J Supercomput (2016) 72:317–346
DOI 10.1007/s11227-015-1565-y

Admission control in cloud computing using game theory

Gaurav Baranwal[1] · **Deo Prakash Vidyarthi**[1]

Published online: 25 November 2015

Abstract Cloud computing is emerging as a promising platform for ubiquitous computing where various types of resources are offered on pay-per-use basis. Cloud services are basically offered at three levels; infrastructure, platform and software. Service providers of these services are often interested to maximize their revenue and at the same time Cloud users expect for optimum quality of services. Sometimes, these two may conflict and admission of the requests is to be done that satisfies both Cloud providers and consumers. Game theory is a mathematical study of strategic decision making in which two players are involved in decision making based on their strategic moves. This work, applies the concept of game theory in admission control for Cloud requests. A model has been proposed and its performance study is done by simulating it in CloudSim simulator. Results are encouraging and may suggest for its possible inclusion in the Cloud middleware.

Keywords Cloud services · Admission control · Churning · Game theory · Nash equilibrium

1 Introduction

In current business world, Cloud computing has made the technology, specifically software, as ubiquitous. SaaS, a software service model of cloud computing, is the delivery of web-based software to customer over the internet, i.e., a user can access

✉ Deo Prakash Vidyarthi
dpv@mail.jnu.ac.in

Gaurav Baranwal
gaurav11_scs@jnu.ac.in; gaurav.vag@gmail.com

[1] School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India

the software through a web browser only. SaaS providers offer customers to access pre-deployed application over the Internet and customers do not have to bother about the license, the upgradation or the maintenance of the software. Users pay only for access to a particular application or software. SaaS providers are often interested in maximizing their profit and ensuring the quality of service (QoS) for its customers to enhance its reputation in the Cloud market. Currently, many corporate customers are migrating to SaaS leading to the emergence of many SaaS providers in the Cloud market. Thus, customers have plenty of opportunity to choose their service providers. Some works [1–3] even propose Cloud search engine that helps Cloud users in the selection of best Cloud provider according to their QoS requirements.

Since customers have the option to choose a suitable provider, it brings a new perspective in the Cloud market, called 'churning'. If a user is not satisfied with current service provider's offering, she/he may leave the current provider and switch to another suitable service provider, a process called as churning. Providers try to meet their customer's need without sacrificing its profit. Any provider may stop its services to the customer if that service is not fetching a gain to the provider. Cloud providers' decision space includes two decision problems [4]; first is to decide the cost that customer has to pay for a particular service and second is to accept a right kind of request in the system. Right kind of request here employs a request that increases provider's revenue as well as satisfies the customer. Many researchers, currently in the field of Cloud computing, focus on the second problem, i.e., to maximize the provider's revenue by admitting a request that gives maximum profit. As the Cloud market is quite global now, to consider only the revenue maximization does not suffice and the churning of customers should also be given a heed. For example, consider a situation in which an application service provider (ASP) provides online gaming services. A group of friends fetching the services from ASP may result in delay of services to some of its existing gamers because of resource overutilization. It is possible that some existing gamers, being unsatisfied from the current ASP, may switch their ASP. So, if a provider accepts all incoming requests, it may cause SLA (service level agreement) violation of accepted as well as upcoming requests resulting in resource overloading. Admission control has been used to deal with the aforementioned problem [5]. Thus, providers must use some admission control strategy to decide whether to accept a new request considering the load of existing requests being served by the provider.

Generally, Cloud users believe that resources available in a cloud are infinite which is not true. For example, a company can hire out its spare data center resources to some external users as the cost that an external user pays may contribute to amortize the expenditure of the data center. This work considers the SaaS providers in the Cloud with finite resources. The objective of a SaaS provider is to maximize its revenue as well as to satisfy its customers. If number of requests is more than the capacity of the provider, it decreases the customer's satisfaction as well as provider's revenue. The proposed model for admission control allows such requests that maximize the revenue of SaaS provider considering the churning behavior of the customers in satisfying the Quality of Service.

Game theory is a mathematical study of strategic decision making. Strategic decision makers, also called players, take the decisions from their available set of alternatives to optimize (maximize/minimize) some objective. Objective function for

a player depends on the actions (choices) of player and also its opponents. Thus, objective function of a player cannot be optimized independently of other player's action. This binds the players together in a decision making. In the present work, SaaS provider has to take decisions for admission of request based on the objective of revenue maximization and customer has to take decision of availing the provider's services based on the satisfactory performance in terms of service quality and cost. So, this whole scenario can be observed as a game where SaaS provider and customers are the players. Game theory is most appropriate to define this scenario and therefore is used to find the solution of this game.

Several challenges that arise, in designing the proposed model, are as follows. First is to define the admission control problem in form of a game. Second is to include churning behavior of the users in the game. Third is to formulate the game and to find an equilibrium solution of the game. Finally, a resource control component is to be defined to make the system feasible and stable. With aforementioned challenges, the contribution of the proposed work is as follows.

- Since main objective is to maximize the revenue, game is considered as $n + 1$-person non-cooperative game in which n players are n upcoming requests and another player is SaaS provider. For a specific request, this game is treated as two-player game to understand the real scenario of Cloud where requests come one by one.
- To define the churning behavior of the user, sigmoid function is considered to define the utility. Parameters for sigmoid function are QoS of the requests.
- Formulation of the proposed game achieves either Nash equilibrium in pure strategy or has dominant strategy for SaaS provider.
- Resource control component is defined which assign resources to the users based on their class and make the system feasible and stable.

The outline of this paper is as follows. After introduction in Sects. 1, 2 gives an overview of some work related to admission control and game theory. It also compares and contrasts the proposed work with other existing works. The architecture of the proposed framework is introduced in Sect. 3. In Sect. 4, admission control policy is described which includes formulation of admission control game, solution of admission control game and churning behavior of customer. Section 5 describes the resource control component. The performance study of the proposed model is done by simulation in Sect. 6, whereas Sect. 7 concludes the work highlighting some suggestions for future work.

2 Related work

Revenue maximization is an important issue while allocating the Cloud resources to the users. Load balancing, scheduling and resource provisioning, all three are important to maximize utilization of resource, etc., which in turn may increase revenue of the provider. With the introduction of service level agreement (SLA), to satisfy the customers and make them comfortable with the Cloud, some new issues have been emerged, e.g., provider has to pay some penalty to the user in case of SLA violation. So, with the objective of revenue maximization and customer satisfaction, only those

requests are to be admitted by the provider that meets the given two objectives before applying scheduling and load balancing algorithm [7]. A proper admission control policy should be in place to take such decisions and is an important research issue. This decision is based on some characteristic functions and constraints. Though, admission control is a fundamental problem in Cloud computing only a negligible amount of work is done on this leaving a good scope for contribution.

In [6] and [8], a general mechanism for admission control with SLA satisfaction by avoiding overutilization of resources is described. In [9], authors proposed a cloud auto-scaling mechanism to automatically scale up and scale down computing instances based on the dynamic workload information and application level performance need, i.e., deadline. In [9], authors created an admission control case in their mode; if budget of request of a customer is not sufficient, customer can leave the system otherwise provider will accept the request and try to run it on the instance with maximum computing power within the constraint of the customer's budget. Thus, [6,8,9] do not give a clear view or a dedicated model for admission control and [9] considers only deadline as a performance desire. Also, they do not incorporate any concept of SLA violation.

An auction-based admission control method is proposed in [10], which is Sybil immune (i.e., a user cannot increase its payoff by submitting a query that he/she cannot value) and strategy proof for continuous queries. User submits a query with a bid she is willing to pay for query processing. Admission control mechanism determines which query it should admit to maximize the profit and users maximize their payoff by bidding their true valuation for having their query run. But in [10], requests are represented by bid only as this work is focused on query only, whereas a request in Cloud may have its QoS requirement, e.g., deadline.

Another admission control policy for deployment of application in data centers is proposed in [11]. Data center may have many physical servers to deploy a requested application, but a situation may arise when demand is very high that the system is not able to handle it. Different requests may have different resource requirements and revenue margin. Markov decision process (MDP) is used in [11] to model this problem and approximate dynamic programming is used to optimize the admission control mechanism. Drawback with [11] is that it does not address QoS requirement of the request and also no attention is given to the customers that are ready to pay more. [4] introduced a policy-based approach that can predict provider's revenue and resource utilization achieved under stochastic demand. The work in [4] rejects the request on the basis of a reservation price only without considering the satisfaction of users.

The problem of optimal allocation of elastic services on virtualized physical resource has been identified and a probabilistic admission control approach is introduced to solve this problem in [12]. It incorporates a probabilistic approach to calculate extra computational resources required for a request for elasticity reasons and allows scaling of corresponding allocated physical resource to services. It incorporates SLA violation in the form of penalty. In [13], an adaptive provisioning technique is proposed based on a queuing model. It uses workload information to adapt the changes in the workload of an application and provides an adaptive management and guaranteed service time. It defines queue length to accept requests for each virtual machine; if

queue is full it does not accept any request. It also does not consider satisfaction of users for admitting the requests. Both [12,13] do not support QoS parameters such as budget and deadline. Also, in [12,13], there is no alternate strategy in case of limited resource availability.

In [14], authors proposed an autonomic pricing mechanism with an admission control policy that considers deadline and budget as hard QoS constraints. Therefore, this model do not incorporate SLA violation, i.e., penalty. A learning-based admission control in Cloud computing, proposed in [15], employs an automatically supervised naive Bayes classifier to classify incoming requests as admissible when request maximizes the utility of Cloud provider and otherwise non-admissible. It does not consider profit as a parameter in utility of provider.

In [5], authors proposed a model comprising of users, SaaS providers, IaaS providers, admission control and scheduling. SaaS provider may have its own resource or it may rent resources from IaaS provider. In former case, administrative cost and maintenance cost may be high and in latter case, QoS may be affected because of variable performance. In aforementioned work, admission control decides that whether to accept a request (by assigning the request to available resource or creating a virtual machine then assigning the request to the new virtual machine) or to reject the request on the basis of QoS parameters of the request. There is no resource control mechanism in [5] that gives priority to high-paying customers. Also, it does not consider the churning behavior of customers.

Some of the current works of admission control in Cloud computing, as discussed above, have many shortcomings. This work tries to remove the shortcomings of existing works as well as add other novel features for admission control. Some of the new features of the proposed model in comparison to the existing works are as follows; it is a dedicated model for admission control which can be applied before scheduling and load balancing; it defines QoS parameters such as budget and deadline for Cloud request and consider them as performance desire; it gives attention to customers who are ready to pay high to increase the profit of the provider; it considers churning behavior of customers to increase their satisfaction in case of limited resource availability; and it also considers imposition of penalty on provider in case of SLA violation by the provider.

The proposed work utilizes the concept of game theory to solve the admission control problem of Cloud computing. Goal is to maximize the SaaS provider's revenue and to satisfy the users' QoS requirement. Some of the related work that uses the concept of Game theory are as follows. Wireless network has used the concept of game theory for admission control [16,17] quite often and motivates researchers to implement admission control framework using game theory in Cloud computing. Though, game theory has been used in Cloud computing to solve some challenging issues, to the best of our knowledge, it has not been used for admission control in Cloud computing. For example, in [18], two aspects of resource allocation are considered: server load balancing and virtual machine placement. In first, assignment of virtual machine (VM) to physical machine (PM) is done to minimize the load on PM. In second, packing a set of VMs on minimum number of PM is done such that load on each physical machine is within the capacity of that PM. To address both these challenges, authors in [18] have used the concept of non-cooperative game and able to

get the pure Nash equilibrium solution. In [19], competitions among Cloud providers are described as a non-cooperative price and QoS games. Authors' of [19] analyzed three inter-organizational economic models of Cloud computing and found a unique pure Nash equilibrium solution in two models out of three. Analysis suggests the providers the price level for their existence in the market and QoS for end users for a particular service type. In [21], game theory is used to solve the resource allocation problem for QoS constrained and computation-intensive tasks using two steps. In first step, Binary Integer Programming is used to solve independent optimization problem of participant without considering multiplexing of resource assignments. In second step, evolutionary mechanism is designed that uses the result obtained from first step to change the multiplexed strategies of participant. Payoff of the participants, in this game, is efficiency loss which has to be minimized. It is shown that there is always a Nash equilibrium solution if the game has feasible solution. Ardagna et al. [20] consider a perspective in which SaaS provider takes resources from IaaS provider and offers services to the end users. In this, SaaS provider is interested to maximize its revenue and satisfaction of the end users and at the same time wishes to pay less to IaaS provider. On the other hand, IaaS provider looks to maximize its revenue by offering resources to SaaS provider. In this, authors proposed a solution by modeling the service provisioning problem as generalized Nash game.

Same as [20], in [45] SaaS provider has same objective, though it implements the provisioning scheme in two stages. In first stage, SaaS provider determines optimal number of on-demand and reserved instances using standard optimization technique and in second stage, SaaS provider bids for spot instances which are unused IaaS capacity [40]. SaaS provider models second stage as Stackelberg game and find equilibrium price. In [46], SaaS provider has same objective as [20,45], i.e., maximization of profit complying with QoS requirements of users, but it hosts application on PaaS provider unlike [20,45]. Authors modeled this scenario as Generalized Nash Equilibrium Price. In [20,45,46], SaaS provider is fetching services from IaaS provider or PaaS provider. Cloud federation is currently an emerging area in Cloud in which an IaaS provider can take services from other IaaS providers, if it feels that current available resources would not be able to satisfy the customers. IaaS provider, taking services from other IaaS providers, wants to maximize its profit. At the same time, IaaS providers, which offers these resources to requesting IaaS provider, also want to maximize their profit. To capture this conflicting and strategic scenario, in [47], a scenario of Cloud federation is considered and a cooperative game mechanism that motivates self-interested IaaS providers to participate in federation is implemented. Authors compared this work with non-cooperative game mechanism and observed that in latter one, there is no guarantee of optimal aggregated benefit received by the provider, while in former one providers cooperatively will be able to satisfy performance requirements of the tasks without increasing number of virtual machines. Above-discussed works are important works for problem solving using game theory in Cloud computing, though more works can be found in [48]. The above-discussed works though apply game theory for their respective problem solving but does not apply it for admission control in Cloud computing. The proposed work applies game theory for admission control in Cloud, an important aspect to admit jobs in Cloud environment.

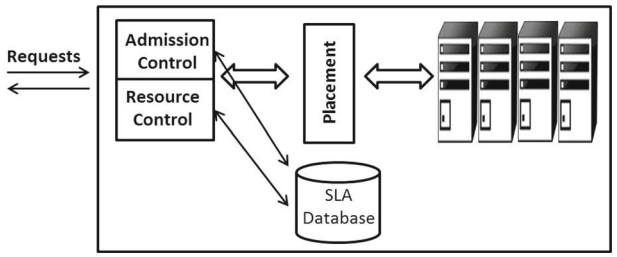

Fig. 1 Cloud framework

3 Cloud framework architecture

The architecture for the proposed Cloud framework is given in Fig. 1. Two components of architecture, admission control block and resource control block (A/R block) closely integrates users' requests for the Cloud services. When a new request arrives, admission control block decides whether to accept or reject the request. If request is accepted, it can influence all on-going accepted requests. The admission control has to accept the incoming requests only if it can maintain the QoS of currently running requests in Cloud. In Cloud, assigning equal amount of resources to each request is not appropriate as the requests may have different requirements and characteristics, e.g., the amount a user is willing to pay for processing of request. So, in the proposed framework, resources are assigned to the different customers according to their class. Admission control block also uses this request classification for the admission of requests. For experimental purposes, this work classifies the customers in three classes based on their budget. Thus, admission control and resource control block work together for admitting requests and resource assignment to the accepted requests based on their class. It does so to maximize SaaS provider's revenue. This A/R block is connected to the SLA database. The SLA database contains the information about customers and their service plans. This work assumes that SLA database is static, i.e., customer cannot change its required QoS in a particular session. It is also assumed that the SLA agreement does not change until one leaves the SaaS provider. If a customer wants to change its SLA agreement, customer needs to leave the provider and re-join as a fresh customer with new SLA agreement in the system. If a request is accepted it is sent to the placement block which places this request to a virtual machine created in SaaS provider's physical resource.

In this architecture, based on the QoS requirements of the requests, it is decided whether to accept or reject a request.

4 Admission control

SaaS provider aims twin objectives; maximizing its revenue and meeting the QoS. If it is not able to satisfy the customers, they may churn to some other provider. Since a SaaS provider has multiple applications to satisfy multiple customers, it is assumed that SaaS provider has to admit multiple requests. In this system, SaaS provider is considered as

one player whereas other players are multiple requests from the customers. Provider has to decide which request it should accept to maximize the revenue and customers need to decide whether they would continue with the current provider or have to churn based on their satisfaction level. This scenario can be formulated as $n+1$ player game, where n is the number of incoming request and 1 is SaaS provider. This $n+1$ game is incorporated into the admission control process. To understand a real online scenario of Cloud, where requests come one by one, $n+1$ game is simplified as two-player game with $n=1$. The nature of the game, formulation of the game and equilibrium solutions is to be identified.

4.1 Nature of Game

The proposed game for the resource provisioning has two properties; non-cooperative and non-zero sum.

Non-cooperative game

In the proposed game, both providers and customers want to maximize their payoff which is revenue for provider and service satisfaction for customers. If a customer is not satisfied, it may churn. Since the discussed goals of provider and customer are conflicting, both the players will not be cooperating with each other. Thus, the proposed game is non-cooperative.

Non-zero sum game

In two-player zero-sum game, sum of payoff of both the players should be one, i.e., if payoff of one player increases, payoff of other player decreases. But, in the proposed game, suppose resources are not fully utilized then SaaS provider can accept incoming new request without impacting QoS of accepted requests with increased provider's revenue. Thus, both the payoffs, SaaS provider (revenue) and customer (customer satisfaction), are increased. So, this game is non-zero sum game.

4.2 Game formulation

In the previous section, it is discussed that admission control algorithm runs when new requests are received by the system. Also, SaaS providing scenario can be formulated as $n+1$ player game. Further, nature of game is non-cooperative and non-zero sum. As the requests are coming from multiple users, the term requests and users are interchangeably used in this paper. When n requests reach to SaaS provider, the provider will decide which request(s) to accept and which one to reject based on some objective function which is to be optimized. Current users have to decide whether they would like to continue the services of the current service provider or would churn to some other provider based on some criteria. Admission of new requests will impact other users that may result in revenue loss. Therefore, the main concern while deriving the objective function for provider is the revenue formulation and modeling of churning. Satisfaction of users will be modeled as objective function for the users.

Notations, used in deriving the objective function to describe the game, are provided below.

K	Number of user's class
N_i	Number of users of class i currently in the system
$curRev$	Current revenue of SaaS provider
$earRev_i$	Average earned revenue of SaaS provider by accepting a request of a user of class i
$losRev_i$	Average revenue loss of SaaS provider when a customer of class i leaves the provider
U_i	Average utility of customer of class i
CR_i	Average churn rate of customer of class i
$wocPO$	Payoff of user without churning
$churnPen$	Penalty of user when user churns
q	A constant value to convert utility in form of money
W_a	Payoff of customer when provider accepts the request
W_b	Payoff of customer when provider rejects the request

4.2.1 Formulation of game for $n + 1$ players

Since, in the proposed game number of players is more than two, representation of this game is not possible in a plane, i.e., representation of outcomes and equilibrium solution of game is a difficult task. In the formulation of the game, it is assumed that SaaS provider has two strategies; to accept a request or to reject a request. Customer also has two strategies; to stay with the current provider or to leave the current provider. If n requests are coming at any time instance and provider has two strategies for each request, there will be 2^n possibility of candidates' policy. Each policy can be represented by a binary vector of length n. ith vector of candidate's policy of provider may be written as in Eq. 1.

$$PS_i = [s_{i1}, s_{i2}, \ldots\ldots s_{in}] \tag{1}$$

where $s_{ij} \epsilon \{1, 2\}$ (i.e., for each jth request, if $s_{ij} = 1$, provider will accept the request and if $s_{ij} = 2$, provider will reject the request) for $1 \leq j \leq n$ and $1 \leq i \leq 2^n$. Each customer j $(1 \leq j \leq n)$, seeking admission of its request also has two strategy $CS_j \epsilon \{1, 2\}$; if $CS_j = 1$, customer leaves the SaaS provider and if $CS_j = 2$, customer stays with the SaaS provider.

Provider's payoff matrix, PP, is of $(n + 1)$ dimension of size $2^n \times 2 \times \cdots \times 2$, as defined in Eq. 2.

$$PP = [pp_i, CS_1, CS_2, \ldots, CS_n]_{2^n \times 2 \times \cdots \times 2} \tag{2}$$

and value of each element of matrix PP is as given in Eq. 3.

$$\begin{aligned} pp_i, CS_1, CS_2, \ldots, CS_n = curRev + \sum\nolimits_{j \epsilon \{1,2\ldots,n\} \& s_{ij}=1} earRev_{k_j} \\ - \sum\nolimits_{c=1}^{K} N_c CR_c losRev_c - \sum\nolimits_{allCS_j=1} losRev_{k_j} \end{aligned} \tag{3}$$

Here, payoff of provider is revenue which includes current revenue and revenue earned by all other requests which the admission control component is going to accept, i.e., $\sum_{j\epsilon\{1,2...,n\}\&s_{ij}=1} earRev_{k_j}$. Since resources are fixed, when a new request enters in the system it will use the resources allocated by the resource control component (discussed in Sect. 5) to its class. This, in turn, may affect the already admitted requests of the same class and thus the revenue of the provider. To add this revenue loss for all admitted requests of a particular class in formulation of revenue of provider, the term $\sum_{c=1}^{K} N_c CR_c losRev_c$ is used. Revenue of provider also has to include loss of revenue incurred by those customers that may leave, i.e., $\sum_{all\ CS_j=1} losRev_{k_j}$.

Payoff matrix of jth customer, same as provider's payoff matrix,CP_j is (n+1) dimension of size $2^n \times 2 \times \cdots \times 2$, and defined as given in Eq. 4.

$$CP_j = \left[cp_i^j, CS_1, CS_2, \ldots, CS_n\right]_{2^n \times 2 \times \cdots \times 2} \tag{4}$$

But at the user side, payoff of a particular user depends on only the strategy of provider for this user and vice versa. So, CP_j is converted into 2×2 matrix (Eq. 5).

$$CP_j = \left[cp_{s_{ij}}, CS_j\right]_{2\times 2} \tag{5}$$

To simplify the expression 5, CP_j is written as in Eq. 6.

$$CP_j = [cp_{lm}^j]_{2\times 2} \tag{6}$$

where $l \epsilon \{1, 2\}$, i.e., provider's strategy and $m \epsilon \{1, 2\}$, i.e., customer's strategy as already discussed.

Let c be the class of customer j who is making a request. Payoff matrix of customer j can be defined as in Eq. 7.

$$cp_{lm}^j = w1 \times wocPO_{lm} - w2 \times churnPen \tag{7}$$

where $wocPO_{lm}$ is payoff of user without churning and $churnPen$ is penalty of user when user churns. $churnPen$ is considered because it is assumed that if a user is availing services from a SaaS provider, user signs some contract according to which user may be charged for early termination or user may lose some money already given to the provider. Also, user has to give some activation fee for the new subscription. $churnPen$ may vary from customer to customer. $w1$ and $w2$ are weights which act as tunable parameter for customer to reflect the preference of user for saving money and user's satisfaction, respectively. If customer stays with current provider, i.e., $m = 2$, $churnPen$ will be zero.

Payoff of customer, without churning ($wocPO_{lm}$), is defined in Eq. 8.

$$wocPO_{lm} = \begin{cases} U_c q + W_a & \text{for } l = 1 \\ 0 + W_b & \text{for } l = 2 \end{cases} \tag{8}$$

where U_c is the utility of customer of class c (discussed in Sect. 4.3), q is a constant value to convert utility in form of money, W_a is payoff of customer when provider

accepts the request and W_b is payoff of customer when provider rejects the request. Here, $W_a > W_b$, because payoff of customer will obviously be high when its request is admitted in comparison to when it is rejected.

4.2.2 Formulation of game for two players ($n = 1$)

If requests are coming one by one, there will be only one customer at a time-seeking admission for its request, i.e., $n = 1$. Let k be the class of incoming request. With the help of Eq. 2, provider's payoff matrix PP' for two-player game can be written as in Eq. 9.

$$PP' = [pp, CS]_{2\times 2} \tag{9}$$

Simplifying Eq. 9, PP' is written as in Eq. 10.

$$PP' = [pp_{lm}]_{2\times 2} \tag{10}$$

where $l \epsilon \{1, 2\}$, i.e., provider's strategy and $m \epsilon \{1, 2\}$, i.e., customer's strategy.

With the help of Eq. 3, value of each element of matrix PP' is as given in Eq. 11.

$$pp_{lm} = curRev + earRev_k - \sum_{c=1}^{K} N_c CR_c losRev_c - losRev_k \tag{11}$$

Payoff matrix of user in two-player game is same as in n+1 player game because payoff of a user depends on only the strategy of provider for this user and vice versa. So payoff matrix of user is as given in Eq. 12.

$$CP' = [cp_{lm}]_{2\times 2} \tag{12}$$

where $l\epsilon \{1, 2\}$, i.e., provider's strategy and $m\epsilon \{1, 2\}$, i.e., customer's strategy.

Value of each element of matrix CP', i.e., cp_{lm} can be obtained from Eqs. 7 and 8.

4.3 Modeling customer's churning behavior

In formulation of admission control game, churning has its own importance. In this section, churning rate CR_c for all class c customer has been derived, where $1 \leq c \leq K$.

Churning of Cloud users from one SaaS provider to another SaaS provider is an important and expected feature, if customer has the freedom to opt for the best provider. If a customer decides to churn, it implies that the customer is not getting its expected satisfaction from its current service provider. So, to define the churning behavior of a customer, customer satisfaction is to be quantified. Number of research models [22–25] exists for the prediction of churning behavior of customer, though mostly not relevant to Cloud. In this work, utility parameter is used to model the customers' churning behavior. In economics, utility is used to define the degree of customer satisfaction. Utility can be modeled by quantity, quality and cost of service. In Cloud, customer satisfaction depends on QoS and cost parameter of the service, so utility can be modeled by QoS and cost parameters. As described in [26] "Many natural

processes, including those of complex system learning curves, exhibit a progression from small beginnings that accelerates and approaches a climax over time. When a detailed description is lacking, a sigmoid function is often used". Therefore, in the research issues of Distributed computing and Cloud computing, many authors have defined utility with a sigmoid function [27–29]. One of the standard sigmoid functions, used for approximation of customer satisfaction in research, is given in Eq. 13.

$$S = \frac{1}{1 + \exp^{-\alpha(w-\beta)}} \tag{13}$$

In the proposed work, sigmoid function is modified a bit to incorporate QoS and cost parameter. Turnaround time is considered as QoS. Thus, in this work, a standard sigmoid function [30], which takes response time and cost as input and estimate utility, is directly used with some modification. In place of response time, turnaround time of a request is used. Also, since in the experiments (in Sect. 6), it is assumed that customer has specified its budget so the price that a customer is going to pay is fixed. Thus, price parameter is removed. Utility of user i is as given in Eq. 14.

$$U_i = \frac{1}{1 + \exp^{-\alpha_i[(dl_i - tt_i)]}} \tag{14}$$

where tt_i is turnaround time of request i, α_i determines the steepness of curve, dl_i is deadline defined by request i that determines the center of curve for QoS. α_i and dl_i can be tuned to customize the utility of user i. In other words, with the increase in turnaround time (tt_i) utility decreases if the turnaround time crosses the threshold dl_i. α_i determines how fast the utility decreases. So, if a customer is more satisfied its utility will be high leaving a little scope for the customer to churn. The churn rate of customer i may be given by Eqs. 15, 16 or 17 written in different forms.

$$CR_i = 1 - U_i \tag{15}$$

$$CR_i = 1 - \frac{1}{1 + \exp^{-\alpha_i[(dl_i - tt_i)]}} \tag{16}$$

$$CR_i = \frac{1}{1 + \exp^{-\alpha_i[(tt_i - dl_i)]}} \tag{17}$$

Churning rate of customer of class c is given by Eq. 18.

$$CR_c = \sum_{i=1}^{n} CR_i / N_c \tag{18}$$

where N_c is the number of customers in class c.

Utility of customers of class c is defined as in Eq. 19.

$$U_c = 1 - CR_c \tag{19}$$

Relative advantage of different utility function is an important subject of discussion in research, but beyond the scope of this paper. Though, in this work, utility function

is defined by the sigmoid function, it can be replaced by some other suitable function if required.

4.4 Equilibrium solutions of game

Before discussing about equilibrium solution of the proposed game, some important terms of game theory, used in this section, requires a brief elaboration.

Pure strategy When a player makes a specific move or action in every possible situation of the game, without involving randomization or any probability distribution, is called pure strategy. Payoff of all players is deterministic, when every player plays a pure strategy.

Mixed strategy When a player includes some probability with each possible strategy, it is called mixed strategy. A player uses mixed strategy when player is not able to select a best pure strategy and want to keep opponent(s) guessing.

Dominant strategy In game theory, dominant strategy of a player means the player has a strategy which yields better payoff than its other strategy no matter which remaining players select their strategy [31].

Nash equilibrium In non-cooperative game, Nash equilibrium is a solution concept (i.e., a set of strategy) one for each player such that no player gets better payoff by changing its own strategy assuming all other players do not change their strategy. In other words, if players are non-cooperative, it is a point where no player gets incentive by changing its strategy [32].

Let, there are N players denoted by P_i for $1 \leq i \leq N$ and m_i be the finite number of alternatives available to player P_i. Also, let the player P_i has the index set M_i taking the value from the set $\{1, 2, \ldots, m_i\}$. In a N-person game defined by the payoff matrices $PO_i = \left[a^i_{n_1,n_2,\ldots,n_N}\right]$, the N-tuple $(n_1^*, n_2^*, \ldots, n_N^*)$ is said to constitute a non-cooperative (Nash) equilibrium solution to the game, if the following N inequalities are satisfied for all $n_i \in M_i$, where $1 \leq i \leq N$

$$\begin{cases} a^1_{n_1^*,n_2^*,\ldots,n_N^*} \leq a^1_{n_1,n_2^*,n_3^*\ldots,n_N^*} \\ a^1_{n_1^*,n_2^*,\ldots,n_N^*} \leq a^1_{n_1^*,n_2,n_3^*\ldots,n_N^*} \\ \vdots \\ a^1_{n_1^*,n_2^*,\ldots,n_N^*} \leq a^1_{n_1^*,n_2^*,\ldots n_{N-1}^*,n_N} \end{cases}$$

In N-player non-cooperative game, there may be Nash equilibrium in pure strategy but there exists at least one non-cooperative (Nash) equilibrium solution in mixed strategy. However, finding a non-cooperative (Nash) equilibrium solution in pure strategy or in mixed strategy is an expensive computational process [32]. In the proposed admission control game, two cases are studied based on the utilization of resources for both $n+1$ player game and two-player game and it is proved that there is Nash equilibrium or dominant strategy for SaaS provider to take correct admission control decision. The two cases are as follows.

Case 1 When resources are not fully utilized for every class c where $1 \leq c \leq K$.

Case 2 When resources are fully utilized or overutilized for at least one class c where $1 \leq c \leq K$.

4.4.1 Equilibrium solutions of $n+1$ players game

In case 1, it is found that Nash equilibrium exists in pure strategy where provider's strategy is to accept all the incoming requests and all the users' strategies are to stay with provider.

Theorem 1 *When, for every class c, $1 \leq c \leq K$, resources are not fully utilized, there is a pure strategy Nash equilibrium in game which corresponds to candidate's policy of provider as* $\text{PS}^* = [1, 1, \ldots, 1]$*, i.e., SaaS provider admits all n requests and $CS_j = 2$, for $1 \leq j \leq n$, i.e., all users decide to stay with SaaS provider.*

Proof Since resources are not fully utilized for all the class c, churning rate $CR_c \approx 0$ where $1 \leq c \leq K$. In other words, $U_c \approx 1$. It has already been discussed that there are 2^n possibility of candidates' policy of provider and payoff value of provider is given in Eq. 3. This equation will have maximum value when provider admits all the requests and all the users stay with the provider. It is because in this case value of $\sum_{j \epsilon \{1,2\ldots,n\} \& s_{ij}=1} earRev_{k_j}$ will be maximum (since provider accepts all request), value of $\sum_{c=1}^{K} N_c CR_c losRev_c$ will be zero (since $CR_c \approx 0$) and value of $\sum_{all\, CS_j=1} losRev_{k_j}$ will also be zero (since all user stays with provider).

Since $U_c \approx 1$, user's payoff matrix $CP_j = [cp_{lm}^j]_{2\times 2}$ is reduced as shown in Eq. 20.

$$CP_j = \begin{bmatrix} cp_{11}^j = w1q + w1W_a - w2churnPen & cp_{12}^j = w1q + w1W_a \\ cp_{21}^j = w1W_b - w2churnPen & cp_{22}^j = w1W_b \end{bmatrix} \tag{20}$$

where $1 \leq j \leq n$. In this matrix, cp_{12}^j is greater than all other elements of matrix $(cp_{11}^j, cp_{21}^j, cp_{22}^j)$, so user j selects the strategy to stay with current provider, i.e., $CS_j = 2$, for $1 \leq j \leq n$, while provider admits the request.

From the above discussion, it is found that candidate's policy of provider as $PS^* = [1, 1, \ldots, 1]$ and $CS_j = 2$, for $1 \leq j \leq n$ constitutes Nash equilibrium for the game. Remaining possibilities of combination of strategy do not give Nash equilibrium.

In case 2, concept of dominant strategy is used for equilibrium. Since the main goal of this work is to find the best strategy for the provider, the concept of dominant strategy is used to solve this case in a simple manner. For simplicity, Eq. 3 is re-written as Eq. 21.

$$pp_i, CS_1, CS_2, \ldots, CS_n = curRev + PS_{\text{aff}}^i - CS_{\text{aff}} \tag{21}$$

where

$$PS_{\text{aff}}^i = \sum_{j \epsilon \{1,2\ldots,n\} \& s_{ij}=1} earRev_{k_j} - \sum_{c=1}^{K} N_c CR_c losRev_c \tag{22}$$

and

$$CS_{\text{aff}}\left(CS_1, CS_2, \ldots, CS_n\right) = \sum_{all\ CS_j=1} los Rev_{k_j} \tag{23}$$

Here, it is noted that PS^i_{aff} is only affected by the provider's strategy and CS_{aff} is affected only by the customer's strategy. □

Theorem 2 *When resources are fully utilized or overutilized for at least one class c where* $1 \leq c \leq K$*, there exists a dominant strategy* PS^d_{aff} *for SaaS provider such that*

$$\hat{D}\left(PS^d_{\text{aff}}\right) = \max\left(PS^j_{\text{aff}}\right) \quad for \quad 1 \leq j \leq n. \tag{24}$$

Proof In this game, there are 2^n possibility of candidates' policy of provider and since each user has two strategies, for n users there are 2^n possibility and for a given combination of users' strategies $(CS_1, CS_2, \ldots, CS_n)$, $CS_{\text{aff}}(\cdot)$ is fixed. So for any given combination of users' strategies, there exists dominant strategy PS^d_{aff} such that

$$pp_d, CS_1, CS_2, \ldots, CS_n \geq pp_i, CS_1, CS_2, \ldots, CS_n \text{ where } i \neq d \text{ and } 1 \leq i \leq 2^n \tag{25}$$

Since for each combination of provider's strategy, there are 2^n combination of user's strategies. If we choose PS^d_{aff}, definitely there is $CS_{\text{aff}}(PS_d)$ from the 2^n combination of user's strategies such that

$$CS_{\text{aff}}(PS_d) \geq CS_{\text{aff}}(PS_i) \quad \text{where } i \neq d \text{ and } 1 \leq i \leq 2^n \tag{26}$$

Since $\hat{D}\left(PS^d_{\text{aff}}\right)$ always exist, there is always a dominant strategy PS^d_{aff} for SaaS provider.

In case 1, provider admits all the requests and in case 2 provider needs to search dominant strategy from 2^n possibilities. To find a dominant strategy from 2^n possibilities is an NP-hard problem. Therefore, evolutionary approaches potentially can be applied to solve this problem as evolutionary approaches can better solve an NP-hard problem efficiently. For this, one needs to design an objective function for the parameter to be optimized in the problem. In this work, game theory helps to find the objective function of admission control problem analytically. Genetic algorithm (GA) [33] has been applied as an evolutionary approach to solve this problem. GA is a metaheuristic search inspired by the process of natural evolution and natural genetics and widely used to solve an optimization problem involving a big search space. GA starts with an initial population of chromosome represented as a string and is a potential solution of the problem. To measure the goodness of the solution, an objective function, called fitness function, is used. Some GA operators, e.g., crossover, mutation and selection are used to evolve the initial population towards a better solution. Structure of the chromosome, fitness function and the various GA operations, e.g., crossover, mutation, and selection specific to this problem are discussed as follows. □

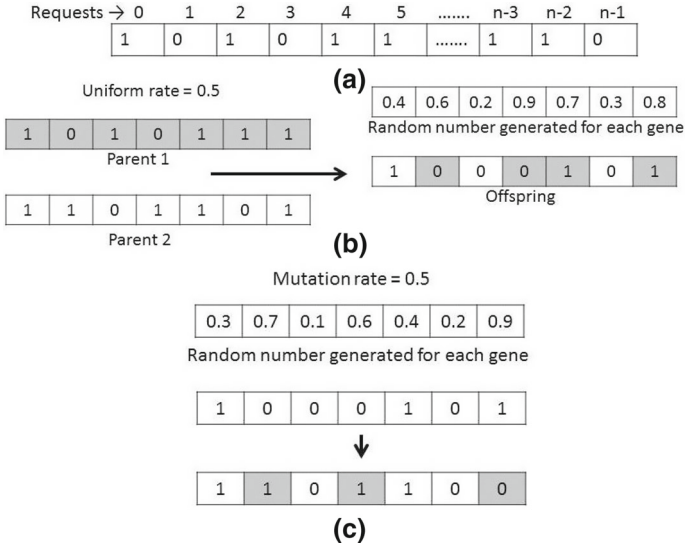

Fig. 2 **a** Chromosome representation. **b** Uniform crossover. **c** Mutation

Chromosome representation

In this problem, the size of the chromosome string is the number of requests n. Since SaaS provider has two strategies for each request, each gene in chromosome has two values. Binary encoding is used to represent the chromosome indicating 1 as accepted request and 0 as rejected. For example, Fig. 2a shows the provider's strategy for each request indicating that request 0 is accepted, request 1 is rejected and so on.

Fitness function

Fitness function is the objective function for the problem. In this problem, objective is to find the best set of request i out of n requests (i.e., 2^n possibility) for which PS^i_{aff} (Eq. 22) will have the maximum value. Objective function for this problem is as given in Eq. 27.

$$fitness = PS^i_{\text{aff}} \tag{27}$$

Substituting PS^i_{aff} from Eq. 22, fitness is as shown in Eq. 28.

$$fitness = \sum\nolimits_{j\epsilon\{1,2\ldots,n\}\&s_{ij}=1} earRev_{k_j} - \sum\nolimits_{c=1}^{K} N_c CR_c losRev_c \tag{28}$$

Crossover

Uniform crossover is used in this problem. In uniform crossover, a probabilistic uniform rate is defined for each gene of the chromosome. A random number is generated between 0 and 1 corresponding to each gene. If the random number is greater than uniform rate, gene from parent one is transferred to the child otherwise from gene from parent two is transferred. This way, a new chromosome is obtained. Figure 2b depicts the process of uniform crossover.

Mutation

For mutation, biological point mutation is used. In this technique, a mutation rate is defined and a random number is generated between 0 and 1 for each gene in the chromosome. If for a gene, random number is greater than mutation rate, that gene will be flipped (1 to 0 and vice versa) otherwise it remains same. Figure 2c shows the mutation process.

Selection

A predefined number of most fit chromosomes are selected by sorting all the chromosomes of a population in each generation.

Since main goal of the proposed work is to find the best strategy for SaaS provider, Theorems 1 and 2 are sufficient for the proposed admission control method.

4.4.2 Equilibrium solutions of two-player game

Two cases of $n+1$ game are simplified as a two-player game and equilibrium solutions are discussed.

Theorem 1 *When, for every class* c, $1 \leq c \leq K$, *resources are not fully utilized, there is a pure strategy Nash equilibrium in game which corresponds to candidate's policy of provider as* $PS = 1$, *i.e., SaaS provider admits the request and* $CS = 2$, *i.e., user decides to stay with SaaS provider.*

Proof Since resources are not fully utilized for all the class c, churning rate $CR_c \approx 0$ where $1 \leq c \leq K$. In other words, $U_c \approx 1$. Payoff value of provider as given in Eq. 11 will have maximum value when provider admits the request and user stays with the provider. It is because, in this case, value of $earRev_k$ will be positive (since provider accept the request), value of $\sum_{c=1}^{K} N_c CR_c losRev_c$ will be zero (since $CR_c \approx 0$) and value of $losRev_k$ will also be zero (user stays with provider).

Since, $U_c \approx 1$, user's payoff matrix $CP = [cp_{lm}]_{2\times 2}$ is reduced as in Eq. 29.

$$CP = \begin{bmatrix} cp_{11} = w1q + w1W_a - w2churnPen & cp_{12} = w1q + w1W_a \\ cp_{21} = w1W_b - w2churnPen & cp_{22} = w1W_b \end{bmatrix} \tag{29}$$

In this matrix, cp_{12} is greater than all other elements of matrix $(cp_{11}, cp_{21}, cp_{22})$, so user j selects the strategy to stay with the current provider, i.e., $CS = 2$, while provider admits the request.

From the above discussion, it is found that candidate's policy of provider as $PS = 1$ and $CS = 2$ constitutes Nash equilibrium for the game. Remaining three possible combinations of strategy do not give Nash equilibrium. □

Theorem 2 *When resources are fully utilized or overutilized for at least one class* c *where* $1 \leq c \leq K$, *SaaS provider accepts the request if* $earRev_k \geq \sum_{c=1}^{K} N_c CR_c losRev_c$ *and rejects otherwise.*

Proof From Eq. 11, the following can be observed.

When $earRev_k \geq \sum_{c=1}^{K} N_c CR_c losRev_c$, $pp_{11} \geq pp_{21}$ and $pp_{12} \geq pp_{22}$. In this case, dominant strategy of provider is to accept the request which yields better payoff than its other strategy no matter customer stays with provider or not.

When $earRev_k < \sum_{c=1}^{K} N_c CR_c losRev_c$, $pp_{21} \geq pp_{11}$ and $pp_{22} \geq pp_{12}$. In this case, dominant strategy of provider is to reject the request which yields better payoff than its other strategy no matter customer stays with provider or not.

In summary, in the online cases, if resources are not fully utilized in any class just accept the incoming request. But if the resources of at least one class is fully utilized or overutilized, then accept incoming request belonging to class k if revenue generated by accepting this request, i.e., $earRev_k$ is greater than total possible revenue loss by accepting this request $\sum_{c=1}^{K} N_c CR_c losRev_c$. Otherwise reject the request. □

5 The proposed resource control technique

As far as resource allocation is concerned, the users should get the amount of resources as per their budget. In this work, users are assumed belonging to different classes based on their budget availability since admission control does not impose any restriction on how much resources can be assigned to a particular class. Resource control algorithm is based on a concept that Cloud assigns resources to its customers in a way that allows least churning by dynamic allocation of resources to different class. At the same time, it provides freedom to the provider to assign resources to customers according to their priority (class). If a customer is not satisfied with the services, it may churn if dissatisfaction prevails. Since the proposed resource control technique classifies the customers on the basis of their budget, provider may give preference to customers with high budget. This work assumes that provider is more focused on long-term benefit in comparison to a short-term gain. The goal of resource control algorithm is to assign more resources to its customers who belong to higher class but not to ignore low-budget customers totally.

The proposed resource control algorithm dynamically assigns resources to customers of different class using a tunable parameter which helps providers to give priority to their customers. Let us assume that currently there are N_i active requests in the system of class i. So, total work load on system by the customers of class i can be represented by Eq. 30,

$$workload_i = \sum_{j=1}^{N_i} numMI_j^i \tag{30}$$

where $numMI_j^i$ is length of request j of class i in MI.

When new requests are admitted in the system, workload on system by class i will be given as in Eq. 31.

$$workload_i^{new} = \sum_{j=1}^{N_i + N_i^{new}} numMI_j^i \tag{31}$$

where N_i^{new} is the number of new admitted requests of class i. If resources will be fixed for a particular class and number of admitted requests is more for that class, it may increase the churn rate of that class. The aim of the provider is not only to

reduce the churn rate but also to increase its revenue. When admission control component accepts new requests, resource control component is initiated which dynamically assigns resources to a particular class based on its priority. Since provider wants to increase its revenue, it will assign more resources to high-paying requests (i.e., higher priority requests). So, provider must have some tunable parameter to assign more or less resources according to its wish. Let αt_i be the tunable parameter for class i,then new tuned workload according to provider is as shown in Eq. 32.

$$workload_i^t = \alpha t_i \times workload_i^{new} \tag{32}$$

Resources are assigned according to their tuned workload, i.e., if provider has R resources, then assignment of resources to class i will be given by Eq. 33.

$$assignR_i = \left(workload_i^t \Big/ \sum_i^K workload_i^t \right) \times R \tag{33}$$

6 Experimental study

There are two basic approaches to study the performance of a system; analytical approach and experimental approach [34]. In analytical approach, mathematics is used as a tool to study the system, while sometimes complexity of system makes analytical approach insufficient. For complex system, experimental approach is good for observations. In experimental approach, experiments are performed to study the system (sometimes by simulating the model). Study of a system by both the approaches validates the system. For strategic decision making, game theory is a better alternative. In the proposed model, game theory is used to decide the best strategy in case of admission of upcoming requests. Different cases for admission control are derived and it is proved that there is existence of best strategy in every case. To validate the proposed model by experiments, a Cloud computing environment is required. Evaluation of performance of different policies related to provisioning, workload and resource scenarios in a real Cloud environment in a repeatable and controllable manner under different configuration and requirements of system and user is a difficult task as building a Cloud environment on a real infrastructure is tedious, time-consuming and expensive [35].

Simulation tools are a better alternative for the evaluation of a model in controllable environment without paying any cost. In addition, reproduction of results is also easy. Currently, Cloud specific simulators such as NetworkCloudSim [36], GreenCloud [37], CloudSim [35] are useful. NetworkCloudSim (an extension of CloudSim) is suitable for message passing parallel applications, GreenCloud is focused on Cloud networking and energy awareness and CloudSim is a most advanced simulator to establish and simulate a heterogeneous distributed computing environment for Cloud computing on a physical node. Some novel features of CloudSim are listed below [35].

- A self-contained platform for modeling datacenters, service brokers, scheduling, and allocations policies.

- Availability of a virtualization engine, which aids in the creation and management of multiple, independent, and cohosted virtualized services on a datacenter node.
- Flexibility to switch between space-shared and time-shared allocation of processing cores to virtualized services.

Therefore, to evaluate the performance of the proposed admission control algorithm, CloudSim platform [35] is used. Experiments are performed on Intel(R) Core(TM) i7-3770 CPU @ 3.40 GHz with 2.0 GB RAM.

Though using game theory, analytically it is proved that solutions exist for both $n + 1$ players game and two-player game. A simulation environment is created in CloudSim to test only the proposed algorithm for $n + 1$ players game because it is more complex in comparison to two-player game which is a simplified form of $n + 1$ player game. Two providers, Provider 1 and Provider 2, are considered. Provider 1 uses the proposed framework with conventional techniques of CloudSim and Provider 2 uses only conventional techniques of CloudSim. Although scheduling algorithm can change the performance of the accepted requests, the focus of this work is admission control. Therefore, default scheduling algorithm of CloudSim is used as scheduling algorithm for both the providers. Performance is observed for both SaaS provider and users. Profit of provider is considered for SaaS provider's point of view and average response time is considered for users' point of view.

In the system when n requests arrive, admission control takes decision whether to admit a request or not. Once a request is admitted, resource control component allocates resources to different class of users on the basis of current load and provider's requirement (provider's tunable parameter discussed in Sect. 5). In the experiment, to clarify the working of resource control component in a simple manner, it is assumed that cloud resources are homogeneous virtual machines. It has already been discussed in Sect. 4.2 that there is K class of customers. For experimental purpose, requests are classified into three classes on the basis of their budget estimate for their execution.

It has been observed [38,39] that arrival rate of requests follows Poisson distribution. This work also assumes λ_1, λ_2, λ_3 as the arrival rate of requests of class 1, class 2 and class 3, respectively. Each request is defined by the four tuple ⟨Request Length, Budget, Deadline, Penalty Rate⟩. Following is the brief description of each parameter and its considered value, corresponding to a request in the experiment, keeping the fact that class is also an important factor of request.

Request length Each request has some length in MI (Million Instruction). Since in CloudSim, request is represented by number of million instruction, range is generated in MI. For the experiment, service rate values of VMs and size of request are taken in such a way that the dynamic behavior and effectiveness of the proposed model can be depicted. Length of the requests is generated between 25,000 MI to 125,000 MI for the experimental purposes.

Budget Budget is the money a user is willing to pay to a SaaS provider for the execution of its request. Budget is generated in the range of \$0.1 to \$1.0 as the variation in results is very less after \$1.0. A request belongs to class 3 if its budget is in between \$0.1 to \$0.3, request belongs to class 2 if its budget is in between \$0.3 to \$0.6 and request belongs to class 1 if its budget is in between \$0.6 to \$1.0. Because requests are classified on the basis of budget, class 1 is the highest class.

Deadline This is the time limit to complete a request as defined by the user. Turnaround time of each request must be less than the deadline, otherwise SaaS provider needs to pay some penalty. The deadline of a request is generated in between 1.5 times and 3.5 times of the processing time of the request. Since class 1 requests are paying higher, their deadline should be strict. The deadline of a request of class 1 is generated between 1.5 × processing time of request and 2 × processing time of request. Similarly, deadline request of class 2 and class3 are generated between 2 × processing time of request and 3.5 × processing time of request.

Penalty rate If a request is not completed in the defined time (i.e., deadline), provider pays some penalty to the user. The penalty rate of a request i of class c is defined as given in Eq. 34.

$$Penalty\ Rate_i = Budget_i/(Deadline_i \times C_c) \tag{34}$$

where C_c is a constant to define strictness of penalty rate. If value of C_c is high, penalty rate is low. Since requests of class1 are paying more in comparison to the requests of class 2 and class 3, penalty rate of class 1 should be high, i.e., value of C_1 should be low. For experimental purpose, four set of values considered for each class are (C_1, C_2, C_3) = {(4, 6, 10), (8, 12, 20), (12, 18, 30), (16, 24, 40)}.

Utility of SaaS provider is the profit obtained for each accepted request in the system. Profit from a particular request can be defined as given in Eq. 35.

$$Profit_i = Budget_i - Cost_i \tag{35}$$

where

$$Cost_i = ProcessingCost_i + PenaltyCost_i \tag{36}$$

$$ProcessingCost_i = ProcessingTime_i \times PriceofVM \tag{37}$$

$$PenaltyCost_i = ProcessingDelay_i \times PenaltyRate_i \tag{38}$$

$$ProcessingDelay_i = (FinishTime_i - SubmissionTime_i) - Deadline_i \tag{39}$$

To define the price of virtual machine ($PriceofVM$), different Cloud resource providers such as Amazon EC2 [40], Microsoft Azure [41], RackSpace [42], and GoGrid [43] are observed and it is found that cost of virtual machines varies from \$0.05 to \$1.32 per hour. For the experiment, \$0.12 is considered as the cost of each virtual machine.

Some other parameters for the simulation study of the proposed model are defined as below.

Average earned revenue of provider by class i ($earRev_i$) can be obtained using the budget parameter. Since range of budget for each class is fixed, average revenue taken as earned revenue for each class is $earRev_1 = \$0.75$, $earRev_2 = \$0.45$ and $earRev_3 = \$0.20$.

If a customer churns, SaaS provider loses some revenue. This loss can be estimated from the cost which provider incurred to acquire a customer. This cost includes

advertisement, promotion, etc. [44]. In other words, this loss can be formulated as customer acquisition cost per customer. So average revenue loss of provider of class ($losRev_i$) can be considered as customer acquisition cost of customer of class i. Various experiments are performed with different values of loss revenue to decide an appropriate value of loss revenue. It is found that with loss revenue value taken as $losRev_1 = \$\ 300$, $losRev_2 = \$\ 250$ and $losRev_3 = \$\ 200$, the model shows more variation in results. Thus, the same values of loss revenue is considered for the various experiments performed in this work.

In resource control, defined in Sect. 5, value of tunable parameter of αt_i for each *class i* is considered such that $\alpha t_1 > \alpha t_2 > \alpha t_3$. It is because customers of class 1 are paying more money in comparison to other classes and so entitled for more resources.

For simplicity, in the experiment of Sects. 6.1–6.4, it is considered that arrival rate $\lambda = \lambda_1 + \lambda_2 + \lambda_3$ and $\lambda_1 = \lambda_2 = \lambda_3$. λ_1, λ_2 and λ_3 vary from 10 to 100, i.e., λ varies from 30 to 300.

6.1 Observation on profit, average response time, decision time and acceptance of requests

In this experiment, 50 virtual machines for each provider are created. Value of αt_1, αt_2 and αt_3 are fixed to 4, 6 and 10, respectively, i.e., $(\alpha t_1, \alpha t_2, \alpha t_3) = (4, 6, 10)$. Figure 3 depicts the comparison of profit of provider 1 and provider 2 with different arrival rate. From Fig. 3, it is observed that when the system initiates, both providers get the same profit (up to $\lambda = 60$ provider 1 and provider 2 both get same profit). As the arrival rates increases, provider 1 is getting more profit in comparison to provider 2 because virtual machines are busy. In fact, provider 2 is not getting any profit at all. Provider 1 is getting profit up to $\lambda = 240$, but thereafter it also does not get any profit as all 50 virtual machines are saturated. This fact is also observed in experiment of Sect. 6.2 with varying number of virtual machines.

Figure 4 represents the average response time of provider 1 and provider 2 for the same scenario as discussed above. The graph shows that average response time of provider 1 is very low in comparison to provider 2. The reason for this result is

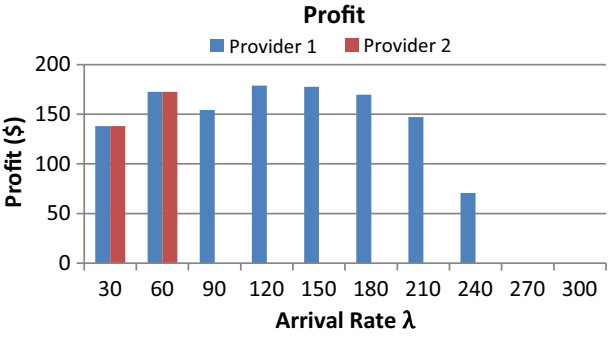

Fig. 3 Profit observations of providers on the arrival rate

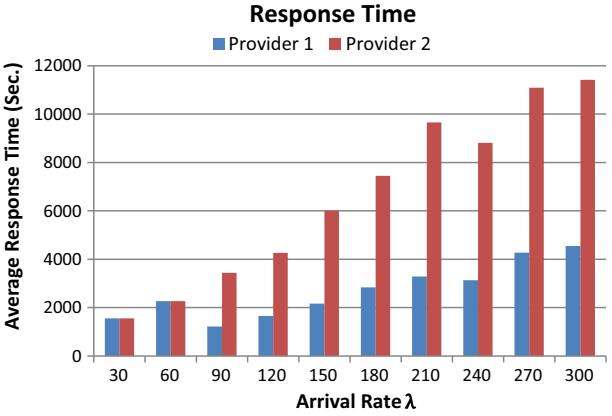

Fig. 4 Average response time of providers on different arrival rate

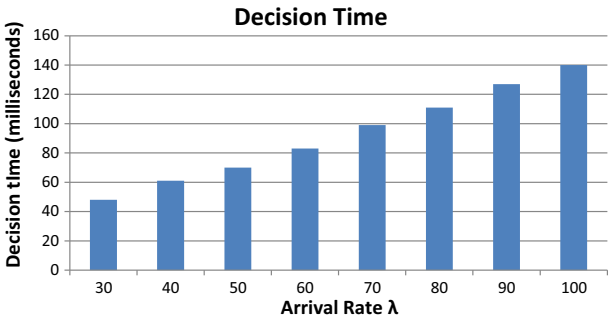

Fig. 5 Decision time on different arrival rates

that provider 1 is using the proposed model and admitting only those requests whose requirements can be fulfilled along-with some profit.

Figure 5 represents the decision time of provider 1 which is the average time A/R block takes to decide which request to be accepted and rejected. Decision time is represented in milliseconds. The graph shows that A/R block takes very less time to decide on the requests. Though, on increase of arrival rate of request, decision time increases, but not substantially.

Total number of accepted requests by provider 1 and provider 2 are shown in Fig. 6. Since provider 2 does not reject any request, total number of accepted request by provider 2 is total number of requests generated in the simulation period. From Fig. 6, it can be observed that provider 1 is rejecting requests as it has limited resources. The model also suggests that if the provider is not able to fulfill QoS of a request it should be rejected otherwise SLA violation will impose a penalty on the provider leading to reduction in the provider's profit. The results from Figs. 3 and 4 show a significant difference between the profit of both the providers and significant difference between customers' satisfactions of both the providers encourage for the implementation of the proposed model in a real Cloud.

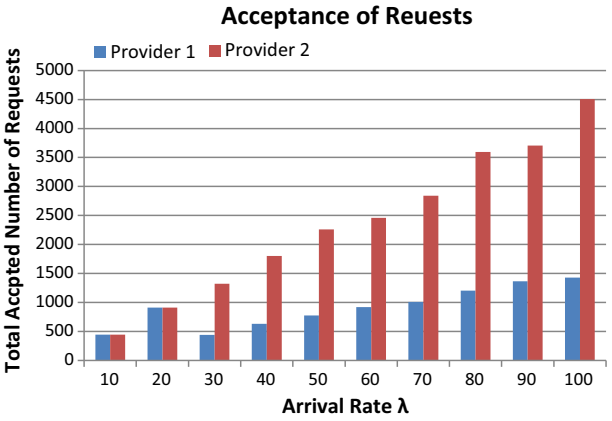

Fig. 6 Acceptance of requests on different arrival rate

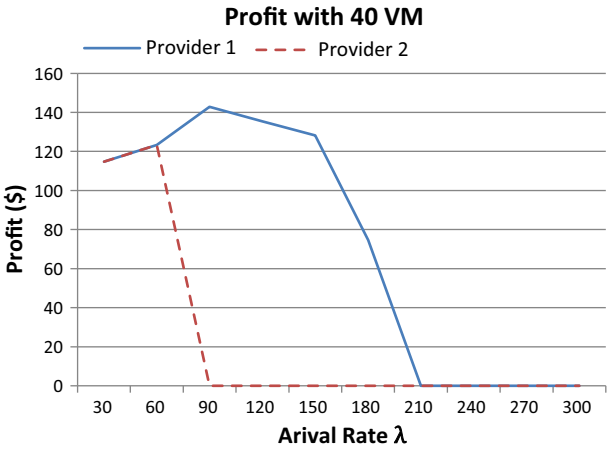

Fig. 7 Profit of providers on 40 virtual machines

6.2 Profit with variant number of VM

In this experiment, all the parameters are kept same as in Sect. 6.1 and profit of provider 1 and provider 2 is observed with different number of virtual machines. In Fig. 7, with VM = 40 and $\lambda = 70$, provider 1 stops getting profit whereas in Fig. 10, with VM = 70 the profit of provider 1 gets nil only after $\lambda > 100$. The zero profit point shifts towards the increasing λ side on increasing the number of VMs from 40 to 70 (Figs. 7, 8, 9, 10). Thus, if a provider has more number of virtual machines, it is likely to get more profit.

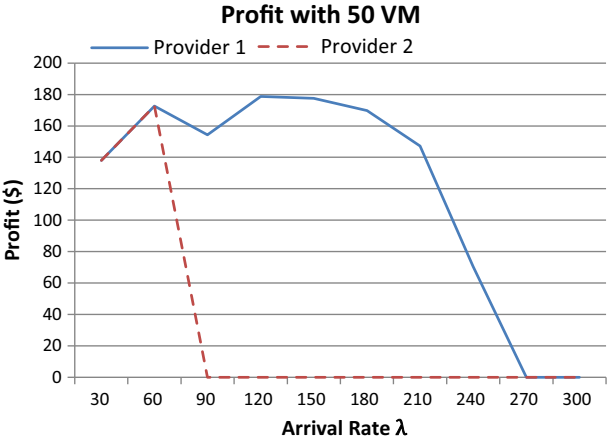

Fig. 8 Profit of providers on 50 virtual machines

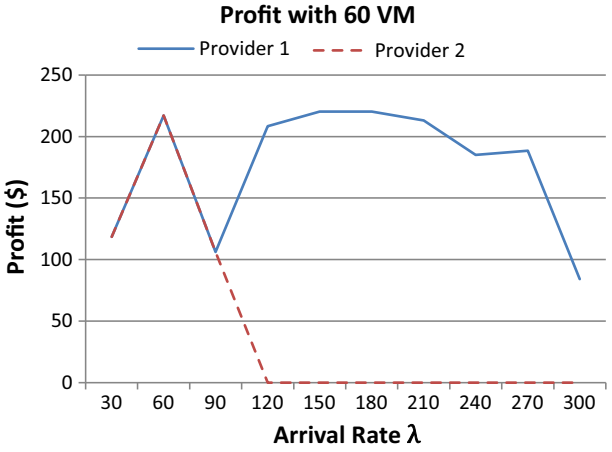

Fig. 9 Profit of providers on 60 virtual machines

6.3 Impact of penalty rate

In this experiment, profit of provider 1 is observed at different penalty rates, i.e., $(C_1, C_2, C_3) = \{(4, 6, 10), (8, 12, 20), (12, 18, 30), (16, 24, 40)\}$ for 50 virtual machines. From Fig. 11, it is observed that as the value C_i of class i increases profit of provider also increases.

6.4 Impact of tunable parameter

In this experiment, value of tunable parameter αt_i of class i is observed. For different set of αt_1, αt_2 and αt_3, profit of provider 1 is observed. From Fig. 12, it is found that provider gets less profit when αt_2 is high in comparison to αt_1 or when αt_1 is very

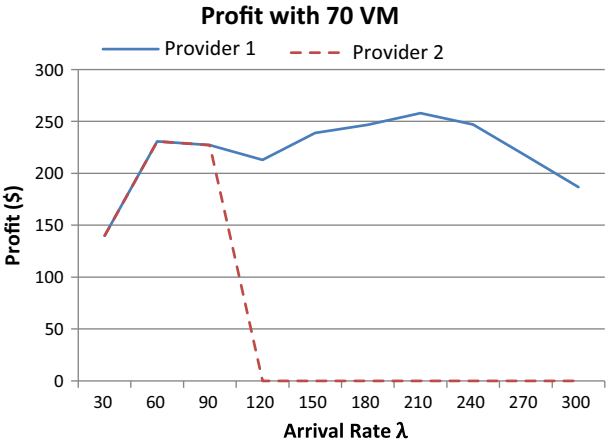

Fig. 10 Profit of providers on 70 virtual machines

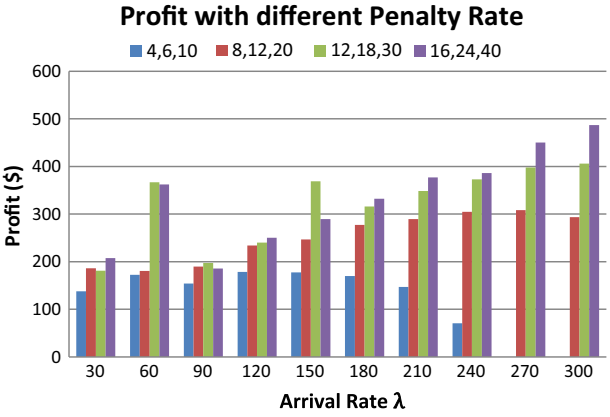

Fig. 11 Profit of providers on different penalty rate

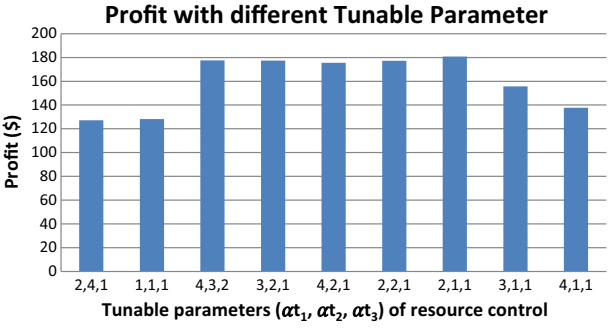

Fig. 12 Profit of providers on different tunable parameter

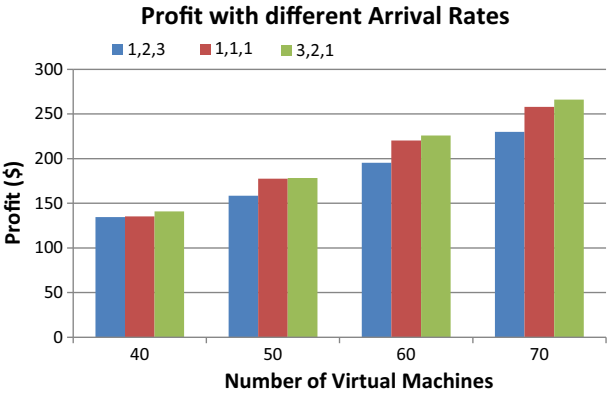

Fig. 13 Profit of providers vs. virtual machine with different arrival rate

high in comparison to αt_2 and αt_3 or when αt_1, αt_2 and αt_3 are equal. So, there should be a reasonable difference between αt_1, αt_2 and αt_3.

6.5 Impact of mix arrival rate

From the experiment in Sect. 6.2, it can be noticed that provider gets a significant profit at $\lambda = 120$ when number of virtual machines are 40, at $\lambda = 150$ when number of virtual machines are 50, at $\lambda = 180$ when number of virtual machines 60 and at $\lambda = 210$ when number of virtual machines are 70. So, in this experiment, impact of mix arrival rate is observed only for those λ at which provider gets significant profit to understand the observation in simple manner. For VM = 40, λ is taken as 120 and λ_1, λ_2 and λ_3 are taken in the ratio of 1:1:1, 1:2:3 and 3:2:1, respectively, to observe the profit of the provider. Same is done for (VM = 50, $\lambda = 150$), (VM = 60, $\lambda = 180$) and (VM = 70, $\lambda = 210$). It is observed from Fig. 13 that when arrival rate of class 2 and class 3 requests is more in comparison to arrival rate of class 1 requests, profit of provider is less and as the arrival rate of requests of class 1 increases, profit of provider also increases.

7 Conclusions and future work

A novel framework for admission control in Cloud computing is proposed in this work using game theoretic approach. Admission control problem is formulated as $n + 1$ player non-cooperative and non-zero sum game between SaaS provider and n upcoming requests and then this game is simplified as two-player game. It has been shown that there is a dominant strategy and Nash equilibrium in pure strategy. To the best of our knowledge, for the formulation of payoff of provider, churning rate, i.e., probability of customer leaving the provider is considered for the first time with revenue of provider in Cloud computing. Definitely, if requests of higher classes are paying more money, they deserve more resources. So resource control framework is integrated

with admission control to allocate the resources to the different class of users as per the provider's choice. The proposed model is simulated on CloudSim simulator and it has been shown than if a provider uses the proposed admission and resource control framework, it can achieve higher revenue and at the same time average response time of requests decreases for the customer. Different scenarios are considered to show that the proposed framework works well. Resource control component and formulation of churning rate can easily be replaced or enhanced by the providers according to their requirement.

In the work, it is assumed that SLA database is static, i.e., if a customer needs some change in its SLA, customer has to first leave the provider and re-join the provider as a fresh customer with new SLA. This work has the possibility for extension by considering the dynamic SLA database, in which customer will be able to change the SLA without leaving the provider. The proposed framework can be made more robust by adding SLA negotiation and dynamic pricing strategy.

References

1. Garg SK, Versteeg S, Buyya R (2013) A framework for ranking of cloud computing services. Fut Gen Comput Syst 29:1012–1023
2. Li A, Yang X, Kandula S, Zhang M (2010) CloudCmp: comparing public cloud providers. In: Proceedings of the 10th annual conference on internet measurement, Melbourne, pp 1–14
3. Baranwal G, Vidyarthi DP (2014) A framework for selection of best cloud service provider using ranked voting method. In: Proceedings of IEEE international advance computing conference (IACC), pp 831–837
4. Pueschel T, Putzke F, Neumann D (2012) Revenue management for cloud providers-a policy-based approach under stochastic demand. In: Proceedings of 45th Hawaii international conference on system science, pp 1583–1592
5. Wu L, Garg SK, Buyya R (2012) SLA-based admission control for a software-as-a-service provider in cloud computing environments. J Comput Syst Sci 78(5):1280–1299
6. Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I (2009) Cloud computing emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Fut Gen Comput Syst 25(6):599–616
7. Zhang Q, Zhu Q, Boutaba R (2011) Dynamic resource allocation for spot markets in cloud computing environments. In: Proceedings of 4th IEEE international conference on utility and cloud computing (UCC), pp 178-185
8. Buyya R, Garg SK, Calheiros RN (2011) SLA-oriented resource provisioning for cloud computing: challenges, architecture, and solutions. In: Proceedings of international conference on cloud and service computing (CSC), pp 1–10
9. Mao M, Li J, Humphrey M (2010) Cloud auto-scaling with deadline and budget constraints. In: Proceedings of 11th IEEE/ACM international conference on grid computing (GRID), pp 41–48
10. Moakar LA, Chrysanthis PK, Chung C, Guirguis S, Labrinidis A, Neophytou P, Pruhs K (2010) Admission control mechanisms for continuous queries in the cloud. In: Proceedings of IEEE 26th international conference on data engineering (ICDE), pp 409–412
11. Feldman Z, Masin M, Tantawi AN, Arroyo D, Steinder M (2011) Using approximate dynamic programming to optimize admission control in cloud computing environment. In: Proceedings of the 2011 winter simulation conference, 2011, pp 3153–3164
12. Konstanteli K, Varvarigou T, Cucinotta T (2011) Probabilistic admission control for elastic cloud computing. In: Proceedings of IEEE International conference on service-oriented computing and applications, pp 1–4
13. Calheiros RN, Ranjan R, Buyya R (2011) Virtual machine provisioning based on analytical performance and QoS in cloud computing environments. In: Proceedings of IEEE international conference on parallel processing (ICPP), pp 295–304
14. Yeo CS, Venugopal S, Chu X, Buyya R (2010) Autonomic metered pricing for a utility computing service. Fut Gen Comput Syst 26(8):1368–1380

15. Jaideep DN, Varma MV (2010) Learning based Opportunistic admission control algorithms for map reduce as a service. In: Proceedings of the 3rd India software engineering conference (ISEC 2010), Mysore
16. Rouskas AN, Kikilis AA, Ratsiatos SS (2006) Admission control and pricing in competitive wireless networks based on non-cooperative game theory. In: Proceedings of IEEE wireless communications and networking conference, vol 1, pp 205–210
17. Lin H, Chatterjee M, Das SK, Basu K (2005) ARC: an integrated admission and rate control framework for competitive wireless CDMA data networks using noncooperative games. IEEE Trans Mob Comput 4(3):243–258
18. Ye D, Chen J (2013) Non-cooperative games on multidimensional resource allocation. Fut Gen Comput Syst 29(6):1345–1352
19. Pal R, Hui P (2013) Economic models for cloud service markets: pricing and capacity planning. Theor Comput Sci 496:113–124
20. Ardagna D, Panicucci B, Passacantando M (2011) A game theoretic formulation of the service provisioning problem in cloud systems. In: Proceedings of the 20th international conference on World wide web, pp 177–186
21. Wei G, Vasilakos AV, Zheng Y, Xiong N (2010) A game-theoretic method of fair resource allocation for cloud computing services. J Supercomput 54(2):252–269
22. Kim N, Jung Kyu-Hwan, Kim YS, Lee J (2012) Uniformly subsampled ensemble (USE) for churn management: theory and implementation. Expert Syst Appl 39:11839–11845
23. Xie Y, Li X, Ngai EWT, Ying W (2009) Customer churn prediction using improved balanced random forests. Expert Syst Appl 36:5445–5449
24. Tsai Chih-Fong, Lu Yu-Hsin (2009) Customer churn prediction by hybrid neural networks. Expert Syst Appl 36:12547–12553
25. Coussement K, Benoit DF, den Poel DV (2010) Improved marketing decision making in a customer churn prediction context using generalized additive models. Expert Syst Appl 37:2132–2143
26. Gibbs MN, MacKay DJC (2000) Variational Gaussian process classifiers. IEEE Trans Neural Netw 11(6):1458–1464
27. Bennani MN, Menasce DA (2005) Resource allocation for autonomic data centers using analytic performance models. In: Proceedings of the 2nd international conference on autonomic computing, pp 229–240
28. Ai X, Marinescu DC, Boloni L, Siegel HJ, Daley RA, Wang I-Jeng (2008) A macroeconomic model for resource allocation in large-scale distributed systems. J Parallel Distrib Comput 68(2):182–199
29. Paton NW, de Aragao MAT, Lee K, Fernandes AAA, Sakellariou R (2009) Optimizing utility in cloud computing through autonomic workload execution. IEEE Data Eng Bull 32:51–58
30. Roy N, Das SK, Basu K, Kumar M (2005) Enhancing availability of grid computational services to ubiquitous computing applications. In: Proceedings of 19th IEEE international parallel and distributed processing symposium, p 92
31. Shoham Y, Leyton-Brown K (2009) Multiagent systems algorithmic, game-theoretic, and logical foundations, Cambridge University Press
32. Basar T, Olsder GT (1999) Dynamic noncooperative game theory, 2nd edn. Soc. Industrial and Applied Math., Philadelphia
33. Gen M, Cheng R (2000) Genetic algorithms and engineering optimization. Wiley, New York
34. Gustedt J, Jeannot E, Quinson M (2009) Experimental validation in large-scale systems: a survey of methodologies. Parallel Process Lett 19(3):399–418
35. Calheiros RN, Ranjan R, Beloglazov A, De Rose CAF, Buyya R (2011) CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw Pract Exp 1(41):23–50
36. Garg SK, Buyya R (2013) An environment for modeling and simulation of message-passing parallel applications for cloud computing. Softw Pract Exp 43(11):1359–1375
37. Kliazovich D, Bouvry P, Khan SU (2012) GreenCloud: a packet-level simulator of energy-aware cloud computing data centers. J Supercomput 62(3):1263–1283
38. McManus ML, Long MC, Copper A, Litavak E (2004) Queuing theory accurately models the need for critical care resources. Anesthesiology 100(5):1271–1276
39. Wolff RW (1998) Poisson arrivals see time averages. Oper Res 30(2):223–231
40. Amazon EC2 (2014). http://aws.amazon.com/
41. Microsoft Azure (2014). http://azure.microsoft.com/en-us/

42. RackSpace (2014). http://www.rackspace.com/
43. GoGrid (2014). http://www.gogrid.com/
44. Winer RS (2001) A framework for customer relationship management. Calif Manag Rev 43(4):89–105
45. Valerio VD, Cardellini V, Presti FL (2013) Optimal pricing and service provisioning strategies in cloud systems: a Stackelberg game approach. In: Proceedings of 6th international conference on cloud computing, pp 115–122
46. Anselmi J, Ardagna D, Passacantando M (2014) Generalized nash equilibria for saas/paas clouds. Eur J Oper Res 236(1):326–339
47. Hassan MM, Hossain MS, Sarkar AMJ, Huh EN (2014) Cooperative game-based distributed resource allocation in horizontal dynamic cloud federation platform. Inf Syst Front 16(4):523–542
48. Jebalia M, Letaïfa AB, Hamdi M, Tabbane S (2013) A comparative study on game theoretic approaches for resource allocation in cloud computing architectures. In: Proceedings of 22nd International workshop on in enabling technologies: infrastructure for collaborative enterprises (WETICE), pp 336–341