As depicted in the figure, the providers' offers are not changed after reaching Nash equilibrium; this state is called the convergence point of the game.

Fig.4b repeats the experiments for N=10 providers. In some cases, the game runs more than 100 iterations to reach the equilibrium. Providers reach Nash equilibrium in iteration 43rd, where their profit is better than any of other offer, while the other providers do not change their offers.

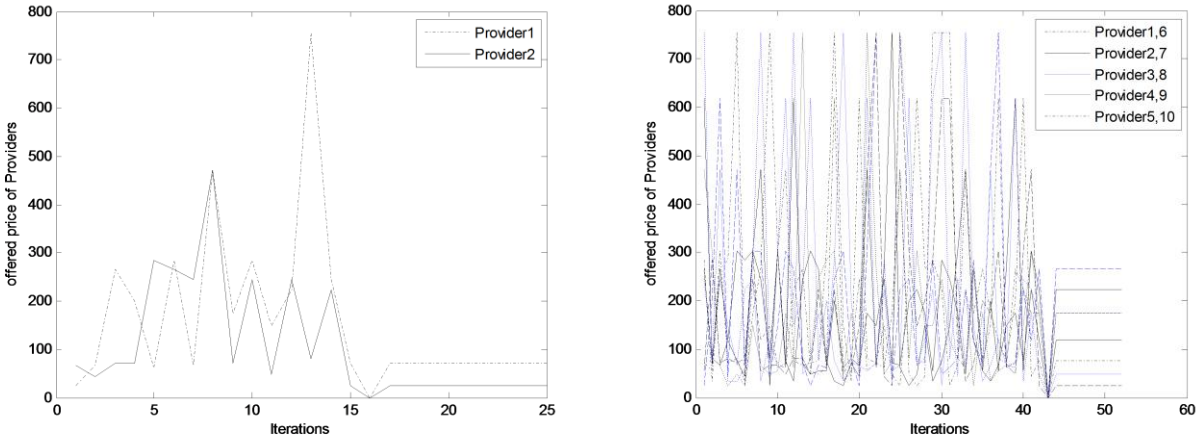
Comparing Fig.4a and Fig.4b, it is illustrated that in a multiple-players game, the convergence of pricing policy in an oligopoly needs longer run, which is expected as the growth of strategy profile of players. The simulation results verify that the considered game always converges to the optimal solution known as Nash Equilibrium. Actually, price offering of providers converges to the optimal price. The optimal price is the least one which satisfies the constraints in Eq.2. In Fig.4a it can be obviously observed that provider 2 is the winner of the game.



(a

(b)

Fig. 3 Payoff of providers in a (a) duopoly, (b) oligopoly Cloud computing market



(a) (b)

Fig. 4 Evolutions of SaaS providers' bids in a (a) duopoly, (b) oligopoly (N=10) Cloud computing market to equilibrium state

The states of providers, while Nash equilibrium is achieved, are depicted in Table5. It is to be noted that a demanding request of service level 1 is supposed; based on the same price of application (θ), and resource appropriation costs (c), different per unit benefits, different strategies (s_i) are generated. The winner is provider i=9; therefore, the profit of all players except 9 is actually zero; however, as illustrated in Table5, their imaginary profits, in case they are the winner, are presented to enable comparison.

Table 5 Presentation of providers' strategies parameters while γ=0.95, θ=65\$, c=295\$

Provider i	α_i	β_i	ω_i	s_i	$Cost_i$	$Profit_i$
1	0.1	0.4	0.001	11.20504	0.0599	11.14514
2	0.3	0.25	0.001	11.56334	0.0975	11.45984
3	0.8	0.25	0.001	11.20504	0.23945	10.96559
4	0.4	0.5	0.001	11.7262	0.1505	11.5757
5	0.2	0.4	0.001	11.23761	0.081	11.15661
6	0.8	0.15	0.001	12.21479	0.25125	11.96354
7	0.1	0.7	0.001	12.37766	0.077	12.30066
8	0.4	0.5	0.001	11.7262	0.1505	11.5757
9	0.2	0.15	0.001	11.0096	0.1026	10.94549
10	0.3	0.7	0.001	11.56334	0.10325	11.43084

Table 6 The length evolution of game while the players of the game increase

Criterion / Approach	Duopoly		Oligopoly	
	average of best price	average of profit	average of best price	average of profit
DPG	12.43	11.21	11.2	10.48
Price discovery	15	11	14.3	10.5

Finally, some experiments are performed to compare our approach with other methods to identify the breakthrough that has been achieved using DPG model. In (Muzaffar, 2017), a price discovery algorithm for searching the optimal price for a service with price-sensitive demand is studied which no information is required on reservation price. We compare our approach with (Muzaffar, 2017) by the average of the best price and the profit which providers get as criterions. For the experiments the same demand rate of the applications is assumed. The results are depicted in Table 6, as follows.

We perform DPG and Price discovery approach (Muzaffar, 2017) in both duopoly and oligopoly markets which have two and 10 SaaS providers. They have multiple instances of Microsoft CRM applications whose costs are presented in Table4. We considered providers equipped with these approaches; as depicted in Table 6, in both markets while using DPG, which considers a competitive pricing policy, best prices are less than (Muzaffar, 2017). Although the offered prices in our approach is less but the average of profits that providers gain is approximately the same in both approaches. The reason is that the number of requests that each provider may serve increases when it wins the game.

5.3 Validating the scalability of algorithm

Previous experiments considered at most N=10 SaaS providers in oligopoly markets; however, the proposed algorithm can scale to a realistic size of Cloud computing SaaS providers without disobeying time limits. On a Macbook Core 2 Duo running at 2.40GHz with 4.0GB RAM, the number of players is exponentially increased, with different requests and parameters. It is expected that as the players' size grows the execution time of the algorithm increases as well; it can be also concluded from Fig.4. As illustrated in Table 7, the affects that the growth of game's size has on the length of the algorithm's run is not exponential. The average numbers of iterations and the average of elapsed time of the game for reaching the equilibrium in ten runs are depicted in Table 7. The algorithm must check whether or not the equilibrium is achieved, in each iteration. As discussed previously, the order of Algoritm1 is $O(n^2)$. The longest time required for reaching the equilibrium, for 1024 SaaS providers, is 834 seconds.

Table 7 The length evolution of game while the players of the game increase

players	2	8	16	32	64	128	256	524	1024
Iteration to reach NE	27	40	48	80	100	110	120	125	140
Elapsed time	1.297	1.711	1.879	2.342	9.421	20.162	48.521	107.945	276.436

6. Conclusion

Recently Cloud computing has been emerged as a new information technology development, which has been noted as a services marketplace. Cloud computing marketplace faces with the competition and cooperation of its providers; this research focuses on the competition of Cloud providers. SaaS providers compete with each other by offering suitable resource provisioning with a desirable price. The scenario is modeled in a finite normal form game. Players of the game are SaaS providers; their strategies are considered as competition-based dynamic pricing policies based on different application properties; their preferences are the revenue, which is obtained by providing the request with offered price. We verified the existence and uniqueness of Nash equilibrium for the game. In addition, the experimental evaluations are performed and the theoretical evaluations are verified. Providers seek equilibria to perform an adaptive pricing strategy; and our considered game, which computes the preferred dynamic prices for each provider, converges to a unique Nash equilibrium, in which none of the providers tend to change their strategies.

Assumption of having just one IaaS provider can be omitted in the case of extending the model in the future to focus on resource provisioning techniques of providers, and also the infinite set of strategies is another issue for our future study.

References

[1] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, *25*(6), 599-616.

[2] Hurwitz, J., Bloor, R., Kaufman, M., & Halper, F. (2010). *Cloud computing for dummies*. John Wiley & Sons.

[3] Szabo, C., Sheng, Q. Z., Kroeger, T., Zhang, Y., & Yu, J. (2014). Science in the cloud: allocation and execution of data-intensive scientific workflows. *Journal of Grid Computing*, *12*(2), 245-264.

[4] Zhang, Q., Cheng, L., & Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, *1*(1), 7-18.

[5] Di Valerio, V., Cardellini, V., & Lo Presti, F. (2013, June). Optimal pricing and service provisioning strategies in cloud systems: a Stackelberg game approach. In *Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on* (pp. 115-122). IEEE.

[6] Truong-Huu, T., & Tham, C. K. (2014). A novel model for competition and cooperation among cloud providers. *Cloud Computing, IEEE Transactions on*, *2*(3), 251-265.

[7] Anselmi, J., Ardagna, D., Lui, J., Wierman, A., Xu, Y., & Yang, Z. (2014). The economics of the cloud: price competition and congestion. *ACM SIGecom Exchanges*, *13*(1), 58-63.

[8] El-Fattah, Y. M., & Henriksen, R. (1976). Simulation of market price formation as a game between stochastic automata. *Journal of Dynamic Systems, Measurement, and Control*, *98*(1), 91-100.

[9] Kauffman, R. J., & Ma, D. (2013). Cost Efficiency Strategy in the Software-as-a-Service Market: Modeling Results and Related Implementation Issues. In *Economics of Grids, Clouds, Systems, and Services* (pp. 16-28). Springer International Publishing.

[10] Nan, Gang, and Ya-min Wang. "QoS-Driven Dynamic Pricing Mechanism of SaaS in Cloud Services." In *The 19th International Conference on Industrial Engineering and Engineering Management*, pp. 939-948. Springer Berlin Heidelberg, 2013.

[11] Narahari, Y., Raju, C. V. L., Ravikumar, K., & Shah, S. (2005). Dynamic pricing models for electronic business. *Sadhana*, *30*(2-3), 231-256.

[12] Yaïche, H., Mazumdar, R. R., & Rosenberg, C. (2000). A game theoretic framework for bandwidth allocation and pricing in broadband networks. *IEEE/ACM Transactions on Networking (TON)*, *8*(5), 667-678.

[13] Lehmann, D. W. I. S., & Buxmann, P. (2009). Pricing strategies of software vendors. *Business & Information Systems Engineering*, *1*(6), 452-462.

[14] Mathew, M., & Nair, S. (2010). Pricing SaaS models: perceptions of business service providers and clients. *Journal of Services Research*, *10*(1), 51.

[15] Fudenberg, D., & Levine, D. K. (1998). *The theory of learning in games* (Vol. 2). MIT press.

[16] Stanoevska-Slabeva, Katarina, Thomas Wozniak, and Santi Ristol, eds. Grid and cloud computing: a business perspective on technology and applications. *Springer Science & Business Media*, 2009.

[17] Chen, Q-B., W-G. Zhou, Ruizhi Chai, and Linlin Tang. Game-theoretic approach for pricing strategy and network selection in heterogeneous wireless networks. *Communications, IET* 5, no. 5 (2011): 676-682.

[18] Shoham, Y., & Leyton-Brown, K. (2008). Multiagent systems: Algorithmic, game-theoretic, and logical foundations. Cambridge University Press.

[19] De Branges, L. (1959). The Stone-Weierstrass theorem. *Proceedings of the American Mathematical Society*, *10*(5), 822-824.

[20] Rudin, Walter. *Principles of mathematical analysis*. Vol. 3. New York: McGraw-Hill, 1964.

[21] Rosen, J. B. (1965). Existence and uniqueness of equilibrium points for concave n-person games. *Econometrica: Journal of the Econometric Society*, 520-534.
[22] Compares CRM software between more than 400 providers, http://www.softwareadvice.com/crm.
[23] AMAZON S3 TEAM. IDC: Quantifying the Business Value of Amazon Web Services. May 2015. Available from: https://aws.amazon.com/resources /analyst-reports.
[24] Muzaffar, Asif, Shiming Deng, and Ammar Rashid. "Non-parametric optimal service pricing: a simulation study." *Quality Technology & Quantitative Management* 14, no. 2 (2017): 142-155.