Game-Theoretic Resource Pricing and Provisioning Strategies in Cloud Systems

Valeria Cardellini, Valerio Di Valerio, Francesco Lo Presti

Abstract—We consider several Software as a Service (SaaS) providers that offer services using the Cloud resources 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 customers. We model the resulting problem as a Stackelberg game. For each pricing scheme, we show 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.

Index Terms—Cloud computing, pricing, resource provisioning, Stackelberg games

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 grow 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 out or in 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 (typically 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-time unit price, which depends on the effective resource usage and is lower than the on demand price. 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-time unit price they are willing to pay. On the basis of the bids and its spare capacity, the IaaS provider sets the spot instances price. Amazon's EC2 cloud [1] offers the three types of VM instances and pricing models (i.e., flat, on demand, and spot instances) just described.

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.

In this paper, we consider a Cloud scenario where an 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 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 provider stipulates with its users.

We assume that resource provisioning is carried out as *two-stage* process. In the *first stage*, each SaaS provider independently determines the number of flat and on demand VM instances (which have a fixed known price). Then, in the *second stage*, the IaaS provider sells the unused capacity as spot instances to the SaaS providers, which compete for this residual capacity: the goal of each SaaS provider is to determine the number of spot instances to allocate which maximizes a suitable utility function, given

- *V. Cardellini and F. Lo Presti are with University of Rome Tor Vergata, Italy. E-mails: {cardellini, lopresti}@disp.uniroma2.it*
- *V. Di Valerio is with Sapienza University of Rome, Italy. E-mail: divalerio@di.uniroma1.it*

1. http://www.idc.com/promo/thirdplatform/fourpillars/cloud

the number of flat and on demand instances bought in the first stage. The goal of the IaaS provider is to determine the price (or set of prices) of the spot instances in order to maximize its profit.

For this two-stage scenario, we study the optimal service provisioning and pricing strategy. The analysis needs to focus on the second stage only, since, in the first stage, the provisioning for each SaaS provider reduces to a convex optimization problem which can be solved by means of standard techniques. In the second stage, instead, we analyze the competition between the IaaS and SaaS providers, each attempting to maximize its own revenue: the IaaS provider by determining the spot instances price, and each SaaS by determining the number of spot instances to buy.

We model the second stage on spot competition as Stackelberg game [2]. 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. In the considered Cloud scenario, the adoption of a leader-follower model well captures the interactions among IaaS and SaaS providers, since the IaaS provider can adjust the spot price (or prices) to maximize its revenue and take advantage of the competition among the SaaS providers for the limited resources.

We consider two different pricing models, 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 model the IaaS provider can conveniently set different spot instance prices for different customers. Our contributions are as follows:

1) We prove the existence of the Stackelberg equilibrium for both the SSPM and MSPM pricing models.

2) We compute the equilibria of the SaaS/IaaS Stackelberg game on spot instances. We cast the equilibrium computation as the solution of one (for the MSPM pricing model) or more (for the SSPM pricing model) suitable Mathematical Program with Equilibrium Constraint (MPEC) problems, which we solve using the solution proposed in [3].

3) We study the behavior of the proposed provisioning and pricing strategies under different workload and bidding configurations through numerical investigation. We show that the IaaS provider can maximize its revenue by properly setting the spot price (or prices) as function of customers demand and bids. Our results show that when the demand exceeds the supply, the SSPM with a single price for all customers yields higher revenues to the IaaS provider. Conversely, when the demand does not saturate the supply, it is the MSPM pricing scheme which yields higher revenues, thanks to the inherent flexibility in the use of different prices for different customers.

Our work has been motivated and inspired by the work by Ardagna et al. in [4], [5]. They considered a one stage version of the provisioning and pricing problem, proposing a MSPM pricing scheme, and modeled the resulting conflicting situation as a Generalized Nash Equilibrium Problem (GNEP), which can be shown to reduce to a potential game. Our results complement and extend their analysis by considering the privileged position of the IaaS provider in the competition among the different stakeholders. To this end, we also provide a thorough discussion and comparison of the two approaches. It is worth observing that, despite the similar setting, the analysis of the competition as a Stackelberg game has required a complete new machinery, *i.e.*, the solution of an MPEC, for the equilibrium analysis.

The rest of the paper is organized as follows. In Section 2 we discuss related works. Section 3 introduces our system model. We define the two-stage resource provisioning and pricing scheme in Section 4 and discuss the solution method of the SSPM and MSPM strategies in Section 5. In Section 6 we analyze through numerical experiments the SSPM and MSPM strategies; we also compare their results to those achieved by the formulation in [4]. We conclude the paper and present directions for future work in Section 7. The paper is accompanied by a supplemental document which contains the proof of Theorem 4.

2 Related Work

Game-theoretical approaches have been widely applied in the Cloud computing context to tackle resource provisioning and pricing problems [4], [5], [6], [7], [8], [9], [10], [11], [12].

From a user perspective, Teng and Magoulès [8] studied, given a simple pricing mechanism, the resource allocation among multiple users and determined the Nash equilibrium. Wei et al. [9] presented a QoS-constrained resource allocation, where Cloud users submit intensive computation tasks; however, they dealt only with on demand instances without considering the pricing problem. The IaaS provider's viewpoint is pursued in [7] to determine the optimal prices suggested by the IaaS provider and the user demands through a Stackelberg game. Although the authors consider the joint resource provisioning and pricing problem, their model does not consider spot instances and QoS constraints of SaaS providers.

The most related work to ours is those by Ardagna et al. [4], [5]. They considered a one-stage game and proposed a GNEP-based formulation. As we will see in Section 6.2, where we present an exhaustive comparison of the two approaches, their solution results in a substantially different provisioning and pricing strategy and lower revenues for the IaaS provider.

Game-theoretical frameworks for modeling the competition among a collection of IaaS providers have

been recently proposed. Roh et al. [10] considered a pricing problem for geographically distributed cloud resources. They formulated a concave game for the resource pricing of the IaaS providers and the resource competition among multiple SaaS providers and characterized the Stackelberg equilibrium of the game; however, pricing of spot resources is not taken into account. Non cooperative price and QoS games among multiple Cloud providers have been proposed by Pal and Hui [11]. Differently from us, they considered 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. Tang and Chen [12] proposed a Stackelberg game formulation for the joint pricing and capacity allocation problem in a scenario with multiple IaaS and SaaS providers. However, they considered a simplified model, in particular without multiple pricing plans and competition on VMs. Ardagna et al. [6] recently proposed a game-theoretic approach to allocate resources of multiple IaaS providers to multiple competing SaaS providers. Similarly to their previous works [4], [5], the authors proposed a GNEP-based formulation and then devised a distributed algorithm for identifying Generalized Nash equilibrium.

Actually, many state-of-the-art approaches focus only on the pricing problem and propose solutions based on auction theory [13], [14], [15], [16], [17], [18], [19], [20], [21]. An $m+1$ price auction in which the customers bid for multiple units of the same VM type is proposed in [13], [14]. These works differ for the mechanism adopted to compute the amount of auctioned VMs. Combinatorial auctions are proposed in [17], [21]. In these auctions a customer bids for a bundle of items, i.e., for a combination of VMs of different types. These works aim at maximizing the social welfare and rely on heuristics to compute the winning customers. Fujiwara et al. [16] proposed a combinatorial double auction considering multiple providers. In a double auction not only the customers submit their bids for a bundle of services, but also providers simultaneously submit their ask prices. Bonacquisto et al. [19], [20] focused on a reverse auction, which is similar to a double auction except that customers do not submit bids. Multiple sellers compete submitting their ask prices and, usually, the customer selects the lowest one. Despite some similarities with our approach, the works that rely on auction theory mainly focus on the pricing problem. The number of VMs each customer desires is computed a priori and is basically assumed to be given. Conversely, in our work we take into account the joint VM provisioning and pricing problem.

Since the launch of Amazon EC2 spot instances in late 2009, researchers have investigated spot pricing and performance issues, applying a variety of methodologies. Several works studied mechanisms, such as checkpointing [22], [23] and resource provisioning [24], to improve the reliability and cost efficacy of spot instances, that are affected by volatility due to market dynamics. While such studies focus on helping customers to better exploit spot instances, our goal is to maximize the IaaS provider revenue while satisfying the QoS constraints of the SaaS providers.

Some works analyzed the Amazon spot instances prices. Javadi et al. [25] performed a statistical analysis of the Amazon spot instances price patterns using traces of 2010-2011. A reverse engineering analysis on how Amazon prices its spare capacity was conducted by Agmon Ben-Yehuda et al. [26]; during the examined period (2011) Amazon prices were set most of the times at random from within a tight price interval via a dynamic hidden reserve price. However, as noted in the epilogue of [26], Amazon radically changed the spot pricing mechanism and the prices are no longer random. Xu and Li in [27] reached the same conclusion of [26] and considered 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.

Spot instance usage is recommended in the industry as an effective choice for batch processing, high performance computing, and development/testing[2], and companies (e.g., Cloudyn and Flux7) offer services for the spot market. On the IaaS side, Google recently introduced preemptible instances, which may be revoked at any time but their price is deterministic.

3 System Model

We consider a set $\mathcal{U}$ of SaaS providers that offer services using the Cloud resources of an IaaS provider. We assume that each SaaS provider $u \in \mathcal{U}$ offers a single service characterized by a SLA, which stipulates the QoS levels, *i.e.*, service response time and the associated cost/penalty for its use[3].

The 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 compute, memory and network capacity[4]. Each 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 providing up to $\mathcal{S}$ VMs which are offered to customers as flat, on demand, and/or spot instances. Flat instances are characterized by one-time payment, e.g., every one or three years, for each reserved VM regardless their actual usage, plus a payment of φ per time unit of

2. http://tinyurl.com/j2jp5jc

3. 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.

4. This is not a limiting assumption because Amazon, for example, runs independent spot markets for each instance type [28].

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 the service of SaaS provider u
U_u^{max}	Maximum allowed utilization of VM executing the service of SaaS provider u
φ	Time unit cost for one *flat* VM
δ	Time unit cost for one *on demand* VM
σ^L	Minimum time unit cost for one *spot* VM, set by IaaS provider
σ_u^U	Maximum time unit cost for one *spot* VM that 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

actual use. On demand instances have no one-time payment and are only charged at a per time unit price δ, which we assume to be strictly larger than φ. Spot instances are charged at a price σ which is dynamically set and depends on the customers' 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], [29]. 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 each SaaS provider u, the per-time unit 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 profit. We assume the 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 utility arising from the 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 of class $\mathcal{C}^2$ with respect to s_u, with $\partial^2\Theta_u/\partial s_u^2 < 0$, to reflect the law of diminishing marginal utility [30].

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

4 Resource Pricing and Provisioning: a Stackelberg Game Approach

In this paper we assume that SaaS providers periodically provision the VMs to run their services. We propose a *two-stage* resource pricing and service provisioning scheme. In the first stage, each SaaS provider determines the number of flat and on demand instances[5] which guarantees the performance level defined in the SLA agreed with its prospective users and maximizes its profit. For the provisioning, the SaaS provider uses a prediction of the workload expected in the next time interval. We assume that the IaaS provider has enough resources to always satisfy the request for on demand instances.

In the second stage, the IaaS provider leases 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, we study two different pricing strategies, named SSPM and MSPM. With SSPM the IaaS provider sets a single price which applies to all customers; customers whose submitted bid is lower than the actual price will not get any instance; with MSPM the IaaS provider can conveniently set different spot instance prices for different customers.

4.1 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 profit, while sustaining the predicted 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[6]:

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

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

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

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

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], [29], 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. In particular, in [5] the authors show in a quite similar setting

5. Observe that, in case of flat instances, this number represents the number of allocated VMs among the reserved ones (a SaaS provider does not pay the per unit of time cost for unused flat instances).

6. 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.

how the gap with the optimal integer solution that can be achieved rounding the fractional allocation is small, especially for large scale services. This is a consequence of current cloud pricing models, that make the weight of a single VM small.

4.2 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 bought at an affordable price, while the IaaS provider can make profit from the otherwise unsold resources. Without lack of generality, we assume that the IaaS provider sets a public 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. Since the spot price is expected to be (much) lower than the flat/on demand cost, the SaaS provider can increase its utility if cheaper VMs are made available. As expected, the formulation of the optimization problems of both the SaaS providers and the IaaS provider strictly depends on the adopted pricing model, i.e., SSPM or MSPM. We analyze them in the following.

4.2.1 Same Spot Price Model (SSPM)

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

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

$$\textbf{Problem SSPM\_SaaS}_{OPT}$$
$$\textbf{max } \Theta_u(\bar{f}_u, \bar{d}_u, s_u, \sigma)$$
$$\textbf{subject to: } \sum_{u \in \mathcal{U}} s_u \leq s^U, \quad s_u \geq 0 \tag{6}$$
$$y_u \in \{0,1\} \tag{7}$$
$$s_u \leq s^U y_u \tag{8}$$
$$(\sigma_u^U - \sigma) \leq M y_u \leq (\sigma_u^U - \sigma) + M \tag{9}$$

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 decision 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 leased 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:

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

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

4.2.2 Multiple Spot Price Model (MSPM)

In the MSPM, the IaaS provider can conveniently set different prices for different customers. 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:

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

where $\bar{f}_u$, $\bar{d}_u$, and s^U are defined as in Section 4.2.1. 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:

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

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.

4.2.3 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 on 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.

We model such a conflicting situation as a Stackelberg game [2]. 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 equilibrium. 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, given the spot price(s), must decide the number s_u of spot instances to lease, competing among themselves for the shared pool of available instances s^U (the SaaS providers subgame). This equilibrium is characterized by the property that for the given set of prices, the SaaS providers adopt a strategy such that none of them could improve its profit by changing it unilaterally, and the IaaS provider would not benefit by modifying the chosen set of prices, since no other SaaS providers equilibrium would improve its revenue.

5 Stackelberg Game Analysis

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. In this section we demonstrate the existence of the Stackelberg equilibrium and how to compute it in the case of the MSPM and SSPM. To this end, we will proceed in two steps: first, in Section 5.1, we will study the SaaS providers subgame, that is the competition that arises among SaaS providers for a given spot price(s); then, in Section 5.2 we will turn to the Stackelberg game equilibrium computation, that is, the IaaS (leader) problem of determining the optimal pricing strategy.

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 denote by $K_u = \{s_u \geq 0\}$ the feasible strategies set for SaaS provider u. Let Ω denote the compact set $\left\{s \,\middle|\, \sum_{u\in\mathcal{U}} s_u \leq s^U\right\}$. We denote by $K = (K_1 \times K_2 \ldots K_N) \cap \Omega$ the set of feasible strategies of all the players. For convenience, we also denote with $g(s) = (g_u(s))_{u=1}^N$ the vector function that represents the compact set K.

5.1 SaaS Providers Subgame

We first consider the followers subgame, i.e., the SaaS providers competition that arises once the IaaS provider fixes its strategy, i.e., the spot price(s). Without loss of generality, we study the SaaS subgame that arises from the MSPM. We will later show that the SSPM subgame can be regarded as a particular case of the MSPM subgame.

Since the SaaS providers act simultaneously, we can model the SaaS subgame as a Generalized Nash game [31]. 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 constraints (11), which includes all SaaS providers decision variables s_u. More specifically, our problem is a *Jointly Convex* GNEP [31]. 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)[7]. 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 [31]. 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 equilibria of a GNEP. Roughly speaking, a variational equilibrium can be considered as a local equilibrium over larger neighborhoods than other equilibria for the players. In this respect, it can be regarded as a *better* equilibrium point and therefore represents a valuable target for an algorithm [31]. Hence, in the remainder of this paper we focus on variational equilibrium as solutions of the SaaS subgame, instead of the set of all generalized Nash equilibria.

7. Given a closed and convex 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 $(s - s^*)^T F(x^*) \geq 0 \quad \forall s \in K$.

We have shown that the problem of finding a variational equilibrium of the SaaS providers subgame for a given IaaS strategy can be formulated as the problem of finding the solution of a proper VI. We now establish two key properties of the $VI(K, F(s;\sigma))$, namely that the function F is strongly monotone[8] and the existence of the generalized Nash equilibrium for the followers subgame.

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

Proof: The Jacobian of function F takes the following form:

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

JF is a diagonal matrix whose generic entry $a_u = -\partial^2\Theta_u/\partial s_u^2$ is, by assumption, a strictly positive continuous function. The Weierstrass theorem ensures that the minimum $\hat{a}_u = \min_{s_u \in [0, s^U]} a_u$ exists in the interval $[0, s^U]$ for each a_u. If we choose a constant $0 < \alpha < \min_{u\in\mathcal{U}} \hat{a}_u$ 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 exactly one variational equilibrium of the followers subgame is a direct consequence of the strong monotonicity of the function $F(s;\sigma)$ [32].

We now turn our attention to the the SSPM subgame. For any given spot price σ, we can consider the SaaS subgame restricted to the 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 **SSPM_SaaS**$_{OPT}$ in Section 4.2.1 reduces to the following optimization problem:

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

It is easy to realize that this optimization problem is a particular case of **MSPM_SaaS**$_{OPT}$ in which $\sigma_u = \sigma$, $\forall u \in U$. Hence, Theorem 1 and 2 hold for the SSPM subgame as well.

5.2 Stackelberg Game Equilibrium

We now turn our attention to the IaaS (leader) problem of determining the optimal pricing strategy. We will deal with the MSPM and SSPM separately.

8. F is strongly monotone on K if there exists a constant $c > 0$ such that for all pairs $s, y \in K$, $(s-y)^T(F(s)-F(y)) \ge c\| s-y\|^2$ holds.

5.2.1 MSPM Game Equilibrium

As a first step, we demonstrate the existence of the Stackelberg equilibrium of the game.

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

Proof: In the case of the MSPM, as shown below, we can map the Stackelberg game to a *Mathematical Program with Equilibrium Constraint* (MPEC) [3]. Under the assumption that: (i) function $F(s;\sigma)$ is strongly monotone, (ii) the set of IaaS provider feasible strategies is a compact set, (iii) K is a compact set, Corollary 2 in [3] ensures the existence of the MPEC solution and, in turn, of the Stackelberg equilibrium.

In order to compute the optimal IaaS pricing strategy, we have to solve the **IaaS**$_{OPT}$ problem with the additional (and non trivial) constraint that the SaaS providers strategy s is a solution of the SaaS subgame. The resulting problem takes the following form:

$$\textbf{Problem IaaS}_{MPEC}$$
$$\textbf{max } \sum_{u\in\mathcal{U}} \bar{f}_u\varphi + \bar{d}_u\delta + s_u\sigma_u$$
$$\textbf{subject to: } \sigma^L \le \sigma_u \le \sigma_u^U, \quad \forall u \tag{14}$$
$$s \in SOL(\sigma) \tag{15}$$

where $SOL(\sigma)$ is the solution of the SaaS subgame. As shown in Section 5.1 $SOL(\sigma)$ is a singleton and can be computed by solving the variational inequality $VI(K, F(s;\sigma))$.

The problem **IaaS**$_{MPEC}$ is a *Mathematical Program with Equilibrium Constraint* (MPEC) [3], that is, an optimization problem whose constraints include variational inequalities. Because of (15) we cannot solve the MPEC directly. We thus follow the approach proposed in [3] that, under the assumption that function $F(s;\sigma)$ is strongly monotone, allows us to compute stationary points of the MPEC.

As a first step, we replace (15) with its Karush Kuhn Tucker (KKT) conditions [33]. We obtain the following non linear programming problem:

$$\textbf{max } \sum_{u\in\mathcal{U}} \bar{f}_u\varphi + \bar{d}_u\delta + s_u\sigma_u$$
$$\textbf{subject to: } \sigma^L \le \sigma_u \le \sigma_u^U, \quad \forall u \tag{16}$$
$$F(s,\sigma) - \nabla_s g(s)\lambda = 0 \tag{17}$$
$$g(s) \le 0 \tag{18}$$
$$\lambda^T g(s) = 0, \quad \lambda \ge 0 \tag{19}$$

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 [3], 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:

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

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

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

$$F(s,\sigma) - \nabla_s g(s)\lambda = 0 \quad (21)$$

$$g(s) + z = 0 \quad (22)$$

$$\sqrt{(z_i - \lambda_i)^2 + 4\mu^2} - (z_i + \lambda_i) = 0, \forall i \in \{1,\ldots,l\} \quad (23)$$

where $z \in \Re^l$ is an auxiliary variable. In $\mathbf{P}(\mu)$, constraint (22) replaces constraint (18), while constraint (23) replaces (19). It can be shown that $\mathbf{P}(\mu)$ corresponds to the original problem when $\mu = 0$[9]. We refer the reader to [3] 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 [3], 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 [3] (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 Section 6, the algorithm converged in no more than 3 iterations using $\epsilon = 10^{-4}$.

Algorithm 1 Algorithm S [3]

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

5.2.2 SSPM Game Equilibrium

The analysis of the SSPM game is complicated by the presence of integer variables in the SaaS providers problem **SSPM_SaaS**$_{OPT}$[10]. As a result, the IaaS revenue Θ_I is in general not continuous and thus the assumptions of Corollary 2 [3] are not satisfied. Discontinuity may arise in correspondence of spot prices equal to the SaaS provider bids, since, when the price exceeds a bid, the corresponding SaaS provider gets no spot VM at all, which might result in a jump of the IaaS provider revenue.

For the SSPM game we resort to the following approach. The basic idea is to divide the original SSPM game in multiple Stackelberg games, so that the following properties hold: (i) the collection of these games is equivalent to the original game SSPM, (ii) the equilibrium of each game can be computed using an MPEC, and (iii) one of these equilibria corresponds to the original SSPM game equilibrium.

Let us assume that all SaaS providers' bids are different and let SaaS providers $u = 1,\ldots,N$ be labeled in increasing bid order, *i.e.*, $\sigma_1^U < \sigma_2^U < \ldots < \sigma_N^U$. The result can be easily generalized to the case where multiple SaaS providers bid the same value. Denote by $I_k = (\sigma_k^U, \sigma_{k+1}^U]$, $k = 1,\ldots,N-1$ and $I_0 = [\sigma_0^U, \sigma_1^U]$, where $\sigma_0^U = \sigma^L$, the non-overlapping intervals delimited by the successive increasing bid values. If we restrict the SSPM problem to any such interval, we obtain a Stackelberg game, denoted by SSPM$_k$, which is basically SSPM with the spot price σ limited to the interval I_k and the set of SaaS providers restricted to $U_k = \{u \in U | \sigma_u^U > \sigma_k^U\} = \{k+1,\ldots,N\}$. By construction, the collection of these games is equivalent to the original game SSPM. Consider a slightly modified version of each problem denoted by SSPM$_k^c$, where the interval I_k is replaced by its closure $I_k^c = [\sigma_k^U, \sigma_{k+1}^U]$. Each such a problem has the same structural properties of the MSPM problem, and Corollary 2 in [3] ensures the existence of a Stackelberg equilibrium. The following key result holds.

Theorem 4: There exists at least one Stackelberg equilibrium of the SSPM game. Let (σ_k^*, s_k^*) be the Stackelberg equilibrium of the subproblem SSPM$_k^c$, $\forall k \in \{0,1,\ldots,N-1\}$, and let σ^* denote the price corresponding to the largest revenue $\Theta^* = \max_{k=0,\ldots,N-1} \sum_{u\in\mathcal{U}} s_{u,k}^* \sigma_k^*$. The equilibrium point (σ^*, s^*) is a Stackelberg equilibrium for SSPM.

For its length, the proof is presented in the supplemental file to this paper.

Theorem 4 ensures the existence of an equilibrium and indicates a simple approach to compute it which involves the solution of (at most) N MPECs: we compute the Stackelberg equilibrium (σ_k^*, s_k^*) of each SSPM$_k^c$, $k = 0,\ldots,N-1$, using Algorithm S, and then we take the spot price σ^* corresponding to the largest $\Theta^* = \max_{k=0,\ldots,N-1} \sum_{u\in\mathcal{U}} s_{u,k}^* \sigma_k^*$. The pair (σ^*, s^*) is the problem equilibrium.

6 Experimental Results

In this section we investigate through numerical experiments the behavior of the proposed provisioning and pricing strategies. We start in Section 6.1 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

9. Observe that for $\mu = 0$, $\sqrt{(z_i - \lambda_i)^2} - (z_i + \lambda_i) = -2\min(z_i, \lambda_i)$ and, making $-2\min(z_i, \lambda_i) = 0, \forall i$, implies $z\lambda = 0$ and satisfaction of constraints (19).

10. We remark that we do not impose any integer constraint on the number of VMs purchased by the SaaS providers. The integer variables in **SSPM_SaaS**$_{OPT}$ are just auxiliary variables.

SaaS providers and the associated spot prices under different workload and bidding configurations. Then, in Section 6.2 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, which satisfies our assumptions in Section 3 (we omit the proof for space reason). 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 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 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-time unit cost for the provisioned VMs (where $\sigma_u = \sigma$ in case of SSPM), the per-time unit SaaS profit is:

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

where the first term is 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.

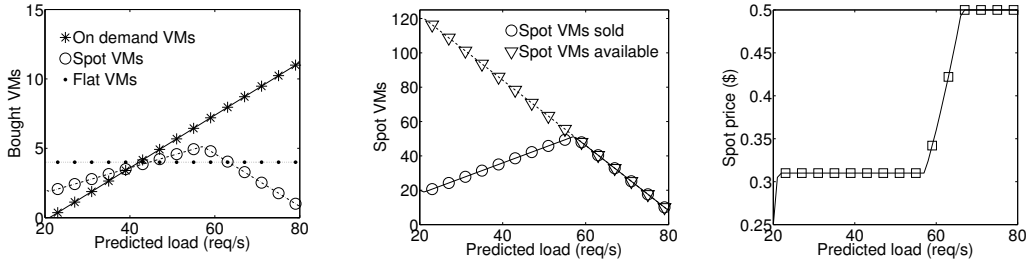
In the experiments, we consider one IaaS provider which leases its resources to 10 SaaS providers. If not differently stated, we set $\mathcal{S} = 160$, $\varphi = 0.24\$$, $\delta = 1.24\$$, $\hat{f}_u = 4$, $\mu_u = 10$ req/s, $U_u^{max} = 0.9$, $\sigma^L = 0.1\$$, $\sigma_u^U = 0.5\$$, $C_u = 1\$$, $R_u^{max} = 1$ s for all $u \in \{1, \ldots, 10\}$ and 1 hour as time unit. We also assume all SaaS providers are characterized by the same value of predicted load Λ_u. Such parameters setting corresponds to that adopted in [4] for the sake of comparison. For the analysis, we implemented in MATLAB the algorithms presented in Section 5 as well as those in [4]. For the solution of the MPEC problem via Algorithm 1, μ is initially set to 10^{-4} and reduced by a factor of 100 at each iteration and the stopping parameter ϵ is set to 10^{-4}.

6.1 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 coincides, 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 1a, 1b, and 1c, for different values of the predicted load Λ_u ranging from 20 to 80 req/s. Figure 1a shows that, independently from the predicted load, each SaaS provider uses all the reserved flat instances ($\hat{f}_u = 4$). The number of on demand VMs bought by each SaaS provider, instead, grows from 0 to 11.18. This is no surprise since, the higher the predicted load, the greater the number of VM instances needed to ensure the SLA. As a consequence, the number of resources unsold by the IaaS provider after the first stage decreases as the load increases (see Figure 1b). This directly affects the optimal spot price (see Figure 1c) and, in turn, the number of spot VMs each SaaS provider can buy during the second stage. At high loads, only few resources are left and the IaaS provider can set the spot price to its maximum value $\sigma^* = 0.5\$$ at which all VMs are sold. For intermediate level of load, more resources remain unsold after the first stage. The IaaS provider decreases the spot price, thus incentivizing the SaaS providers to buy more VMs. In this scenario, the lower per spot VM revenue is compensated by the higher volume of spot instances sold. Eventually, for loads below $\Lambda_u = 57$ req/s the IaaS provider does not further reduce the spot price, as the additional spot VMs sold would not compensate the lower per VM revenue. As a consequence, as the load decreases, the spot price remains fixed at $0.31\$$ and some capacity unsold (see Figure 1b).

We now turn our attention to the heterogeneous scenario. In the next set of experiments we consider a scenario with only two classes with 5 SaaS providers each. 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 SaaS providers varies in the interval $[0.1, 0.5]$. This scenario can be regarded as a simple variation of the homogeneous one, which was characterized by one class of 10 SaaS providers with a maximum bid equal to $\sigma_u = 0.5\$$. It is also worth observing that for both classes of SaaS providers the number of flat and on demand VMs bought does not change with respect to the homogeneous scenario shown in Figure 1a. This simply follows from the fact that the amount of flat and on



(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

Fig. 1: Behavior of SSPM and MSPM strategies in the homogeneous scenario

demand instances depends neither on the spot pricing strategy nor on the bids, but only on the SaaS provider parameters which assume the same values as in the homogeneous experiments.

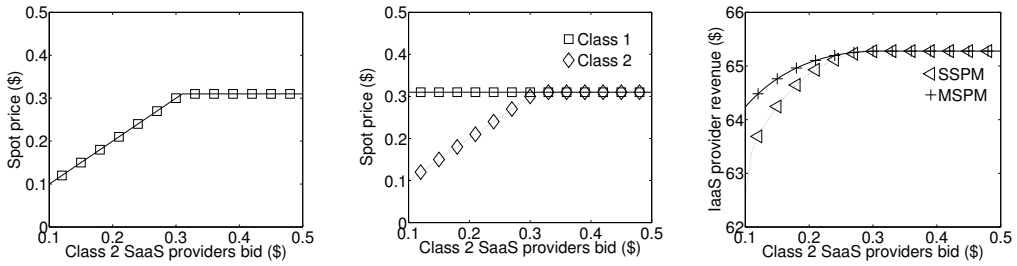
In Figures 2-4 we show the optimal spot price and the IaaS provider revenue achieved under the SSPM and MSPM strategies, for increasing value of predicted load $\Lambda = 40, 60$ and 80 req/s, respectively. Differently from the homogeneous scenario, we can observe that the spot price under SSPM and MSPM exhibits significantly different behaviors as the load varies, as also does the IaaS provider revenue. We analyze the strategies in detail below.

Figure 2 shows the optimal spot price and IaaS revenue under the SSPM and MSPM strategies for a predicted load of $\Lambda_u = 40$ req/s. To analyze the policies behavior, let us first observe that in the homogeneous scenario the optimal IaaS spot price for a predicted load of $\Lambda_u = 40$ req/s was 0.31\$ (see Figure 1c). It is then no surprise that when Class 2 provider bid lies in the interval [0.31,0.5]\$ the optimal spot price is 0.31\$ under both policies (and in the case of MSPM for both classes) as, due to the relative high bid of Class 2 providers, the scenario is practically equivalent to the homogeneous one, despite the different bids. Conversely, the two policies differ when Class 2 providers bids are lower than 0.31\$: under the SSPM policy, the IaaS provider needs to decrease the spot price to match the lower Class 2 providers bid (Class 2 providers would otherwise be cut off with a negative effect on the IaaS revenue); differently, under the MSPM policy, the IaaS provider can differentiate the price and keep Class 1 providers spot price equal to 0.31\$ while setting Class 2 providers spot prices equal to their (lower) bid. The difference is reflected in the IaaS revenues under the two policies, with MSPM yielding higher values than SSPM when Class 2 providers bids are lower than 0.31\$.

Figure 3 shows the results for a predicted load of $\Lambda_u = 60$ req/s. For this value of the predicted load, the optimal IaaS spot price in the homogeneous scenario is 0.36\$ (up from 0.31\$). Following the same arguments as above, it is then clear that when the bid of Class 2 SaaS providers is in the interval [0.36,0.5]\$, the optimal spot price is 0.36\$ under both policies. The policies behavior changes significantly for lower values of the bids, however. Under the SSPM policy, the IaaS provider decreases the spot price to match the lower Class 2 customers' bid, till the bids decrease to 0.19\$. For lower values of Class 2 providers' bids, the IaaS provider maximizes its revenue by increasing the price to 0.31\$, therefore cutting off Class 2 customers and selling spot instances to Class 1 customers only (observe that 0.31\$ is the optimal spot price when resources are not saturated, which is the case, despite the higher load, because only half of the customers receives spot instances). The behavior under the MSPM policy is quite different; as Class 2 bid decreases, as before, the IaaS provider sets the spot price of Class 2 customers equal to their bid; at the same time, the IaaS provider instead increases the spot price for Class 1 providers, exploiting their willingness to pay up to 0.5\$ per spot instance. Also in this scenario, the price differentiation under the MSPM strategy yields higher revenue to the IaaS provider, as shown in Figure 3c.

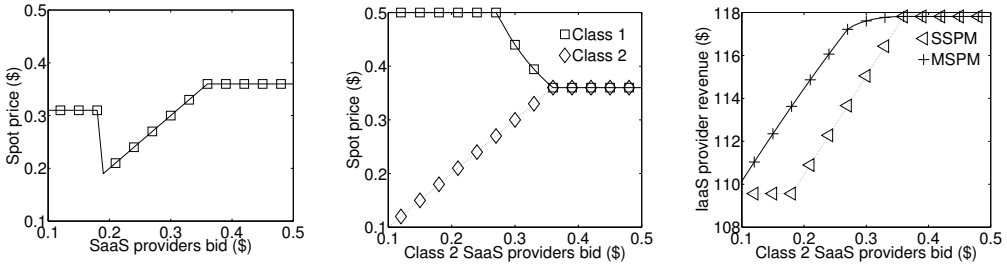
The results for a predicted load of $\Lambda_u = 80$ req/s are shown in Figures 4a, 4b and 4c. Given the high load, this scenario is characterized by the availability of only few spot instances which, in the homogeneous case, are sold by the IaaS provider at the maximum price of 0.5\$. In the heterogeneous case, under the SSPM policy, the IaaS provider sells all the spot VMs to Class 1 SaaS providers at the maximum price. Under the MSPM policy the behavior is again quite different. The IaaS provider, as in the previous case sets Class 2 providers spot price equal to their bid; the spot price assigned to Class 1 providers is kept equal to their bid of 0.5\$ only when the bid of Class 2 providers is higher than 0.31\$, otherwise it is decreased as well. Indeed, at equilibrium, to increase its revenue, the IaaS provider needs to reduce the number of VMs assigned to SaaS providers with lower bids. Since these players cannot be excluded as in SSPM, the IaaS strategy is to decrease the spot price of Class 1 SaaS providers so that they are willing to buy more VMs. As shown in Figure 4c, in this high load scenario, differently from the previous two cases, it is the SSPM strategy to yield higher revenues.

The variety of behavior under the two different policies is well summarized in Figure 5, which plots



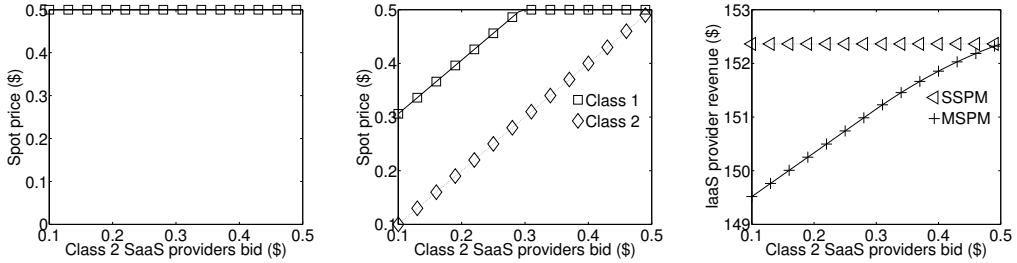
(a) SSPM strategy (b) MSPM strategy (c) IaaS provider revenue

Fig. 2: Behavior of SSPM and MSPM strategies with two classes of SaaS providers (predicted load 40 req/s)



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

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



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

Fig. 4: Behavior of SSPM and MSPM strategies with two classes of SaaS providers (predicted load 80 req/s)

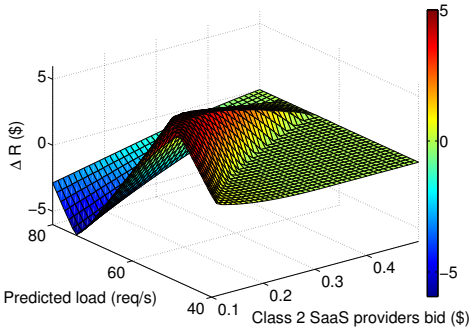
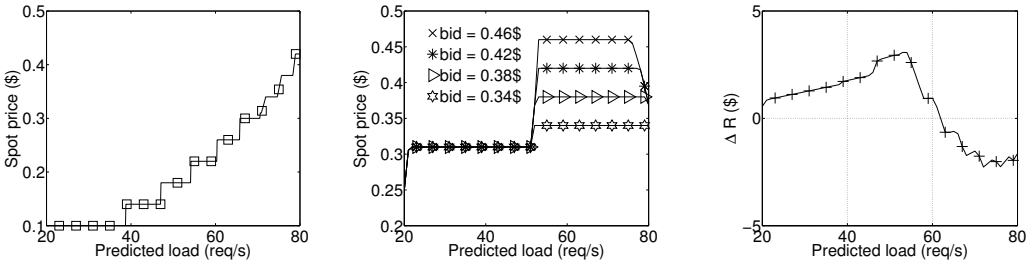


Fig. 5: Difference in the IaaS provider revenue using the MSPM and SSPM strategies with two classes of SaaS providers and an increasing predicted load.

the difference ΔR between the IaaS provider revenue under the MSPM and SSPM strategies for different values of the predicted load and Class 2 providers bids. At lower load rates, when there is a higher amount of available spot VMs, the flexibility of the MSPM strategy ensures higher revenues ($\Delta R > 0$) over SSPM. Conversely, at higher load rates, and less resource available, the IaaS provider revenue is higher under SSPM ($\Delta R < 0$). Moreover, the revenue gap increases as the scenario becomes more heterogeneous, i.e., when Class 2 SaaS providers bid is close to 0.1$.

We conclude the analysis by considering a heterogeneous scenario where all SaaS providers' bids are different, taking distinct values in the set $\{0.1, 0.14, 0.18, 0.22, 0.26, 0.30, 0.34, 0.38, 0.42, 0.46\}$ (one for each SaaS provider). The corresponding results are illustrated in Figure 6. As shown in Figure 6a, under the SSPM policy the IaaS provider sets the spot price to 0.1$ (corresponding to the lowest SaaS providers bid, that is $\sigma_1^U = 0.1\$$) for values of the predicted load lower than 40 req/s. In this interval, the resources demand is low and the IaaS provider finds more convenient to set a low price to incentivate all the SaaS providers to buy (more) VMs. As the predicted load increases above 40 req/s, so does the spot price. This is expected since, as the predicted load increases, the SaaS providers become more aggressive in resource demand, which in turn allows the IaaS provider to sell the unsold resources at higher and higher prices. It is worth observing that in this example the spot price is piecewise constant, taking values in a discrete set corresponding to the different SaaS providers' bids $(0.1\$, 0.14\$, \ldots)$: basically, the IaaS provider sets the price equal to the lowest bid among the SaaS providers still buying spot VMs (this



(a) SSPM strategy: spot price

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

(c) Difference in the IaaS provider revenue using the MSPM and SSPM strategies

Fig. 6: Strategies behavior in a heterogeneous scenario with different bids as function of the predicted load

is exactly the highest price to which all the remaining SaaS providers buy spot VMs). As the load increases, the IaaS provider can increase the price, cutting off each time one IaaS provider, and setting the new price to the lowest bid of the remaining set of customers. Eventually, at high load, the price equal the highest bid of 0.46\$, which means that all the spot VMs are sold to the SaaS provider with the highest bid.

Under the MSPM policy, the IaaS behavior significantly differs from the previous case. For those SaaS providers with a bid lower than 0.34\$, the IaaS always sets the spot price up to the maximum value (the curves for those providers are not shown in the figure). Conversely, for those SaaS providers with higher bids, the behavior varies with the predicted load. When the predicted load is lower than 50 req/s, the spot prices are constant and equal to 0.31\$ (again the price which maximizes the IaaS revenue at low load previously observed in Figures 1(c)). For higher values, the IaaS provider sets the spot prices for those customers up to their bid, thus exploiting their willingness to pay more. When the predicted load is greater than 75 req/s, the spot prices decrease again. In this case, as we have already discussed (previous experiment, Figure 4b), the SaaS providers with lower bids cannot be excluded as in SSPM, and the IaaS strategy is to decrease the spot price of SaaS providers with higher bids, so that they are willing to buy more VMs. In Figure 6c we plot the difference between the revenue under the two policies, which shows that the SSPM strategy yields higher revenue at higher load than MSPM, while at lower load it is the flexibility of the MSPM strategy to ensure higher revenues ($\Delta R > 0$).

These experiments clearly indicate that, when there are few spot VMs and demand is greater than supply, it is more profitable to sell spot VMs only to those SaaS providers that are willing to pay more. The SSPM policy, which uses a single spot VM selling price thus cutting off SaaS providers who bid less, does exactly that. Conversely, when the demand is smaller than the supply, it is more profitable to sell resources to all customers regardless of their bid, possibly adopting different spot prices. This is somehow expected, and MSPM yields indeed higher revenues to the IaaS provider.

6.2 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(s) 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 Stackelberg-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.

For the sake of comparison, we simulated a dynamic scenario using the three different strategies to study how they affect the IaaS provider revenue in 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 randomly generated in the following sets: $\mu_u = \{5, \ldots, 15\}$ req/s, $\hat{f}_u = \{3, 4, 5\}$ and $\sigma_u^U = [0.1, 0.5]\$$. 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 strategies. We run a simulation corresponding to a period of one week (168 hours).

TABLE 2: 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$

Table 2 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[11]. Nevertheless, the MSPM strategy yields 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 maximizes the amount that each SaaS provider invests 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 ones. We remark this approach results in a higher fraction of sold spot instances, which being not a reliable type of instances, can result in a overall degraded services offered 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.

The average per hour SaaS provider profit using the SSPM, MSPM, and the GNEP approach in [4] is 72.62$, 72.61$ and 74.43$, respectively. As already shown in Table 2, our approach results in a higher number of VMs bought by the SaaS providers and a corresponding higher cost. For space reasons, we do not show the profit of each SaaS provider. However, the SaaS profits decrease only by a small fraction and in some cases they even increase. This is not completely unexpected: since the number of VMs per service increases, the service response time decreases, which, in turn, given the SaaS utility function, yields higher revenues.

7 Conclusions

In this paper we proposed service provisioning and pricing strategies for a Cloud system using a game theoretical approach. We considered several SaaS providers which offer services with QoS guarantees by acquiring flat, on demand and spot VMs provided by an IaaS provider and aim at 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; in the second stage, they bid to buy spot VMs the IaaS provider offers using its spare capacity. Given the bids, the IaaS provider determines the spot price so to maximize its revenue. We proposed the SSPM and MSPM pricing strategies for the IaaS provider, which differ for either using a single spot price or multiple ones.

We modeled the second stage as a Stackelberg game where the IaaS provider plays the game leader role by setting the price(s) and the SaaS providers play as followers and compete for the spot resources. We proved the existence of the game equilibrium for both the MSPM and the SSPM and we casted the equilibrium computation as the solution of one (for the MSPM) or more (for the SSPM) suitable MPECs. The results show that when the customer demand exceeds the supply, the SSPM yields higher revenues to the IaaS provider. Conversely, when the demand does not saturate the supply, it is the MSPM pricing scheme which yields higher revenues, thanks to its flexibility in the use of different prices for different customers.

As future work, we will study distributed algorithms and suitable protocols to compute the game equilibrium in a real implementation. We will also extend and generalize the results of this paper towards two directions: by considering more general pricing schemes and by studying a scenario with multiple

11. 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.

IaaS providers from whom the SaaS providers can acquire resources.

Acknowledgments

We would like to thank Francisco Facchinei for his insightful suggestions, the Associate Editor and the anonymous referees for their helpful comments that have been instrumental in improving the quality of this paper.

References

[1] Amazon Web Services LLC, "Amazon Elastic Compute Cloud (EC2)," 2016, http://aws.amazon.com/ec2/.
[2] D. Fudenberg and J. Tirole, *Game Theory*. Cambridge, MA: MIT Press, 1991.
[3] F. Facchinei, H. Jiang, and L. Qi, "A smoothing method for mathematical programs with equilibrium constraints," *Mathematical Programming*, vol. 85, pp. 107–134, 1999.
[4] D. Ardagna, B. Panicucci, and M. Passacantando, "A game theoretic formulation of the service provisioning problem in cloud systems," in *Proc. WWW '11*, 2011, pp. 177–186.
[5] ——, "Generalized Nash equilibria for the service provisioning problem in cloud systems," *IEEE Trans. Serv. Comput.*, vol. 6, pp. 429–442, Oct. 2013.
[6] D. Ardagna, M. Ciavotta, and M. Passacantando, "Generalized Nash equilibria for the service provisioning problem in multi-cloud systems," *IEEE Trans. Serv. Comput.*, 2015, to appear.
[7] M. Hadji, W. Louati, and D. Zeghlache, "Constrained pricing for cloud resource allocation," in *Proc. IEEE NCA '11*, Aug. 2011, pp. 359–365.
[8] F. Teng and F. Magoulès, "Resource pricing and equilibrium allocation policy in cloud computing," in *Proc. IEEE CIT '10*, Jun. 2010, pp. 195–202.
[9] G. Wei, A. V. Vasilakos, Y. Zheng, and N. Xiong, "A game-theoretic method of fair resource allocation for cloud computing services," *J. Supercomput.*, vol. 54, no. 2, pp. 252–269, 2010.
[10] H. Roh, C. Jung, W. Lee, and D.-Z. Du, "Resource pricing game in geo-distributed clouds," in *Proc. IEEE INFOCOM'13*, Apr. 2013, pp. 1519–1527.
[11] R. Pal and P. Hui, "Economic models for cloud service markets: Pricing and capacity planning," *Theor. Comput. Sci.*, vol. 496, pp. 113–124, 2013.
[12] L. Tang and H. Chen, "Joint pricing and capacity planning in the IaaS cloud market," *IEEE Trans. Cloud Comput.*, 2014, to appear.
[13] U. Lampe, M. Siebenhaar, A. Papageorgiou, D. Schuller, and R. Steinmetz, "Maximizing cloud provider profit from equilibrium price auctions," in *Proc. IEEE CLOUD '12*, Jun. 2012.
[14] W. Wang, B. Liang, and B. Li, "Revenue maximization with dynamic auctions in iaas cloud markets," in *Proc. IEEE/ACM IWQoS '13*, 2013, pp. 1–6.
[15] M. M. Nejad, L. Mashayekhy, and D. Grosu, "A family of truthful greedy mechanisms for dynamic virtual machine provisioning and allocation in clouds," in *Proc. IEEE CLOUD '13*, 2013, pp. 188–195.
[16] I. Fujiwara, K. Aida, and I. Ono, "Applying double-sided combinational auctions to resource allocation in cloud computing," in *Proc. IEEE/IPSJ SAINT '10*, 2010, pp. 7–14.
[17] M. Nejad, L. Mashayekhy, and D. Grosu, "Truthful greedy mechanisms for dynamic virtual machine provisioning and allocation in clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 2, pp. 594–603, Feb. 2015.
[18] K. Xu, Y. Zhang, X. Shi, H. Wang, Y. Wang, and M. Shen, "Online combinatorial double auction for mobile cloud computing markets," in *Proc. IEEE IPCCC '14*, Dec. 2014, pp. 1–8.
[19] G. Di Modica, O. Tomarchio, P. Bonacquisto, and G. Petralia, "A procurement auction market to trade residual cloud computing capacity," *IEEE Trans. Cloud Comput.*, vol. 3, no. 3, pp. 345–357, 2015.
[20] M. Tanaka and Y. Murakami, "Strategy-proof pricing for cloud service composition," *IEEE Trans. Cloud Comput.*, vol. 4, no. 3, pp. 363–375, 2016.
[21] L. Mashayekhy, M. Nejad, and D. Grosu, "A PTAS mechanism for provisioning and allocation of heterogeneous cloud resources," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 9, pp. 2386–2399, 2014.
[22] I. Jangjaimon and N. Tzeng, "Effective cost reduction for elastic clouds under spot instance pricing through adaptive checkpointing," *IEEE Trans. Comput.*, vol. 64, no. 2, 2014.
[23] S. Yi, A. Andrzejak, and D. Kondo, "Monetary cost-aware checkpointing and migration on Amazon cloud spot instances," *IEEE Trans. Serv. Comput.*, vol. 5, no. 4, 2012.
[24] W. Voorsluys and R. Buyya, "Reliable provisioning of spot instances for compute-intensive applications," in *Proc. IEEE AINA '12*, Mar. 2012, pp. 542–549.
[25] B. Javadi, R. K. Thulasiram, and R. Buyya, "Characterizing spot price dynamics in public cloud environments," *Future Gener. Comput. Syst.*, vol. 29, no. 4, pp. 988–999, 2013.
[26] O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir, "Deconstructing Amazon EC2 spot instance pricing," *ACM Trans. Econ. Comput.*, vol. 1, no. 3, Sep. 2013.
[27] H. Xu and B. Li, "Dynamic cloud pricing for revenue maximization," *IEEE Trans. Cloud Comput.*, vol. 1, no. 2, Jul. 2013.
[28] Q. Zhang, E. Gürses, R. Boutaba, and J. Xiao, "Dynamic resource allocation for spot markets in clouds," in *Proc. USENIX Hot-ICE '11*, 2011.
[29] V. Di Valerio, V. Cardellini, and F. Lo Presti, "Optimal pricing and service provisioning strategies in cloud systems: A Stackelberg game approach," in *Proc. IEEE CLOUD '13*, 2013.
[30] N. Mankiw, *Principles of Economics*, 6th ed. South-Western Pub, 2011.
[31] F. Facchinei and C. Kanzow, "Generalized Nash equilibrium problems," *4OR: A Quarterly Journal of Operations Research*, vol. 5, pp. 173–210, 2007.
[32] F. Facchinei and J.-S. Pang, *Finite-Dimensional Variational Inequalities and Complementarity Problems, Vol. I*. New York: Springer, 2003.
[33] D. G. Luenberger and Y. Ye, *Linear and Nonlinear Programming*. New York: Springer, 2008.

Valeria Cardellini is Associate Professor in the Department of Civil Engineering and Computer Science of the University of Rome Tor Vergata. She received the Doctorate degree in computer science from the University of Rome Tor Vergata in 2001. Her research interests are in the field of distributed computing systems, with a focus on Web and Cloud systems and services. She has more than 80 publications in international conferences and journals. She has served as TPC member of conferences on Web and performance and as frequent reviewer for well-known international journals.

Valerio Di Valerio is a research fellow at the Computer Science Department of the Sapienza University of Rome. He received the master degree in Computer Engineering in 2010 and the Doctorate degree in Computer Science in 2014, both from the University of Rome Tor Vergata. His research interests concern service oriented systems, Cloud computing and underwater sensors networks, with special emphasis on modeling and performance evaluation.

Francesco Lo Presti is Associate Professor in the Department of Civil Engineering and Computer Science of the University of Roma Tor Vergata. He received the Doctorate degree in computer science from the University of Rome Tor Vergata in 1997. His research interests include measurements, modeling and performance evaluation of computer and communications networks. He has more than 70 publications in international conferences and journals. He has served as TPC member of conferences on networking and performance areas, and as reviewer for various international journals.

PROOF OF THEOREM 4

Proof: For the sake of readability, here we summarize the assumptions and the notations used in the proof of Theorem 4. For the sake of simplicity, in the proof let us assume that all SaaS providers' bids are different and let SaaS providers $u = 1, \ldots, N$ be labeled in increasing bid order, *i.e.*, $\sigma_1^U < \sigma_2^U < \ldots < \sigma_N^U$. The result can be easily generalized to the case where multiple SaaS providers bid the same value.

Denote by $I_k = (\sigma_k^U, \sigma_{k+1}^U]$, $k = 1, \ldots, N-1$ and $I_0 = [\sigma_0^U, \sigma_1^U]$, where $\sigma_0^U = \sigma^L$, the non-overlapping intervals delimited by the successive increasing bid values. If we restrict the SSPM problem to any such interval, we obtain a Stackelberg game, denoted by SSPM$_k$, which is basically SSPM with the spot price σ limited to the interval I_k and the set of SaaS providers restricted to $U_k = \{u \in U | \sigma_u^U > \sigma_k^U\} = \{k+1, \ldots, N\}$. By construction, the collection of these games is equivalent to the original SSPM game.

To establish the main results, we need to consider a slightly modified version of each problem denoted by SSPM$_k^c$, where the interval I_k is replaced by its closure $I_k^c = [\sigma_k^U, \sigma_{k+1}^U]$.

We first demonstrate the following two auxiliary results.

Lemma 1: There exists at least one Stackelberg equilibrium for the SSPM$_k^c$ game, $k = 0, \ldots, N-1$.

Proof: For SSPM$_k^c$, the IaaS problem is a restricted version of $\mathbf{SSPM\_IaaS}_{OPT}$ with the additional constraint $\sigma \in I_k^c$ and a limited subset of providers $u \in U_k$, while the associated followers optimization problem is $\mathbf{Redux\_SSPMSaaS}_{OPT}$. This problem has the same structural properties of the MSPM problem, and Corollary 2 in [3] ensures the existence of a Stackelberg equilibrium.

Denote by $\Theta_{i,k}^c(\sigma)$ the value of the SSPM$_k^c$ objective function, i.e., the IaaS revenue, for a spot price $\sigma \in I_k^c$, $k = 0, \ldots, N-1$. The following holds:

Lemma 2: $\Theta_{i,k-1}^c(\sigma_k^U) \geq \Theta_{i,k}^c(\sigma_k^U)$, $k = 1, \ldots, N-1$.

Proof: It suffices to observe that the SSPM$_k^c$ has one less SaaS provider than SSPM$_{k-1}^c$ (SaaS provider k, to be specific). Intuitively, for $\sigma = \sigma_k^U$, the amount of resources sold by the IaaS provider to the $\{k+1, \ldots, N\}$ SaaS providers can never be larger than the amount sold to the $\{k, \ldots, N\}$ SaaS providers.

More formally, in the case of SSPM$_k^c$ the overall amount $s_{I_k^c}$ of spot VMs sold at σ_k^U is:

$$s_{I_k^c} = \min \left\{ s^U, \sum_{u \in U_k} \bar{s}_u \right\},$$

where $\bar{s}_u$ represents the amount of spot VMs that each SaaS provider would need at σ_k^U, i.e., it is the solution of $\mathbf{Redux\_SSPMSaaS}_{OPT}$ replacing constraint (13) with $s_u \leq s^U$. Note that if $\sum_{u \in U_k} \bar{s}_u \geq s^U$ any allocation of spot VMs $s'_{I_k^c} < s^U$ would not be an equilibrium for the players, since there is at least one SaaS provider that can increase its revenue buying some of the remaining VMs (this is essentially because Θ_u is a strictly concave function).

In the case of SSPM$_{k-1}^c$ the overall amount $s_{I_{k-1}^c}$ of spot VMs sold at σ_k^U is:

$$s_{I_{k-1}^c} = \min \left\{ s^U, \sum_{u \in U_{k-1}} \bar{s}_u \right\}.$$

Since in the SSPM$_{k-1}^c$ game there is one more SaaS provider, it is clear that $\mathcal{S}_{I_{k-1}^c} \geq \mathcal{S}_{I_k^c}$, from which $\Theta_{i,k-1}^c(\sigma_k^U) \geq \Theta_{i,k}^c(\sigma_k^U)$ directly follows. This ends our proof.

Lemma 2 guarantees that for any interval I_k^c, $k = 1, \ldots, N-1$, the objective function at the left-hand extreme for problem SSPM$_k^c$ is not larger than the objective function of problem SSPM$_{k-1}^c$ at the right-hand extreme of interval I_{k-1}^c.

We are now ready to establish the main results, *i.e.* that there exists at least one Stackelberg equilibrium of the SSPM game. Let (σ_k^*, s_k^*) be the Stackelberg equilibrium of the subproblem SSPM$_k^c$, $k = 0, 1, \ldots, N-1$ and let σ^* denote the price corresponding to the largest revenue $\Theta^* = \max_{k=0,\ldots,N-1} \sum_{u \in \mathcal{U}} s_{u,k}^* \sigma_k^*$. We now show that (σ^*, s^*) is a Stackelberg equilibrium for SSPM. To this end, it suffices to show that the maximum Θ^* is attained in points other than the left-extremes of the intervals I_k^c, $k > 0$, which are not solutions of the original SSPM problem. We prove this by contradiction. Assume that the maximum value Θ^* is attained exclusively in correspondence of left-extreme σ_k^U of the interval $I_k^c = [\sigma_k^U, \sigma_{k+1}^U]$, $k > 0$ (observe that indeed, this is not a valid solution for SSPM, as SSPM$_k^c$ has one less SaaS provider than SSPM for $\sigma = \sigma_k^U$). Lemma 2 then ensures that (at least) the same value Θ^* is also achieved in correspondence of σ_k^U at the right-extreme of the interval $I_{k-1} = [\sigma_{k-1}^U, \sigma_k^U]$, which contradicts the hypothesis that Θ^* is attained only at the left extreme of the interval I_k^c. This ends our proof.