A Comparison of Game-Theoretical Pricing and Provisioning Strategies in Cloud Systems

Valeria Cardellini
Department of Civil Engineering and Computer Science Engineering
University of Rome "Tor Vergata", Rome, Italy
cardellini@ing.uniroma2.it

Valerio Di Valerio
Department of Computer Science
Sapienza University of Rome, Italy
divalerio@di.uniroma1.it

Francesco Lo Presti
Department of Civil Engineering and Computer Science Engineering
University of Rome "Tor Vergata", Rome, Italy
lopresti@info.uniroma2.it

Università di Roma "Tor Vergata"
Dipartimento di Ingegneria Civile e Ingegneria Informatica
Technical Report RR-14.4

May 26, 2014

Abstract

We consider several Software as a Service (SaaS) providers that offer applications using the Cloud facilities provided by an Infrastructure as a Service (IaaS) provider which adopts a *pay-per-use* scheme similar to the Amazon EC2 service, comprising flat, on demand, and spot virtual machine instances. For this scenario, we study the virtual machine provisioning and spot pricing strategies. We consider a two stage provisioning scheme. In the first stage, the SaaS providers determine the optimal number of required flat and on demand instances. Then, in the second stage, the IaaS provider sells its unused capacity as spot instances for which the SaaS providers compete by submitting a bid. We study two different IaaS provider pricing strategies: the first assumes the IaaS provider sets a unique price; in the second, instead, the IaaS provider can set different prices for different users. We model the resulting problem as a Stackelberg game. For each pricing scheme, we provide proof of the existence of the game equilibrium and provide the solution algorithms. Through numerical evaluation we compare the provisioning and spot price under the two different pricing strategies as function of the system parameters.

1 Introduction

The burgeoning of Cloud-based services with their promise of efficiency, elasticity and cost-effectiveness, is before our eyes and the analysts predict that in the next years Cloud services will growth at an even fast rate[1]. Computing resources offered by Infrastructure as a Service (IaaS) providers are now used by industrial and academic organizations to run their applications, that can elastically scale up or down as demand changes by provisioning virtual resources almost instantaneously. In their turn, customers of IaaS providers can rapidly offer their innovative applications, thus becoming Software as a Service (SaaS) providers, but without the need to own and maintain development or production infrastructures.

IaaS providers offer their computing resources in the form of Virtual Machine (VM) instances to customers, that generally rent the *on demand* resources as needed on a pay-per-use basis, paying a fixed price for a short time (usually one hour). When resource utilization can be planned in advance, IaaS customers can also reserve *flat* resources, paying a long-term reservation fee plus a per-hour price which depends on the effective resource usage and is lower than the on demand price. In order to achieve high utilization in data centers that are often under-utilized, IaaS providers can also sell their spare capacity in form of *spot* instances by organizing an auction where customers bid, providing a maximum per-hour price they are willing to pay. On the basis of the bids and its spare capacity, the IaaS provider sets the spot instances price. For IaaS customers spot instances represent an attractive and cost-effective solution to deal with unexpected load spikes and run compute-intensive applications, but at the risk of a lower reliability than flat and on demand instances, since the IaaS provider can revoke spot VM instances without notice due to price and demand fluctuations. Amazon's EC2 cloud [3], for example, offers the three types of VM instances and pricing models (i.e., flat, on demand, and spot instances) just described.

From the IaaS perspective, selling resources to multiple customers requires to determine efficient service provisioning and pricing strategies, in order to better utilize unused capacity and generate more revenue, still satisfying the customers expectation. On the other hand, each SaaS provider is interested in maximizing its profit, behaves selfishly and competes with the other SaaS providers to acquire the IaaS resources. Developing such strategies in the Cloud environment is a challenging task, mostly because the Cloud is inherently competitive and dynamic. In this context, game theoretic approaches can provide a formal framework to capture the behavior of IaaS and SaaS providers in a conflicting situation and thus devise strategies to effectively address the resource provisioning and pricing issue. Many works in the Cloud computing area (e.g., [4, 5, 15, 29, 32, 23, 22]) have already exploited game-theoretic tools, because typical problems that also arise in the Cloud, such as quality of service, resource allocation, and pricing, cannot be handled with classical optimization approaches since each player can be affected by the actions of all players, not only by its own actions. In most works, the solution concept of Nash equilibrium has been extensively applied (a set of players' strategies is a Nash equilibrium when no player can improve its revenue by changing unilaterally its strategy).

In this paper, we consider a Cloud scenario where a IaaS provider sells its computing resources to several SaaS providers, offering flat, on demand, and spot VM instances. In their turn, SaaS providers offer to their users Web services with Quality of Service (QoS) guarantees, using the IaaS facilities to host and run the provided services. Revenues and penalties of each SaaS provider depend on the provisioning of an adequate performance level, which is specified in a Service Level Agreement (SLA) contract that each SaaS stipulates with its users. Therefore, each SaaS provider has to face the problem of determining the optimal number of VMs to satisfy the SLA while

[1] `http://www.idc.com/prodserv/FourPillars/Cloud/index.jsp`

maximizing its revenue. However, a proper solution cannot be accomplished in isolation, since the SaaS providers compete among them and bid to acquire the spot instances. On the other hand, the IaaS provider aims at maximizing its revenue and therefore wants to properly choose the price of the spot instances.

To model this conflicting situation we recur to a Stackelberg game [13]. In this class of games, one player, the leader, in our case the IaaS provider, moves first and commits its strategy to the remaining players, the followers, for us the SaaS providers, that consider the action chosen by the leader before acting simultaneously to choose their own strategy in a selfish way through a standard Nash game. Stackelberg games are commonly used to model attacker-defender scenarios, e.g. [18]. For the considered Cloud scenario, the adoption of a leader-follower strategy sounds feasible, since we can reasonably assume that the IaaS provider fixes the price before the SaaS providers compete to acquire the VM resources. Furthermore, a Stackelberg game helps us to devise a revenue-maximizing pricing scheme for the IaaS provider.

The salient contributions of this work are as follows.

1. We devise a *two-stage* service provisioning and pricing strategy. In the *first stage*, each SaaS provider independently determines the optimal number of flat and on demand VM instances (which have a fixed price) so to guarantee the performance level agreed in the SLAwhile maximizing its profit. In the *second stage*, the SaaS providers bid and compete for the unused IaaS provider capacity. The goal of each SaaS provider is to determine the number of spot instances to allocate which maximizes its profit, given the number of flat and on demand instances bought in the first stage. The goal of the IaaS provider is to determine the price of the spot instances in order to maximize its profit.

2. We consider two different pricing models for the IaaS provider, namely, Same Spot Price Model (SSPM) and Multiple Spot Prices Model (MSPM). The former is characterized by a single price which applies to all customers, while in the latter strategy the IaaS provider can conveniently set different spot instance prices for different users.

3. While the first stage of the proposed two-stage strategy involves the solution of standard optimization problems, the second stage requires to compute the equilibria of the SaaS/IaaS Stackelberg game on spot instances. Since Stackelberg games are challenging problems, we have to resort to two different techniques to compute the game equilibrium for the two pricing models. We address the solution of the Stackelberg game arising from the MSPM by solving a suitable Mathematical Program with Equilibrium Constraint (MPEC) problem. The same methodology does not apply to the SSPM.

4. As a further contribution, we devise an ad hoc algorithm for the SSPM that computes the game equilibrium exploiting the game structure.

5. We study through numerical investigation the behavior of the proposed provisioning and pricing strategies under different workload and bidding configurations and compare it with the service provisioning and pricing policy proposed in [4]. Using our strategies, the IaaS provider can set a price of the spot instances lower than the maximum price in the bids in order to incentivize SaaS providers to buy more instances, thus increasing its revenue with a higher volume of sold instances.

This paper integrates and extends the Stackelberg formulation for the MSPM we proposed in [7], adding the contributions 2) and 4) listed above and presenting a mostly new numerical evaluation of the proposed strategies.

The remaining of the paper is organized as follows: Section 2 introduces our system model. In Section 3, we define the two-stage service provisioning and pricing scheme. We discuss the solution method of the SSPM and MSPM strategies that arise from our problem formulation in Section 4. In Section 5 we analyze through numerical experiments the behavior of the proposed SSPM and MSPM strategies; we also compare their results to those achieved by the formulation in [4]. In Section 6 we discuss related works. Finally, we conclude the paper and present directions for future work in Section 7.

2 System Model

We consider a set $\mathcal{U}$ of SaaS providers that offer Web services/applications using the cloud facilities provided by a IaaS provider. We assume that each SaaS provider $u \in \mathcal{U}$ offers a single Web service characterized by a SLA, which stipulates the service QoS levels, *i.e.*, service response time and the associated cost/penalty for its use[2].

Web services are hosted on virtual machines instantiated by the IaaS provider. For the sake of simplicity, we assume that the IaaS provider offers only one type of VMs, *i.e.*, all the VMs have the same RAM and CPU capacity[3]. Each Web service can be distributed on multiple VMs and in that case we assume the workload to be evenly split among them.

The IaaS provider manages an infrastructure which can provide to its users up to $\mathcal{S}$ VMs which are offered to users as flat, on demand, and/or spot instances. Flat instances are characterized by one-time payment plus a payment of φ unit per hour of actual use. On demand instances have no one-time payment and are charged at a price δ, which we assume to be strictly larger than φ. Spot instances are charged at a price σ which is set dynamically and depends on the users bids and competition for the unused resources and the IaaS provider optimal pricing strategy. In this paper, we also consider a more general spot pricing model, in which the spot price σ_u varies from SaaS provider to SaaS provider, as considered for example in [4, 5, 7]. In the rest of this paper, we refer to the former model as **Same Spot Price Model (SSPM)** and to the latter as **Multiple Spot Prices Model (MSPM)**.

In the general case of MSPM, given the spot prices σ_u and the number of flat f_u, on demand d_u, and spot instances s_u allocated to the Web service of each SaaS provider u, the associated per-hour IaaS revenue can be defines as:

$$\Theta_I = \sum_{u \in \mathcal{U}} \varphi f_u + \delta d_u + \sigma_u s_u \tag{1}$$

Each SaaS provider determines for its service the number of flat f_u, on demand d_u, and spot s_u VMs to be allocated which maximizes its revenue. We assume the revenue/utility function of each SaaS provider to take the form $\Theta_u(f_u, d_u, s_u, \sigma_u)$, where Θ_u represents the revenue/utility arising from the Web service. In this paper, we assume that neither Θ_u has a specific expression nor it is the same for each SaaS provider. We just require Θ_u to be strictly concave and twice differentiable with respect to s_u, with $\partial\Theta_u/\partial^2 s_u < 0$, to reflect the law of diminishing marginal utility [20].

For ease of reference, we summarize in Table 1 the main notation adopted in this paper.

[2]We focus on a single service for ease of explanation, but our model can be easily generalized to consider also SaaS providers offering multiple services.

[3]This is not a limiting assumption because Amazon, for example, runs independent spot markets for each instance type [36]

System parameters	
$\mathcal{S}$	Total amount of VMs managed by IaaS provider
$\mathcal{U}$	Set of SaaS providers
$\hat{f}_u$	Number of reserved flat instances for SaaS provider u
Λ_u	Predicted next hour arrival rate for SaaS provider u
μ_u	Maximum service rate of VM executing SaaS provider u service
U_u^{max}	Maximum allowed utilization of VM executing SaaS provider u service
φ	Time unit cost for one *flat* VM
δ	Time unit cost for *on demand* VM
σ^L	Minimum time unit cost for *spot* VM, set by IaaS provider
σ_u^U	Maximum time unit cost for *spot* VM SaaS provider u is willing to pay

Decision variables	
f_u	Number of *flat* VMs acquired by SaaS provider u
d_u	Number of *on demand* VMs acquired by SaaS provider u
s_u	Number of *spot* VMs acquired by SaaS provider u
σ	Time unit cost for *spot* VM (SSPM)
σ_u	Time unit cost for *spot* VM of SaaS provider u (MSPM)

Table 1: Main notation

3 Service Pricing and Provisioning: a Stackelberg Game Approach

We assume that SaaS providers periodically allocate and deallocate the VMs used for offering its service according to a prediction of the workload expected in the next time slot. In this paper we consider a *two-stage* pricing and provisioning scheme. In the first stage, each SaaS provider determines the number of flat and on demand instances[4] which guarantees the performance level defined in the SLA agreed with its prospective users and maximizes its revenue. We assume that the IaaS provider has enough resources to always satisfy the request for on demand instances generated by its customers. Then, in the second stage, the IaaS provider sells to the SaaS providers its unused capacity as spot instances. Differently from the first stage, each SaaS provider competes with the others for these additional resources by submitting to the IaaS provider a bid, which defines the maximum per VM price it is willing to pay. The IaaS provider, given the resource availability and the submitted bids, determines the spot instances price so to maximize its revenue. For the second stage, in this paper we study two different pricing strategies, named SSPM and MSPM. In the first strategy, the IaaS provider sets a single price which applies to all users; users whose submitted bid is lower than the actual price will not get any instance; in the latter strategy, the IaaS provider can conveniently set different spot instance prices for different users.

3.1 First Stage: Flat and on Demand VMs Provisioning

In the first stage, each SaaS provider determines independently from the others the optimal number of flat and on demand VMs that allows to maximize its revenue, while sustaining the predicted

[4]Observe that, in case of flat instances, this number represents the number of *already* allocated flat instances which will be used to implement the offered services (a SaaS provider does not pay the per unit of time cost for unused flat instances).

arrival rate Λ_u and satisfying the SLA agreed with its users.

For each SaaS provider $u \in \mathcal{U}$ we have the following optimization problem [5]:

$$\textbf{max } \Theta_u \tag{2}$$

$$\textbf{subject to: } f_u \leq \hat{f}_u \tag{3}$$

$$\frac{\Lambda_u}{\mu_u(f_u + d_u)} \leq U_u^{max} \tag{4}$$

$$f_u, d_u \geq 0 \tag{5}$$

Constraint (3) ensures that the actual number of flat instances allocated to SaaS provider u is less than or equal to the number of reserved flat instances $\hat{f}_u$. Constraint (4) guarantees that the resources utilization is less than a threshold U_u^{max} to avoid resource over-utilization and ensure adequate performance. For the sake of simplicity, as in [4, 5, 7], we do not impose that the decision variables for the numbers of instances are integers. Nevertheless, our findings apply to the actual problem as well.

3.2 Second Stage: Spot VMs Pricing

In the second stage, the SaaS providers compete for the unused IaaS provider resources made available via a bidding mechanism, in which each SaaS provider u specifies σ_u^U, the maximum time unit cost per spot VM it is willing to pay. The rationale is that the SaaS providers can increase their revenues by using additional resources, while the IaaS provider can make profit from the otherwise unsold resources. Without lack of generality, we assume that the IaaS provider sets a minimum reserve bid σ^L to account for the operative cost to run a VM.

The goal of each SaaS provider is to determine the number of spot instances in addition to the flat and on demand instances already provisioned in the first stage so to maximize its revenue. As expected, the formulation of the optimization problems of both the SaaS providers and the IaaS provider depends strictly on the adopted pricing model, i.e., SSPM or MSPM. We analyze them in the following.

3.2.1 Same Spot Price Model (SSPM)

In the SSPM, the IaaS provider sets the same spot price σ for all the users.

SaaS problem Each SaaS provider u determines the optimal number of spot instances to acquire s_u by solving the following optimization problem:

$$\textbf{Problem SSPM\_SaaS}_{OPT}$$

$$\textbf{max } \Theta_u(\bar{f}_u, \bar{d}_u, s_u, \sigma)$$

$$\textbf{subject to: } \sum_{u \in \mathcal{U}} s_u \leq s^U, \quad s_u \geq 0 \tag{6}$$

$$y_u \in \{0, 1\} \tag{7}$$

$$s_u \leq s^U y_u \tag{8}$$

$$(\sigma_u^U - \sigma) \leq M y_u \leq (\sigma_u^U - \sigma) + M \tag{9}$$

[5] Recall that we assume that the IaaS provider has always enough resources to accommodate all flat and on demand instances requests, otherwise we would have competition also at this stage.

where $\bar{f}_u$ and $\bar{d}_u$ represent respectively the number of flat and on demand instances already allocated and $s^U = \mathcal{S} - \sum_{u \in \mathcal{U}} (\bar{f}_u + \bar{d}_u)$ the amount of unused IaaS capacity, being $\mathcal{S}$ the total amount of VMs the IaaS provider manages and $\sum_{u \in \mathcal{U}} (\bar{f}_u + \bar{d}_u)$ the amount of VMs acquired by all the SaaS providers in the first stage. Constraints (8)-(9), where M is a large constant, impose that SaaS provider u will not get any spot VM if its bid σ_u^U is lower than the spot price σ. To this end, we introduce the integer variable y_u equal to 1 if the bid of SaaS provider u is greater than the actual spot price, 0 otherwise. Constraint (6) guarantees that the total number of spot VMs sold to the SaaS providers is less than or equal to that available at the IaaS provider. Note that differently from the first stage problem, we now have a constraint which involves the decision variables of all the SaaS providers and an utility function which depends on the spot price σ, the IaaS decision variable.

IaaS problem The IaaS provider goal is to determine the spot price σ in order to maximize its revenue. The IaaS provider optimization problem is:

$$\begin{aligned} &\textbf{Problem SSPM\_IaaS}_{OPT} \\ &\textbf{max } \Theta_I = \textbf{max} \sum_{u \in \mathcal{U}} s_u \sigma \\ &\textbf{subject to: } \sigma^L \leq \sigma \end{aligned} \tag{10}$$

where σ^L is the minimum spot price set by the IaaS provider.

3.2.2 Multiple Spot Price Model (MSPM)

In the MSPM, the IaaS provider can conveniently set different prices for different users. The resulting SaaS providers and IaaS provider problems are as follows.

SaaS problem Each SaaS provider determines the optimal number of spot instances to acquire s_u by means of the following optimization problem:

$$\begin{aligned} &\textbf{Problem MSPM\_SaaS}_{OPT} \\ &\textbf{max } \Theta_u(\bar{f}_u, \bar{d}_u, s_u, \sigma_u) \\ &\textbf{subject to: } \sum_{u \in \mathcal{U}} s_u \leq s^U, \quad s_u \geq 0 \end{aligned} \tag{11}$$

where $\bar{f}_u$, $\bar{d}_u$, and s^U are defined as in Section 3.2.1. As in the SSPM, constraint (11) ensures that the total number of spot VMs acquired by the SaaS providers is less than or equal to the spare capacity of the IaaS provider and it involves the decision variables of all the SaaS providers. We also observe that the utility function depends on the multiple spot prices σ_u, which are the IaaS decision variables.

IaaS problem The goal of the IaaS provider is to determine the spot price σ_u for each SaaS provider u in order to maximize its revenue. The IaaS provider optimization problem is:

$$\begin{aligned} &\textbf{Problem MSPM\_IaaS}_{OPT} \\ &\textbf{max } \Theta_I = \textbf{max} \sum_{u \in \mathcal{U}} s_u \sigma_u \\ &\textbf{subject to: } \sigma^L \leq \sigma_u \leq \sigma_u^U, \quad \forall u \in \mathcal{U} \end{aligned} \tag{12}$$

where σ_u^U is the maximum spot price each SaaS provider u is willing to pay and σ^L is the minimum spot price imposed by the IaaS provider.

3.2.3 Second Stage as Stackelberg Game

In the second stage, independently from the considered pricing model, the decisions of the SaaS providers and the IaaS provider depend mutually from each other. Indeed, the objective function of the IaaS provider depends on s_u, the decision variables of the SaaS providers, while the objective function of each SaaS provider depends on the spot price(s), which is(are) the decision variable(s) of the IaaS provider. Moreover, the decision of each SaaS provider depends also on what the other providers do, since constraints (6) and (11) couple the variables of all the SaaS providers.

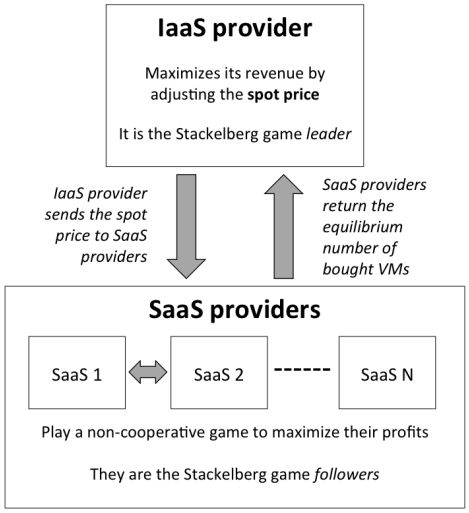


Figure 1: Interaction between SaaS providers and IaaS provider in our pricing and provisioning strategy

We model such a conflicting situation as a Stackelberg game [13]. Stackelberg games are a particular type of non-cooperative game whereby one player (the leader) takes its decision before the other players (the followers). Given the leader decision, the followers then simultaneously take their own decision. Assuming a rational behavior, the leader can take advantage of the fact that the followers basically react to its decisions, which leads to a follower *subgame* equilibrium (if any exists), and drives the system to its own optimum (possibly a global one). Figure 1 illustrates the second stage of our service pricing and provisioning scheme. In our model, the IaaS provider acts as a leader by deciding the spot price(s) $\sigma(\sigma_u)$. The SaaS providers act as followers which must decide the number s_u of spot instances to buy. *Given* the spot prices, the SaaS providers thus compete among themselves (the SaaS providers subgame) for the shared pool of available instanced s^U. Following the Nash equilibrium concept, given the spot prices, the SaaS providers should adopt a strategy such that none of them can improve its revenue by unilaterally changing its strategy. In turn, the IaaS provider can determine the sets of prices which maximizes its revenue and which represents the Stackelberg equilibrium of the game: this equilibrium is characterized by the property that for the given set of prices, the SaaS providers have adopted a strategy such that none of them could improve its revenue by changing it unilaterally, and the IaaS provider would not benefit by modifying the chosen set of prices.

4 Solution Method

Our pricing and provisioning strategies comprise two stages. The first stage involves the solution of a set of independent convex optimization problems which can be solved by means of standard techniques. The second stage requires the computation of the *equilibria* of the SaaS/IaaS spot instance Stackelberg game. This is a challenging problem for which no general solution exists. Indeed, we have to recur to two different methodologies to solve the same and multiple price spot models. In this section, we first address the solution of the SaaS providers subgame. We show how the MSPM SaaS subgame can be regarded as a generalization of the SSPM SaaS subgame and we demonstrate the existence of at least one subgame Nash equilibrium. Then, we demonstrate the existence of the Stackelberg equilibrium and how to compute it in the case of the MSPM and SSPM.

Throughout the rest of this section, we denote by s_u the strategy of a SaaS provider $u \in \mathcal{U}$. Furthermore, we indicate with $s = (s_u)_{u=1}^N$ the set of strategies of all the SaaS providers and with s^{-u} the set of the strategies of all the SaaS providers except the SaaS provider u, where $N = |\mathcal{U}|$. We define with K_u the feasible strategies set for SaaS provider u and we further define $\Omega = \left\{ s \,\middle|\, \sum_{u \in \mathcal{U}} s_u \leq s^U \right\}$. Hence, we can define the set $K = (K_1 \times K_2 \dots K_N) \cap \Omega$ of the feasible strategies of all the players. Note that the set K is compact, since each variable s_u is bounded by s^U. For convenience, we also denote with $g(s) = (g_u(s))_{u=1}^N$ the vector function that represents the compact set K.

4.1 SaaS Providers Subgame

We first consider the followers subgame, i.e., the SaaS providers competition that arises once the IaaS provider fixes its strategy (the spot price(s)). In particular, we study the SaaS subgame that arises from the MSPM, because it is a more general case than the one that arises from the SSPM. Indeed, let us consider the SSPM SaaS subgame and, without loss of generality, only those SaaS providers whose bids are greater or equal than the spot price (for each SaaS provider u with $\sigma_u^U < \sigma$ the only feasible strategy is to play $s_u = 0$, regardless the others do). Once σ is fixed, the problem $\mathbf{SSPM\_SaaS}_{OPT}$ in Section 3.2.1 reduces to the following optimization problem:

$$\begin{aligned} &\textbf{Problem Redux\_SSPMSaaS}_{OPT} \\ &\textbf{max } \Theta_u(\bar{f}_u, \bar{d}_u, s_u, \sigma) \\ &\textbf{subject to: } \sum_{u \in \mathcal{U}} s_u \leq s^U, \quad s_u \geq 0 \end{aligned} \tag{13}$$

As we can see, this optimization problem is a particular case of $\mathbf{MSPM\_SaaS}_{OPT}$ in which $\sigma_u = \sigma,\ \forall u \in U$.

Since the SaaS providers act simultaneously, the SaaS subgame can be modeled as a Generalized Nash game [10]. Generalized Nash Equilibrium Problems (GNEPs) differ from Nash Equilibrium Problems (NEPs) in that, while in NEP only the players objective functions depend on the other players strategies, in GNEP both the objective functions and the strategy sets depend on the other players strategies. In our problem, the dependence of each player strategy set on the other players strategies is represented by constraint (13), which includes all SaaS providers decision variables σ_u. More specifically, our problem is a *Jointly Convex* GNEP [10]. This property follows from the fact that the objective function of each SaaS provider is concave on its own decision variable, the strategy space is convex, and the constraint involving all players variables is the same for all the players.

Jointly Convex GNEPs are a particular class of GNEP whose solution can be computed by solving a proper *variational inequality* (VI)[6]. In particular, under the condition that the objective function of each player is continuously differentiable, every solution of the $VI(K, F(s;\sigma))$, where $F(s;\sigma) = -(\nabla_{s_1}\Theta_1(s;\sigma), \nabla_{s_2}\Theta_2(s;\sigma), \ldots, \nabla_{s_N}\Theta_N(s;\sigma))'$, is also an equilibrium of the GNEP [10]. Such equilibrium is known as *variational equilibrium*. In general, a GNEP has multiple or even infinite equilibria, and not all of them are also a solution of the VI. However, the variational equilibrium is more "socially stable" than the other equilibrium of a GNEP and therefore it represents a valuable target for an algorithm [10].

We now establish two key properties of the $VI(K, F(s;\sigma))$, namely that the function F is strongly monotone[7] and the existence of the generalized Nash equilibrium for the followers subgame.

Theorem 1. *Function* $F(s;\sigma) = -(\nabla_{s_1}\Theta_1(s;\sigma), \ldots, \nabla_{s_N}\Theta_N(s;\sigma))'$ *is strongly monotone.*

Proof. The function F and its Jacobian take the following forms:

$$F(s;\sigma) = \begin{bmatrix} -\dfrac{\partial\Theta_1}{\partial s_1} \\ \cdots \\ -\dfrac{\partial\Theta_N}{\partial s_N} \end{bmatrix}$$

and its Jacobian is:

$$JF(s;\sigma) = \begin{bmatrix} a_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & a_N \end{bmatrix}$$

JF is a diagonal matrix whose generic entry $a_u = -\partial\Theta_u/\partial^2 s_u$ is, by definition, strictly positive. The Weierstrass theorem ensures that $\hat{a}_u = \min_{s_u\in[0,\ldots,s^U]} a_u$ exists in the closed interval $[0, \ldots, s^U]$ for each a_u. Hence, if we choose a constant $0 < \alpha < \min_{k\in\mathcal{A}_u, u\in\mathcal{U}} \hat{a}_u$ then the matrix $JF(s;\sigma) - \alpha I$ is still diagonal with each term strictly greater than 0, for all $s \in K$. Hence the function $F(s;\sigma)$ is strongly monotone. □

Theorem 2. *There exists exactly one variational equilibrium of the followers subgame.*

Proof. The existence of at least one generalized Nash equilibrium of the followers subgame is a direct consequence of the strong monotonicity of the function $F(s;\sigma)$ [11]. □

4.2 Stackelberg Game

We now turn our attention to the IaaS (leader) problem of determining the optimal pricing strategy. As a first step, we demonstrate the existence of the Stackelberg equilibrium of the game (the proof considers the general case of the MSPM).

Theorem 3. *There exists at least one Stackelberg equilibrium of the game.*

[6]Given a subset K of $\Re^n$ and a function $F : K \to \Re^n$, the VI problem, denoted by $VI(K, F)$, consists in finding a point $s^* \in K$ such that $(s - s^*)^T F(x^*) \geq 0 \quad \forall s \in F$.

[7]F is strongly monotone on K if there exists a constant $c > 0$ such that for all pairs $s, y \in K$

$$(s - y)^T (F(s) - F(y)) > c\| s - y\|^2$$

holds.

Proof. The proof is straightforward. Let us define $\Sigma = \{\boldsymbol{\sigma} = (\sigma_u)^{u\in\mathcal{U}} \mid \sigma_u \in [\sigma^L,\ \sigma_u^U]\}$ as the set of all IaaS provider feasible strategies. For each $\boldsymbol{\sigma} \in \Sigma$, Theorem (2) ensures the existence of the variational equilibrium. Because Σ is a compact set, there exists at least one $\boldsymbol{\sigma}^* \in \Sigma$ so that $\Theta_I(\boldsymbol{\sigma}^*, s^*) \geq \Theta_I(\boldsymbol{\sigma}, s),\ \forall \boldsymbol{\sigma} \in \Sigma$, where s^* and s are the follower subgame equilibria parametrized by $\boldsymbol{\sigma}^*$ and $\boldsymbol{\sigma}$, respectively. This completes the proof, because the couple $(\boldsymbol{\sigma}^*,\ s^*)$ is, *by definition*, a Stackelberg equilibrium of the game. □

Now, we show how to compute the game equilibrium for both the MSPM and the SSPM.

4.2.1 MSPM Game Equilibrium

In the previous section, we have shown that the problem of finding an equilibrium of the SaaS providers subgame for a given IaaS strategy can be reformulated as the problem of finding the solution of a proper VI. This allows us to solve compute the Stackelberg equilibria for the MSPM by solving a suitable *Mathematical Programs with Equilibrium Constraint* (MPEC) [9]. An MPEC is an optimization problem whose constraints include variational inequalities. In particular, the MPEC arising from our IaaS optimization problem takes the following form:

$$\textbf{Problem IaaS}_{MPEC}$$

$$\textbf{max} \sum_{u\in\mathcal{U}} \bar{f}_u\varphi + \bar{d}_u\delta + s_u\sigma_u$$

$$\textbf{subject to: } \sigma^L \leq \sigma_u \leq \sigma_u^U, \quad \forall u \tag{14}$$

$$s \in SOL(\sigma) \tag{15}$$

where $SOL(\sigma)$ is the solution of the $VI(K, F(s;\sigma))$. Observe that this is just the IaaS problem $\textbf{IaaS}_{OPT}$ with the additional constraint that the SaaS providers strategy s is a solution of the SaaS subgame.

Because of (15) we cannot solve the MPEC directly. We thus follow the approach proposed in [9] that, under the assumption that function $F(s;\sigma)$ is strongly monotone, allows us to compute stationary points of the MPEC. Hence, only necessary conditions for optimality are satisfied.

As a first step, we replace (15) with its Karush Kuhn Tucker (KKT) conditions [19]. We obtain the following non linear programming problem:

$$\textbf{max} \sum_{u\in\mathcal{U}} \bar{f}_u\varphi + \bar{d}_u\delta + s_u\sigma_u$$

$$\textbf{subject to: } \sigma^L \leq \sigma_u \leq \sigma_u^U, \quad \forall u \tag{16}$$

$$F(\sigma) - \nabla_s g(s)\lambda = 0 \tag{17}$$

$$g(s) \leq 0 \tag{18}$$

$$\lambda^T g(s) = 0, \quad \lambda \geq 0 \tag{19}$$

where $\lambda \in \Re^l$ is the vector of Lagrangian multipliers, with l the number of constraints that define K. Such problem cannot be directly solved because the constraints do not satisfy any standard constraint qualification and the complementary-type constraints (19) are very complicated and difficult to handle. Again, following the general framework presented in [9], we consider a sequence of smooth and regular problems, obtained by perturbing the original problem, the solutions of which converge to a solution of the original problem, i.e., to a stationary point of the MPEC. Specifically,

we consider the perturbed problem $\mathbf{P}(\mu)$ with parameter μ:

$$\textbf{Problem } \mathbf{P}(\mu)$$

$$\textbf{max} \sum_{u \in \mathcal{U}} \bar{f}_u \varphi + \bar{d}_u \delta + s_u \sigma_u$$

$$\textbf{subject to: } \sigma^L \leq \sigma_u \leq \sigma_u^U, \forall u \tag{20}$$

$$F(\sigma) - \nabla_s g(s)\lambda = 0 \tag{21}$$

$$g(s) + z = 0 \tag{22}$$

$$\sqrt{(z-\lambda)^2 + 4\mu^2} - (z + \lambda) = 0 \tag{23}$$

where $z \in \Re^l$ is an auxiliary variable. In $\mathbf{P}(\mu)$, constraint (22) replaces constraint (18), while constraint (23) replaces (19). It is easy to realize that $\mathbf{P}(\mu)$ corresponds to the original problem when $\mu = 0$[8]. We refer the reader to [9] for further details.

Problem $\mathbf{P}(\mu)$, $\mu \neq 0$, is a smooth regular problem which can be solved using standard optimization tools. Let $\sigma^*(\mu)$ denote a solution of $\mathbf{P}(\mu)$. From [9], we have that $\sigma^*(\mu)$ converges to a stationary point of the $\mathbf{IaaS}_{MPEC}$ as $\mu \to 0$. To compute a solution we use Algorithm S presented in [9] (see Algorithm 1), which solves a sequence of problems $\mathbf{P}(\mu)$. The algorithm stops when the Euclidean distance between two successive iterations is lower than a suitable threshold ϵ. We verified that in practice the algorithm converges very quickly. In the experiments described in the next section, the algorithm converged in no more than 3 iterations using $\epsilon = 10^{-4}$.

Algorithm 1 Algorithm S [9]

Let $\{\mu^k\}$ be any sequence of nonzero numbers with $\lim_{k\to\infty} \mu^k = 0$.
Choose $w^0 = (\sigma^0, x^0, z^0, \lambda^0) \in \Re^{3N+l+l}$, and set $k = 1$.
while $||e|| > \epsilon$ **do**
  Find a stationary point w^k of $P(\mu^k)$
  $e = w^k - w^{k-1}$
  $k = k + 1$
end while

4.2.2 SSPM Game Equilibrium

We have just seen that to compute the Stackelberg equilibrium of the MSPM, we resort to an MPEC. Unfortunately, we cannot use the same approach to solve the SSPM, since $\mathbf{SSPM\_SaaS}_{OPT}$ is a Mixed Integer Linear Programming (MILP) problem; even if we could rewrite it as a convex problem, the resulting MPEC problem cannot be solved, since the resulting objective function would be discontinuous. Discontinuity arises in correspondence of spot prices equal to the SaaS provider bids. This is because when the price exceeds the bid, the corresponding SaaS provider gets no spot VM at all, which results in a jump in the IaaS provider revenue. Moreover, after each discontinuity point, the revenue function can either increase or decrease depending on the actual SaaS providers revenue functions.

As a consequence, in order to compute the Stackelberg equilibrium we need to resort to the brute force approach we sketch below:

[8]Observe that for $\mu = 0$, if $s \in SOL(\sigma)$ is a solution of the original problem then either $g(s) < 0$ and $\lambda = 0$, in which case the constraint (23) reduces to $\sqrt{(z)^2} - z = 0$ or $g(s) = 0$, which implies that $z = 0$, and (23) reduces to $\sqrt{(-\lambda)^2} - \lambda = 0$.

1. discretize the interval of feasible spot prices $[\sigma^L, \max_u \sigma_u^U]$;
2. solve the SaaS providers subgame for every sample;
3. pick up the price σ^* that corresponds to the greatest IaaS provider revenue.

The Stackelberg equilibrium will be, *by definition*, the couple (σ^*, s^*).

In principle, the SaaS subgame solution can be computed by solving the associated variational inequalities $VI(K, F(s;\sigma))$. However, in this case, we can take advantage of the subgame structure: once the price σ is fixed, the revenue/utility of each SaaS provider u depends only on its own strategy s_u. Hence, the followers subgame is actually a potential game [21], which is amenable to a simple solution. Indeed, we can compute the subgame equilibrium using the algorithm proposed in [5] (see Algorithm 2), that allows for a fully distributed implementation.

Algorithm 2 Followers subgame Nash Equilibrium

Require: spot price σ
1: **for** $u \in \mathcal{U}$ **do**
2: compute s_u solving **Redux_SSPMSaaS**$_{OPT}$ replacing constraint 13 with $s_u \leq s^U$
3: **end for**
4: **if** $\sum_{u\in\mathcal{U}} s_u \leq s^U$ **then**
5: STOP
6: **else**
7: $s_u^U = \dfrac{s_u}{\sum_{u\in\mathcal{U}} s_u} s^U, \ \forall u \in \mathcal{U}$
8: **end if**
9: **for** $u \in \mathcal{U}$ **do**
10: compute s_u as the solution of **Redux_SSPMSaaS**$_{OPT}$ replacing (13) with $s_u \leq s_u^U$
11: **end for**

Algorithm 2 consists of three steps. Again, we consider only those SaaS providers whose bids are greater or equal than the spot price σ, because for the others the only feasible strategy is to play $s_u = 0$. In the first step (line 1 to 3), each SaaS provider solves problem **Redux_SSPMSaaS**$_{OPT}$ replacing constraint (13) with $s_u \leq s^U$, i.e., the variables of other SaaSs are discarded. This is a strictly concave optimization problem defined on a compact set, hence it has exactly one solution. If the solution s satisfies constraint (13), then it is a Nash equilibrium. Otherwise, in the second step (line 4 to 8) the spot VMs are shared proportionally among the SaaS providers according to the relaxed solutions. In other words, we set an upper bound $s_u^U = \dfrac{s_u}{\sum_{u\in\mathcal{U}} s_u} s^U$ for the number of spot VMs each SaaS provider u can buy. At the last step (line 9 to 11), each SaaS provider solves its optimization problem with the new individual bound s_u^U and a Nash equilibrium is then found.

Finally, note that the discretization of the price interval allows us to relax some assumptions on the SaaS provider utility function Θ_u. Instead of requiring Θ_u to be twice differentiable with respect to s_u with $\partial\Theta_u/\partial^2 s_u < 0$, we can simply require Θ_u to be strictly concave with respect to s_u. This is an important result, because it enlarges the set of functions that can be adopted. Note that requiring the function to be strictly concave simply means that for each spot price the optimal amount of VMs is unique, so it is a very reasonable assumption.

Theorem 4. *If Θ_u is strictly concave with respect to s_u, $\forall u \in \mathcal{U}$, then Algorithm 2 finds a Nash equilibrium of the followers subgame.*

Proof. We prove it by contradiction. Let's define $\hat{s}$ as the amount of spot VMs requested after step 1 and s as the final solution computed by Algorithm 2. Suppose s is not an equilibrium. If s is not

an equilibrium, there exists at least a SaaS u and a strategy $\bar{s}_u$ so that $\bar{\Theta}_u(\bar{s}_u, \sigma) > \Theta_u(s_u, \sigma)$. We identify two cases.

Case one: $\hat{s} = s$, i.e., the algorithm ends after step 1 because the amount of spot VMs requested is less than s^U. In this case, $\hat{s}_u$ is computed as the solution of **ReduxSaaS**$_{OPT}$ replacing constraint 13 with $s_u \leq s^U$. *Case two*: $\hat{s} \neq s$. In this case, s is computed as the solution of **ReduxSaaS**$_{OPT}$ replacing constraint 13 with $s_u \leq s_u^U$.

In both cases, if $\bar{\Theta}_u(\bar{s}_u, \sigma) > \Theta_u(s_u, \sigma)$, then we have multiple local maximum values on the compact set defined by the constraints, but this is impossible because Θ_u is strictly concave. So, $\hat{s}$ must be a subgame equilibrium. □

Computational Complexity The last step of Algorithm 2 can be executed only in correspondence of the spot price σ^*. Indeed, to compute σ^* how the spot VMs are spread among the individual SaaS providers is irrelevant: only the global amount of sold VMs matters. As a results, if we discretize the interval of feasible spot prices $[\sigma^L, \max_u \sigma_u^U]$ in p samples, computing the Stackelberg equilibrium requires to solve $(p+1)N$ strictly convex optimization problems. Hence, the complexity of Algorithm 2 is polynomial. We can further reduce such complexity if we can solve **Redux_SSPMSaaS**$_{OPT}$ relying on a closed formula expression, as we are going to see in Section 5.

5 Experimental Results

In this section we investigate through numerical experiments the behavior of the proposed provisioning and pricing strategies. We start in Section 5.1 with a comparison between the SSPM and the MSPM strategies: we compute the system equilibria in different scenarios, and study how flat, on demand, and spot VMs are allocated among the competing SaaS providers and the associated spot prices under different workload and bidding configurations. Then, in Section 5.2 we compare our strategies with the service provisioning and pricing policy studied in [4].

For the sake of comparison, in our experiments we consider the SaaS providers utility function proposed in that work. The authors of [4] assume that, for each SaaS provider, the SLA takes the form of an upper bound on the service response time R_u^{max}. The SLA also specifies the user per-request cost $\mathcal{C}_u = C_u(1 - \frac{R_u}{R_u^{max}})$, which it is assumed to be a linear function of the service response time R_u. We note that $\mathcal{C}_u$ is a decreasing function of the response time and becomes negative (hence, the SaaS provider incurs into a penalty) when $R_u > R_u^{max}$. This model allows to consider a soft constraint on the response time, which enables the SaaS provider to trade-off revenues and infrastructural costs, and to model the SaaS providers gain to let them buy more VMs than the bare essential so to satisfy the SLA, increase the service reliability and performance and, in turn, their reputation.

Each Web service hosted on a VM is modeled as an M/G/1/PS queue with an application dependent service rate μ_u. Assuming a perfect load sharing among multiple VMs assigned to the service, the service average response time is given by:

$$E[R_u] = \frac{f_u + d_u + s_u}{\mu_u(f_u + d_u + s_u) - \Lambda_u}$$

provided the stability condition $\frac{\Lambda_u}{\mu_u(f_u+d_u+s_u)} < 1$ holds. Taking into account the infrastructural per-hour cost for the provisioned VMs (where $\sigma_u = \sigma$ in case of SSPM), the per-hour SaaS profit

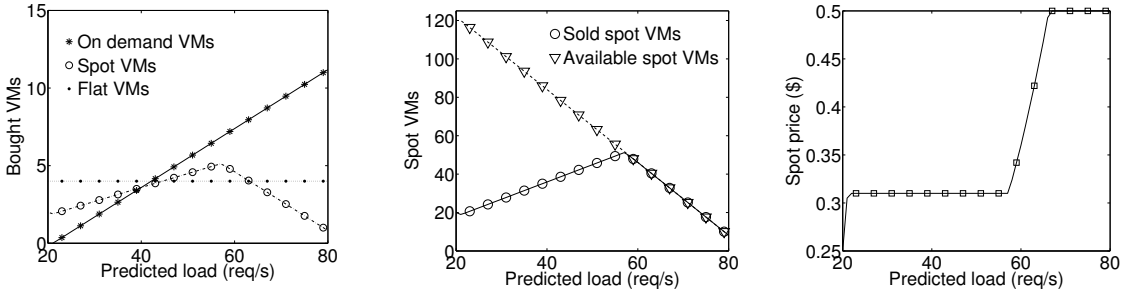
is:

$$\Theta_u = \Lambda_u C_u \left(1 - \frac{1}{R_u^{max}} \frac{f_u + d_u + s_u}{\mu_u(f_u + d_u + s_u) - \Lambda_u}\right) - \varphi f_u - \delta d_u - \sigma_u s_u \tag{24}$$

where the first term is the sum of the average per service revenues $\Lambda_u \mathcal{C}_u = \Lambda_u C_u (1 - \frac{E[R_u]}{R_u^{max}})$ and the remaining terms the VMs costs.

This utility function satisfies the assumptions in Section 2 (we omit the proof for space reason). It also allows each SaaS provider to compute *analytically* the solution of **Redux_SSPMSaaS**$_{OPT}$ by solving $\nabla_{s_u} \Theta_u(s;\sigma) = 0$. Hence, this function allows to significantly decrease the computational complexity of the SSPM Stackelberg equilibrium, because no optimization problem has to be solved.

In the following experiments, we consider one IaaS provider which sells its resources to 10 SaaS providers. If not differently stated, we set $\mathcal{S} = 160$, $\varphi = 0.24\$$, $\delta = 1.24\$$, $\hat{f}_u = 4$, $\mu_u = 10$ req/s, $U_u^{max} = 0.9$, $\sigma^L = 0.1\$$, $\sigma_u^U = 0.5\$$, $C_u = 1\$$, and $R_u^{max} = 1$ s for all $u \in \{1, \ldots, 10\}$. Such parameters setting corresponds to that adopted in [4] for the sake of comparison. For the analysis,



(a) Number and type of VMs bought by each SaaS provider

(b) Number of spot VMs available and sold by the IaaS provider

(c) Optimal spot price

Figure 2: Behavior of SSPM and MSPM strategies in the homogeneous scenario

we implemented in MATLAB the algorithms presented in Section 4 as well as those in [4]. For the solution of the MPEC problem via Algorithm 1, the parameter μ is initially set to 0.0001 and reduced by a factor of 100 at each iteration and the stopping parameter ϵ is set to 10^{-4}.

5.1 SSPM versus MSPM

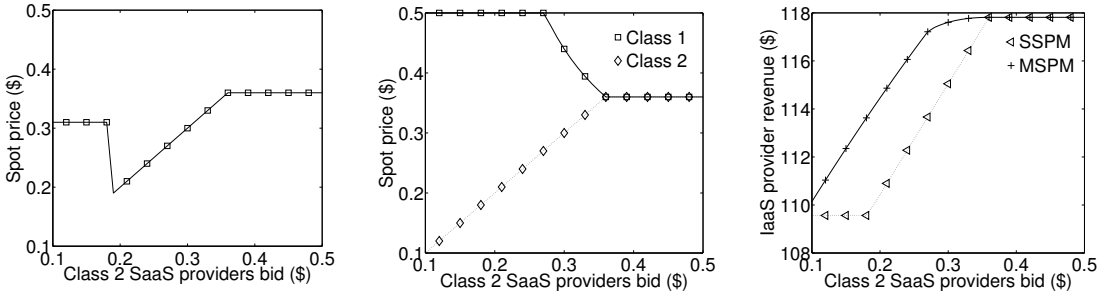
In this section we compare the SSPM and MSPM strategies. We first consider a homogeneous scenario in which all SaaS providers are characterized by the same parameters. In this scenario the behavior of the two policies coincide, but it is useful to understand the general features of the two strategies. We then consider three heterogenous scenarios to better assess the differences of the two pricing strategies.

In the homogeneous scenario, where all SaaS providers are characterized by the parameters setting described above, by symmetry all SaaS providers acquire the same number of VMs. The results are shown in Figures 2a, 2b, and 2c, for different values of the predicted load Λ_u ranging from 20 to 80 req/s. Figure 2a shows that, independently from the predicted load, the flat instances are always used by each SaaS provider up to the maximum value $\hat{f}_u = 4$. On the other hand, the number of on demand VMs bought by each SaaS provider grows from 0 to 11.18. Intuitively, the higher the predicted load, the greater the number of VM instances needed to ensure the SLA. As

a consequence, for higher loads the number of resources unsold by the IaaS provider after the first stage decreases (see Figure 2b). This directly affects the optimal spot price (see Figure 2c) and, in turn, the number of spot VMs each SaaS provider can buy during the second stage. When the number of on demand VMs sold is high, the spot price is set to $\sigma^* = 0.5$\$, because all the remaining resources can be sold as spot instances at the maximum price. When the load decreases, the spot price decreases as well, so that the IaaS provider can incentivate the SaaS providers to buy more VMs, thus increasing its revenue (the lower per spot VM revenue is compensated by the higher volume of spot instances sold). Note that, when the predicted load is lower than $\Lambda_u = 57$ req/s some spare capacity remains unsold even after the second stage (see Figure 2b), since the SaaS providers reach the equilibrium between the cost charged to their users (which is a function of the service response time) and the cost of additional VMs. Furthermore, the IaaS provider has no incentive to further reduce the spot price, i.e., the higher number of spot VMs sold does not compensate the lower per VM revenue. The IaaS provider maximizes its revenue by charging the maximum price each SaaS provider is willing to pay only when the spot instances are scarce; otherwise, it is more profitable to lower the spot price so to sell more spot VMs to the SaaS providers.

We now turn our attention to the heterogeneous scenario. In the first experiment we consider two classes with 5 SaaS providers each so to study the bid sensitivity of the SSPM and MSPM strategies. We fix to 0.5\$ the bid of the SaaS providers belonging to class 1 and we study the IaaS provider behavior when the bid of class 2 providers increases from 0.1 to 0.5\$. The predicted load Λ_u is fixed to 60 req/s.

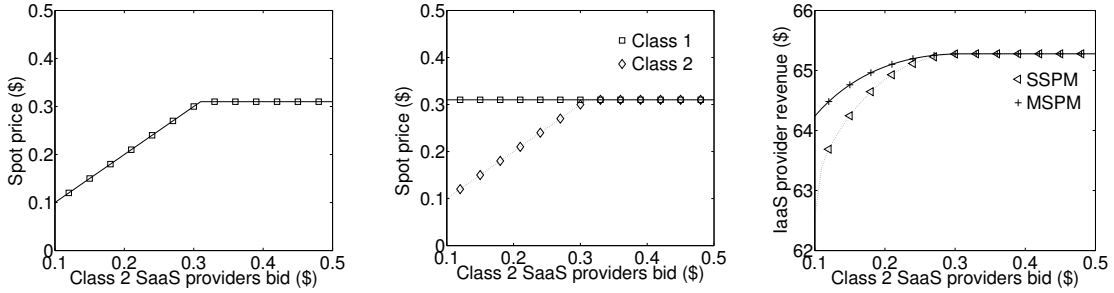
We first note that for both classes of SaaS providers the number of flat and on demand VMs bought does not change with respect to the previous homogeneous scenario in Figure 2a. The motivation is that in the first stage the SaaS providers determine independently the amount of flat and on demand instances to buy (under the implicit assumption that the IaaS provider has enough resources to allocate all the requested instances) and the final provisioning depends only on the provider parameters, but not on the bids.



(a) SSPM strategy (b) MSPM strategy (c) IaaS provider revenue

Figure 3: Behavior of SSPM and MSPM strategies with two classes of SaaS providers (predicted load 60 req/s)

Figures 3a, 3b and 3c show the optimal spot price achieved under the SSPM and MSPM strategies and the different IaaS revenue, respectively. As we can see from Figures 3a and 3b, the different bids sensibly affect the optimal spot price achieved by both policies. When the bid of class 2 providers is in the interval [0.36,0.5]\$, the optimal spot price remains constant and the two policies behave the same; in particular, their behavior does not change with respect to the homogeneous scenario (see Figure 2c when $\Lambda_u = 60$req/s). However, the policies behavior changes significantly for lower values of the bids. Under the SSPM policy, in the range [0.19,0.36]\$ the optimal spot



(a) SSPM policy (b) MSPM policy (c) IaaS provider revenue

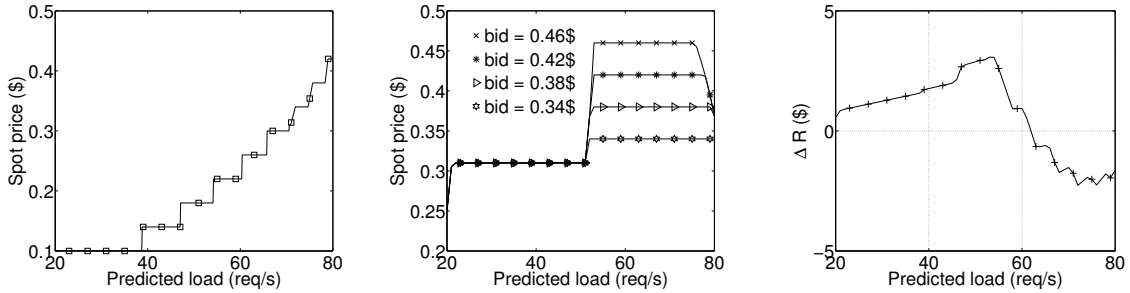
Figure 4: Behavior of SSPM and MSPM strategie with two classes of SaaS providers (predicted load 40 req/s)

price equals the bid of class 2 providers, while in the range [0.1,0.19]$ the IaaS provider has no incentive to further decrease the spot price. When the bid is lower than 0.19$, it is more convenient to increase the spot price to 0.31$ and to sell the spot VMs only to those SaaS providers of class 1, that bid 0.5$. In this case, it is better to sell less VMs to less SaaS providers, but at a price significantly larger. On the other hand, the behavior of the MSPM policy is quite different. When the bid of class 2 providers varies in [0.1,0.36]$, the IaaS provider always fixes their spot price equal to their bid, because it has no incentive to further decrease it. Even better, to compensate the revenue loss, it increases the spot price for class 1 providers, exploiting those customers that are willing to pay more. The MSPM strategy turns out to be more convenient for the IaaS provider: as Figure 3c shows, the IaaS provider revenue is greater under MSPM.

In the second experiment we consider a lower predicted load equal to 40 req/s, with the aim to study the policies behavior when the SaaS providers demand is lower and there is a larger amount of available spot instances. The results are shown in Figures 4a, 4b and 4c. When the bid of class 2 providers is in the range [0.31$,0.5$], the optimal spot price remains constant and the two strategies perform in the same way, as in the previous experiment. However, when the bid of class 2 providers is lower than 0.31$, their behavior changes. Under the SSPM strategy, the optimal spot price always equals the bid of class 2 providers and the spot VMs are always sold to all customers. In this scenario, it is not convenient to sell VMs only to class 1 providers as in the previous experiment. Conversely, under the MSPM policy, the spot price assigned to those providers is kept constant to 0.31$. As we have already seen before, exploiting customers that are willing to pay more is an effective strategy that ensures to MSPM a greater IaaS provider revenue, as shown in Figure 4c.

The last experiment investigates a heterogeneous scenario in which the 10 SaaS providers bid are $[0.1, 0.14, 0.18, 0.22, 0.26, 0.30, 0.34, 0.38, 0.42, 0.46]$ (one for each SaaS provider). The corresponding results are shown in Figures 5a, 5b and 5c for different values of the predicted load Λ_u, ranging from 20 to 80 req/s. In particular, Figure 5a shows the spot price under the SSPM policy; Figure 5b shows the spot prices for those SaaS providers with a bid greater than or equal to 0.34$ (for the other providers the spot price always equals the bid); Figure 5c shows the difference ΔR between the IaaS provider revenue using the MSPM and the SSPM policies.

As shown in Figure 5a, under the SSPM policy the IaaS provider sets the spot price to the minimum allowed value (we recall that $\sigma_1^U = 0.1$) until the predicted load is lower than 40 req/s. The resources demand is low and the IaaS provider finds more convenient to set a low price to incentivate all SaaS providers to buy VMs, rather than to cut off some of them. Only when the predicted load becomes greater than 40 req/s, the spot price starts to increase. In particular, every



(a) SSPM policy: spot price

(b) MSPM policy: spot prices for a subset of providers (for the not shown ones the price equals the bid)

(c) Difference between the IaaS provider revenue using the MSPM policy and the SSPM policy

Figure 5: Policies behavior in a heterogeneous scenario with different bids as function of the predicted load

increase in the spot price matches a decrease in the amount of spot VMs sold. The motivation is that every time the spot price rapidly increases, it becomes greater than the bid of a SaaS provider, that is no more allowed to buy spot VMs. The key point is that, as the predicted load increases, the SaaS providers become more aggressive in resource demand. In this scenario, the IaaS provider no longer needs to incentivate the SaaS providers to buy VMs, but it rather can increase the spot price to match the increasing customers demand.

Under the MSPM policy, the IaaS behavior is much simpler. For SaaS providers with a bid lower than 0.34$, the IaaS always sets the spot price up to the maximum value. Conversely, for those SaaS providers with higher bids, we notice a different behavior. When the predicted load is lower than 50 req/s, the spot prices are lower than the bids. From that point onwards, the resources become scarse and the spot prices increase up to the maximum value. As for the SSMP strategy, the IaaS provider no longer needs to incentivate the SaaS providers to buy VMs, but it can exploit providers that are willing to pay more. Interestingly, when the predicted load is greater than 75 req/s, the spot prices decrease again. At this load rate, the customers demand is very high and the spot VMs are just a few (less than 20). So, to sell less VMs to SaaS providers with lower bids the IaaS decreases the prices of those customers that are willing to pay more, so that they are more incentivized to buy VMs and become more aggressive acquiring more VMs. As we can see from Figure 5c, the IaaS revenue under the MSPM strategy is slightly lower at higher rates compared to that achieved by SSPM ($\Delta R < 0$). This suggests that when resources are scarse, it is more convenient to sell VMs to only a subset of the customers. Conversely, at lower load rates and higher amount of available VMs, the flexibility of the MSPM strategy ensures higher revenues ($\Delta R > 0$). In general, this experiment clearly shows the main characteristics of the proposed Stackelberg game based policy: its ability to increase and decrease the spot price as function of the customers demand in order to maximize the IaaS provider revenue.

5.2 Comparison with the Strategy in [4]

We now compare the SSPM and MSPM strategies with the one presented in [4]. Ardagna et al. studied a provisioning and pricing problem which is similar to the MSPM strategy presented in this paper. Differently from our two-stage strategy, they consider a one-stage scheme, where, at the same time: the SaaS providers determine the number of flat, on demand, and spot instances to buy as to maximize their profit given the SLA of the offered service; and, the IaaS provider determines the spot price σ_u as to maximize its revenue, taking into account that each SaaS

provider is characterized by a maximum price σ_u^U it is willing to pay for spot instance per hour. The conflicting situation is modeled as a GNEP and the provisioning and pricing policy is derived from the game equilibrium. In particular, given the specific problem structure, they showed that the dominant IaaS provider strategy consists in setting $\sigma_u = \sigma_u^U$, *i.e.*, in charging each SaaS provider always the maximum price.

Despite the similarities, our pricing and provisioning strategies are substantially different: in our two-stage approach, the SaaS providers first buy only flat and on demand instances, while the spot instances are provisioned only in the second stage, with the IaaS provider determining their price so to maximize its revenue. The resulting conflicting situation is modeled as a Stackelberg game, where the IaaS provider takes the role of the leader. Comparing the GNEP and Stackalberg-based approaches, we expect that under our strategies the SaaS providers are more likely to buy a higher number of flat and especially of on demand instances, which are more expensive (but also more reliable, as the IaaS provider cannot terminate them), because they are provisioned in the first stage. This should result in higher cost for the SaaS providers and higher revenue for the IaaS provider. Moreover, in the second stage, since the competition for the spot instances is modeled as a Stackelberg game, we expect the IaaS provider to experience higher revenues from the spot instances auction.

Table 2: SaaS providers parameters

	μ_u (req/s)	$\hat{f}_u$	σ_u^U (\$)
SaaS 1	11	5	0.38
SaaS 2	5	5	0.49
SaaS 3	13	3	0.16
SaaS 4	14	5	0.83
SaaS 5	11	4	0.28
SaaS 6	12	3	0.23
SaaS 7	12	3	0.44
SaaS 8	8	4	0.3
SaaS 9	11	5	0.54
SaaS 10	6	5	0.42

For the sake of comparison, we simulated a dynamic scenario using the three different policies to study how they affect IaaS provider revenue on the long run. We considered 10 SaaS providers, each offering a single service. Every hour each SaaS provider, given the forecasted load for the next hour, determines the number and type of VMs to allocate while the IaaS provider determines the spot price. The predicted load Λ_u of each SaaS provider is uniformly generated in the interval $[20, 80]$ req/s for every hour. For all the SaaS providers we set $R_u^{max} = 2s$, $C_u = 2\$$, and $U_u^{max} = 0.9$. The remaining parameters are shown in Table 2 and were kept constant during the simulation. Since in [4] the variable s^U is fixed a priori and is independent from the amount of sold flat and on demand instances, we fixed $s^U = 30$ for the whole simulation for all the policies. We run a simulation corresponding to a period of one week (168 hours).

Table 3 shows the breakdown of the number of allocated VMs and IaaS revenue per type of instance. As anticipated, our strategies result in a higher number of on demand instances (more than twice as much). The number of spot instances is the same, except for the SSPM strategy, but this is actually a consequence of having a fixed σ^U[9]. Nevertheless, the MSPM strategy yields

[9]For a more detailed comparison we should have modified the model in [4] to reflect our approach where the number of available spot instances depends on the number of allocated flat and on demand instances.

Table 3: Total number of VMs sold and relative revenue for the IaaS provider

	flat	on demand	spot	total
SSPM	6961.75	8706.31	4981.84	20649.9
MSPM	6961.75	8706.31	5040	20708.06
GNEP	6961.75	3910.19	5040	15911.94
SSPM	1670.82$	10795.82$	1551.05$	14017.69$
MSPM	1670.82$	10795.82$	1734.65$	14201.29$
GNEP	1670.82$	4848.63$	1575.48$	8094.93$

higher spot VMs revenues. This follows from the fact that the IaaS provider can choose for each SaaS provider the optimal price (from the IaaS perspective point of view), i.e., the price which maximises the amount the SaaS provider invest on spot instances, and which does not necessarily correspond to the SaaS provider maximal bid. Note that, in this experiment, even the approach from [4] yields a greater revenue from spot instances than the SSPM. This stems from the fact that in [4] the authors consider a single stage provisioning where the number of flat, on demand and spot instances is determined all at once. In such a case, it is no surprise that spot instances, which are cheaper, are preferred to the on demand. We remark this approach results in higher fraction of spot instances, which being not a reliable type of instances, can result in a overall degraded services to the SaaS users. In our two stage approach, instead, we guarantee the QoS using only flat and on demand, and use spot instances only to further improve performance.

Table 4: Average SaaS providers profit

	SSPM ($)	MSPM ($)	GNEP ($)
SaaS 1	74.14	73.96	75.55
SaaS 2	57.15	56.1	55.76
SaaS 3	76.56	78.04	80.79
SaaS 4	82.3	81.79	81.54
SaaS 5	70.55	71.36	73.84
SaaS 6	74.21	75.44	78.5
SaaS 7	80.32	79.96	80.31
SaaS 8	72.9	73.4	78.75
SaaS 9	75.35	74.86	74.49
SaaS 10	62.79	62.2	64.86

Tables 4-6 show, respectively, the profit, the cost and the average number of bought VMs for the different SaaS providers. We can observe that, indeed, our approach results in a higher number of VMs bought by the SaaS providers and a corresponding higher cost, which justifies the significant larger IaaS provider revenue (+64%). This holds for both SSPM and MSPM. Interestingly, the SaaS profits decrease only by a small fraction and in some cases (SaaS providers 2, 4 and 9) they even increase. This is not completely unexpected since as the number of VMs per service increases, the service response time decreases which in turn, given the SaaS revenue function, yields higher profits.

Table 5: Average SaaS providers cost to buy VMs

	MSPM ($)	SSPM ($)	GNEP ($)
SaaS 1	5.60	5.71	2.53
SaaS 2	18.15	18.56	15.15
SaaS 3	5.82	5.15	1.35
SaaS 4	4.02	4.24	3.47
SaaS 5	6.08	5.48	1.98
SaaS 6	6.47	5.68	1.71
SaaS 7	7.16	7.30	5.28
SaaS 8	11.28	10.92	3.30
SaaS 9	5.71	5.9	4.83
SaaS 10	14.20	14.48	8.53

Table 6: Average number of bought VMs

	MSPM	SSPM	GNEP
SaaS 1	9.99	10.74	7.87
SaaS 2	21.20	24.64	17.09
SaaS 3	10.76	6.56	6.96
SaaS 4	7.58	9.03	6.63
SaaS 5	10.37	8.21	7.66
SaaS 6	10.47	7.03	7.31
SaaS 7	9.5	10.76	7.52
SaaS 8	15.76	14.65	11.54
SaaS 9	9.43	10.95	7.90
SaaS 10	18.15	20.32	14.18

6 Related Work

Game theory is a useful formal tool for decision-making whenever the actions of multiple players are interdependent. It has been successfully applied to typical ICT problems, like resource allocation, Quality of Service (QoS), pricing, and load balancing, as surveyed in [2] for the networking area. Many studies (e.g., [6, 8]) proposed game-theoretic methods to solve resource allocation in networked systems from the resource owners' viewpoint. Load balancing in distributed systems has been also tackled through game theory, mostly exploiting solutions based on the Nash equilibrium [14].

More recently, game-theoretical approaches have been applied in the Cloud computing context, mainly to tackle resource allocation and pricing problems [4, 5, 15, 29, 32, 23, 22]. From a user perspective, Teng and Magoulès [29] studied the resource pricing and allocation among multiple users and determined the Nash equilibrium. Wei et al. [32] presented a QoS-constrained resource allocation, where Cloud users submit intensive computation tasks; however, they dealt only with on demand instances.

The IaaS provider's viewpoint is pursued in [15] to determine the optimal prices suggested by the IaaS provider and the user demands. In their model the provider suggests differentiated prices according to demand and the users update their requests on the basis of the proposed price. To this end, the formulation is based on a Stackelberg game and the policy consists in finding the Stackelberg equilibrium. However, the model does not consider spot instances and QoS constraints

of SaaS providers as well.

The most related work to ours is those by Ardagna et al. [4, 5] and in Section 5.2 we have compared their multiple price strategy to the single and multiple price ones we propose.

Game-theoretical frameworks for modeling the competition among a collection of IaaS providers have been recently proposed [23, 22]. Roh et al. [23] formulated a pricing problem for geographically distributed cloud resources. They formulated a concave game which describes the resource pricing of the IaaS providers as well as the resource request competition among multiple SaaS providers and characterized the Stackelberg equilibrium of the game. While they considered multiple IaaS providers and geographically distributed end users, pricing of spot resources is not taken into account. Non cooperative price and QoS games between multiple Cloud providers existing in a cloud market have been proposed by Pal and Hui [22]. Differently from our work, they consider a market of competing Cloud providers, each one offering a given application type for which a price and a QoS level has to be selected. Therefore, their work takes a different perspective and is complementary to ours.

Since the start of the Amazon's spot instance offering in 2009, a consistent number of research efforts have investigated pricing and performance issues related to such kind of instances, resulting in a variety of applied methodologies.

Several works, including [16, 24, 30, 34], studied mechanisms to improve the reliability and cost efficacy of spot instances, that are affected by volatility due to market dynamics. These research directions include fault-tolerance related mechanisms such as checkpointing [16, 34], resource provisioning and scheduling strategies [24, 30]. While such studies focus on helping customers (including SaaS providers) to improve the exploitation of spot instances, our goal is to maximize the IaaS provider revenue while satisfying the QoS constraints of the SaaS providers.

Focusing on the IaaS provider perspective, Zhang et al. [36] faced the problem of determining an optimal resource allocation in such a way to optimize the IaaS revenue while minimizing energy cost through a constrained discrete-time finite-horizon optimal control formulation. Wang et al. [31] presented an economic analysis of the long-term impact of spot instance pricing on present and future revenue.

Various works, including [28, 25, 26, 27, 35], consider the problem of designing efficient bidding strategies. Zaman and Grosu [35] explored combinatorial auction techniques to allocate VM resources. Tang et al. [28] proposed an optimal bidding strategy for spot instances to minimize the cost and volatility of resource provisioning. Song et al. [26] formulated the spot instance market as a modified repeated single price auction and proposed a pricing mechanism which considers the profit-reliability trade-off. The same authors [27] also proposed a truthful auction mechanism for spot market adapted from a repeated uniform price auction. Shi et al. [25] presented an online combinatorial auction in which VMs of heterogenous types are allocated in multiple consecutive time slots.

Some works have been devoted to analyzing the Amazon spot instances prices. Javadi et al. [17] performed a statistical analysis of the Amazon spot instances price patterns using traces of 2010-2011 and modeled the spot price and inter-price time of the spot instances using a mixture of Gaussians distribution. A reverse engineering analysis on how Amazon prices its spare capacity has been conducted by Agmon Ben-Yehuda et al. [1]; during the examined period (2011) Amazon prices were not market-driven but rather set most of the times at random from within a tight price interval via a dynamic hidden reserve price. However, the authors mentioned in the epilogue that Amazon radically changed the spot instances' pricing mechanism with respect to what observed in that study and the prices are no longer random. Also Xu and Li in [33] reached the same conclusion of [1] and, on the basis of this finding, chose to rather consider a scenario where the IaaS provider updates the spot price according to marked demand and studied dynamic pricing from a

revenue maximization perspective, formulating a stochastic dynamic program. Although still only one IaaS provider offers spot VMs, many argue that market economies will be increasingly prevalent to achieve high utilization in data centers that are often under-utilized. An open market to buy Cloud infrastructure capacity has been recently announced[10]. Companies have begun offering new services related to the spot instances market: CloudCheckr helps simplify bidding by tracking historical spot prices and using them to create predictive price models; VyScale [12] provides a cost optimization solution that includes Amazon spot instances.

7 Conclusions

In this paper we presented and analyzed service provisioning and pricing strategies for a Cloud system using a game theoretical approach. We considered a scenario where several SaaS providers offer a set of Web services with QoS constraints using the Cloud facilities provided by a IaaS provider which offers flat, on demand and spot VMs. The SaaS providers periodically acquire VMs from the IaaS provider to guarantee QoS to their users while maximizing their revenue. We modeled the resource provisioning as a two-stage process: in the first stage, the SaaS providers buy flat and on demand VMs at a fixed price; then, in the second stage, they bid to buy spot VMs instantiated on the unused IaaS provider capacity. The price of these spot instances is determined by the IaaS provider given the bids as to maximize its revenue.

We proposed two different pricing strategies for the IaaS provider, namely, SSPM and MSPM. The former is characterized by a single price which applies to all customers, while in the latter strategy the IaaS provider can conveniently set different spot instance prices for different users.

For the analysis, we modeled the second stage as a Stackelberg game where the IaaS provider takes the role of the game leader by setting the price(s) and the SaaS providers take the role of the followers competing for the spot resources. Given the complexity of the two underlying games, we had to resort to two different techniques to compute the game equilibrium for the two pricing schemes. In the case of MSPM, we show that it is possible to compute the equilibrium price and provisioning strategy by solving a suitable MPEC problem. In the case of the SSPM, we needed to resort to a brute force approach which, for each spot price, requires the solution of a suitable potential game which can be easily solved and is amenable to a distributed implementation.

Our experimental results analyzed the proposed policies behavior under different scenarios to study the impact of traffic load and providers heterogeneity on the pricing and resource provisioning. The results show that, not surprisingly, the MSPM pricing strategy, thanks the use of different prices for the different users, provides higher revenues to the IaaS providers with respect to the SSPM. As future work, we plan to device distributed algorithms and suitable protocols to compute the game equilibrium for a realistic implementation. Furthermore, we plan to extend and generalize the results of this paper: on the one hand, we plan to consider additional pricing schemes which generalize those considered in this paper; on the other hand, we plan to study the multiple IaaS provider setting, whereby the SaaS providers can acquire resources offered from multiple IaaS providers.

References

[1] O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir. Deconstructing Amazon EC2 spot instance pricing. *ACM Trans. Econ. Comput.*, 1(3), Sept. 2013.

[10] `http://tinyurl.com/lcd4xjn`

[2] E. Altman, T. Boulogne, R. El-Azouzi, T. Jiménez, and L. Wynter. A survey on networking games in telecommunications. *Comput. Oper. Res.*, 33(2):286–311, 2006.

[3] Amazon Web Services LLC. Amazon Elastic Compute Cloud (EC2), 2014. `http://aws.amazon.com/ec2/`.

[4] D. Ardagna, B. Panicucci, and M. Passacantando. A game theoretic formulation of the service provisioning problem in cloud systems. In *Proc. of WWW 2011*, pages 177–186, 2011.

[5] D. Ardagna, B. Panicucci, and M. Passacantando. Generalized Nash equilibria for the service provisioning problem in cloud systems. *IEEE Trans. Serv. Comput.*, 6:429–442, Oct. 2013.

[6] J. Bredin, R. T. Maheswaran, c. Imer, T. Başar, D. Kotz, and D. Rus. A game-theoretic formulation of multi-agent resource allocation. In *Proc. of AGENTS 2000*, pages 349–356. ACM, 2000.

[7] V. Di Valerio, V. Cardellini, and F. Lo Presti. Optimal pricing and service provisioning strategies in cloud systems: A Stackelberg game approach. In *Proc. of IEEE CLOUD 2013*, pages 115–122, June 2013.

[8] S. Dobzinski, N. Nisan, and M. Schapira. Approximation algorithms for combinatorial auctions with complement-free bidders. In *Proc. of ACM STOC 2005*, pages 610–618, 2005.

[9] F. Facchinei, H. Jiang, and L. Qi. A smoothing method for mathematical programs with equilibrium constraints. *Mathematical Programming*, 85:107–134, 1999.

[10] F. Facchinei and C. Kanzow. Generalized Nash equilibrium problems. *4OR: A Quarterly Journal of Operations Research*, 5:173–210, 2007.

[11] F. Facchinei and J.-S. Pang. *Finite-Dimensional Variational Inequalities and Complementarity Problems, Vol. I.* Springer, New York, 2003.

[12] Flux7. Vyscale - first spot-strategy-as-a-service, 2014. `http://flux7.com/vyscale/`.

[13] D. Fudenberg and J. Tirole. *Game Theory.* MIT Press, Cambridge, MA, 1991.

[14] D. Grosu and A. T. Chronopoulos. Noncooperative load balancing in distributed systems. *J. Parallel Distrib. Comput.*, 65(9):1022–1034, 2005.

[15] M. Hadji, W. Louati, and D. Zeghlache. Constrained pricing for cloud resource allocation. In *Proc. of IEEE NCA 2011*, pages 359–365, Aug. 2011.

[16] I. Jangjaimon and N. Tzeng. Effective cost reduction for elastic clouds under spot instance pricing through adaptive checkpointing. *IEEE Trans. Comput.*, 2014. to appear.

[17] B. Javadi, R. K. Thulasiram, and R. Buyya. Characterizing spot price dynamics in public cloud environments. *Future Gener. Comput. Syst.*, 29(4):988–999, 2013.

[18] Y. A. Korilis, A. A. Lazar, and A. Orda. Achieving network optima using Stackelberg routing strategies. *IEEE/ACM Trans. Netw.*, 5(1):161–173, Feb. 1997.

[19] D. G. Luenberger and Y. Ye. *Linear and Nonlinear Programming.* Springer, New York, 2008.

[20] N. Mankiw. *Principles of Economics.* South-Western Pub, 6 edition, 2011.

[21] D. Monderer and L. S. Shapley. Potential games. *Games and economic behavior*, 14(1):124–143, 1996.

[22] R. Pal and P. Hui. Economic models for cloud service markets: Pricing and capacity planning. *Theor. Comput. Sci.*, 496:113–124, 2013.

[23] H. Roh, C. Jung, W. Lee, and D.-Z. Du. Resource pricing game in geo-distributed clouds. In *Proc. of IEEE INFOCOM 2013*, pages 1519–1527, Apr. 2013.

[24] S. Shen, K. Deng, A. Iosup, and D. Epema. Scheduling jobs in the cloud using on-demand and reserved instances. In *Euro-Par 2013 Parallel Processing*, volume 8097 of *LNCS*, pages 242–254. Springer, 2013.

[25] W. Shi, L. Zhang, C. Wu, Z. Li, and F. Lau. An online auction framework for dynamic resource provisioning in cloud computing. In *Proc. of ACM SIGMETRICS 2014*, June 2014.

[26] K. Song, Y. Yao, and L. Golubchik. Exploring the profit-reliability trade-off in Amazon's spot instance market: A better pricing mechanism. In *Proc. of IEEE/ACM IWQoS 2013*, June 2013.

[27] K. Song, Y. Yao, and L. Golubchik. Improving the revenue, efficiency and reliability in data center spot market: A truthful mechanism. In *Proc. of IEEE MASCOTS 2013*, pages 222–231, 2013.

[28] S. Tang, J. Yuan, and X.-Y. Li. Towards optimal bidding strategy for Amazon EC2 cloud spot instance. In *Proc. of IEEE CLOUD 2012*, pages 91–98, 2012.

[29] F. Teng and F. Magoulès. Resource pricing and equilibrium allocation policy in cloud computing. In *Proc. of IEEE CIT 2010*, pages 195–202, June 2010.

[30] W. Voorsluys and R. Buyya. Reliable provisioning of spot instances for compute-intensive applications. In *Proc. of IEEE AINA 2012*, pages 542–549, Mar. 2012.

[31] P. Wang, Y. Qi, D. Hui, L. Rao, and X. Liu. Present or future: Optimal pricing for spot instances. In *Proc. of IEEE ICDCS 2013*, pages 410–419, July 2013.

[32] G. Wei, A. V. Vasilakos, Y. Zheng, and N. Xiong. A game-theoretic method of fair resource allocation for cloud computing services. *J. Supercomput.*, 54(2):252–269, 2010.

[33] H. Xu and B. Li. Dynamic cloud pricing for revenue maximization. *IEEE Trans. Cloud Comput.*, 1(2), July 2013.

[34] S. Yi, A. Andrzejak, and D. Kondo. Monetary cost-aware checkpointing and migration on Amazon cloud spot instances. *IEEE Trans. Serv. Comput.*, 5(4), 2012.

[35] S. Zaman and D. Grosu. Combinatorial auction-based allocation of virtual machine instances in clouds. *J. Parallel Distrib. Comput.*, 73(4):495–508, 2013.

[36] Q. Zhang, E. Gürses, R. Boutaba, and J. Xiao. Dynamic resource allocation for spot markets in clouds. In *Proc. of USENIX Hot-ICE 2011*, 2011.