ABSTRACT
Cloud computing is an emerging paradigm which allows the on-demand delivering of software, hardware, and data as services. As cloud-based services are more numerous and dynamic, the development of efficient service provisioning policies become increasingly challenging. Game theoretic approaches have shown to gain a thorough analytical understanding of the service provisioning problem.

In this paper we take the perspective of Software as a Service (SaaS) providers which host their applications at an Infrastructure as a Service (IaaS) provider. Each SaaS needs to comply with quality of service requirements, specified in Service Level Agreement (SLA) contracts with the end-users, which determine the revenues and penalties on the basis of the achieved performance level. SaaS providers want to maximize their revenues from SLAs, while minimizing the cost of use of resources supplied by the IaaS provider. Moreover, SaaS providers compete and bid for the use of infrastructural resources. On the other hand, the IaaS wants to maximize the revenues obtained providing virtualized resources. In this paper we model the service provisioning problem as a Generalized Nash game, and we propose an efficient algorithm for the run time management and allocation of IaaS resources to competing SaaSs.

1. INTRODUCTION
Cloud Computing has been a dominant IT news topic over the past few years. It is essentially a way for IT companies to deliver software/hardware on-demand and for costumers to store and access data over the Internet. Cloud computing applications are generally priced on a subscription model, so end-users may pay a yearly usage fee, for example, rather than the more familiar model of purchasing software to run on desktop. The Cloud-based services are not only restricted to software applications (Software as a Service – SaaS), but could also be the platform for the development and deployment of cloud applications (Platform as a Service – PaaS) and the hardware infrastructure (Infrastructure as a Service – IaaS). In the SaaS paradigm applications are available over the Web and provide Quality of Service (QoS) guarantees to end-users. The SaaS provider hosts both the application and the data, hence the end-user is able to use and access the service from all over the world. With PaaS, applications are developed and deployed on platforms transparently managed by the Cloud provider. The platform typically includes databases, middleware and development tools, all delivered as a service through the Internet. In IaaS systems, virtual computer environments are provided as services and servers, storage, and network equipment can be outsourced by customers without the expertise to operate them.

Many Companies, e.g. Google and Amazon, are offering Cloud computing services such as Google’s App Engine and Amazon’s Elastic Compute Cloud (EC2) or Elastic Block Store (EBS). Large data centers provide the infrastructure behind the Cloud and virtualization technology makes Cloud computing resources more efficient and cost-effective both for providers and customers. Indeed, end-users obtain the benefits of the infrastructure without the need to implement and administer it directly adding or removing capacity almost instantaneously on a “pay-as-you-use” basis. Cloud providers can, on the other hand, maximize the utilization of their physical resources also obtaining economies of scale.

The development of efficient service provisioning policies is among the major issues in Cloud research. Indeed, modern Clouds live in an open world characterized by continuous changes which occur autonomously and unpredictably. In this context, game theoretic methods and approaches allows to gain a in-depth analytical understanding of the service provisioning problem. Game Theory has been successfully applied to diverse problems such as Internet pricing, flow and congestion control, routing, and networking [7]. One of the most widely used “solution concept” in Game Theory is the Nash Equilibrium approach: A set of strategies for the players constitute a Nash Equilibrium if no player can benefit by changing his/her strategy while the other players keep their strategies unchanged or, in other words, every player is playing a best response to the strategy choices of his/her opponents.

In this paper we take the perspective of SaaS providers which host their applications at an IaaS provider. Each SaaS provider want to maximize its profit while complying with QoS requirements, specified in Service Level Agreement (SLA) contracts with the end-users, which determine the revenues and penalties on the basis of the achieved performance level. The profit of the SaaS is given by the revenues from SLAs minus the cost sustained for using the re-
resources supplied by the IaaS. However, each SaaS competes with others SaaS and bid for the use of infrastructural resources supplied by the IaaS. The IaaS, in his turn, wants to maximize the revenues obtained providing the resources.

To capture the behavior of SaaSs and IaaS in this conflicting situation in which the best choice for one depends on the choices of the others, we recur to the Generalized Nash equilibrium concept [16], which is an extension of the classical Nash equilibrium [34]. In this paper the service provisioning problem will be modelled as a Generalized Nash game. We then use Game Theory results for developing an efficient algorithm for the run time management and allocation of IaaS resources to competing SaaSs.

The remainder of the paper is organized as follows. Section 2 introduces the reference system under study. In Section 3 we introduce our model based on the concept of Generalized Nash equilibrium and we prove existence. In Section 4 we provide analytical results for a simple case study, while a general solution method is proposed in Section 5. The experimental results discussed in Section 6 demonstrate the efficiency of our method. Other approaches proposed in the literature are discussed in Section 7. Conclusions are finally drawn in Section 8.

2. PROBLEM STATEMENT AND ASSUMPTIONS

As stated in the previous section, we consider SaaS providers using Cloud computing facilities to offer services, according to the IaaS paradigm. We assume that a SaaS provider offers multiple transactional Web services (WSs) and each service represents a different application.

The hosted WSs can be heterogeneous with respect to resource demands, workload intensities and QoS requirements. The set of WS applications offered by the p-th SaaS provider are denoted with $A_p$.

An SLA contract, associated with each WS application, is established between the SaaS provider and its end-users. In particular, as in other approaches [9, 10, 38], for each WS application $k \in A_p$, a linear utility function specifies the per request revenue (or penalty) $V_k = \nu_k + m_k R_k$ incurred when the end-to-end response time $R_k$ assumes a given value. The slope of the utility function is $m_k = \nu_k / R_k < 0$ and $R_k$ is the threshold that identifies the revenue/penalty region, that is, if $R_k > R_k$, the SLA is violated and the SaaS incurs in penalties.

Linear utility functions are a flexible mechanism to rank different applications (e.g., assigning higher slopes to more important applications), and allow also to implement soft constraints on response times since the SaaS goal is to keep the infrastructure working in a profitability region, i.e., to provide an average response time lower than $R_k$ looking for the trade-off between the SLA revenues and infrastructural costs [9].

Applications are hosted in virtual machines (VMs) which are dynamically instantiated by the IaaS provider. We make the simplifying assumption that each VM hosts a single WS application. Multiple VMs implementing the same WS application can also run in parallel. In that case, we further assume that the running VMs are homogeneous in terms of RAM and CPU capacity $C$ and evenly share the incoming workload.

IaaS providers usually charge software providers on an hourly basis [8]. Hence, the SaaS has to face the problem of determining every hour the optimal number of VMs for each WS class in order to maximize the net revenues.

The SaaS performs resource allocation on the basis of a prediction of future WS workloads [1, 10]. The SaaS needs also an estimate of the future performance of each VM in order to determine application average response time. In the following we model each WS class hosted in a VM as an M/G/1 queue [11], as done in [32, 30, 2] and we assume, as common among Web service containers, that requests are served according to the processor sharing scheduling discipline. Multiple VMs can run in parallel to support the same application. In that case, the workload is evenly shared among multiple instances.

For the IaaS provider we consider a model similar to Amazon EC2 [8] and we assume that the IaaS provider offers: (i) flat VMs, for which SaaS providers applies for a one-time payment (currently every one or three years) for each instance they want to reserve, (ii) on demand VMs, which allows SaaS to access computing capacity with no long-term commitments, and (iii) on spot VMs, for which SaaS providers bid and compete for unused infrastructure provider capacity.

The VM instances are charged with the on spot cost $\sigma_k$ for application $k$, which is set by the IaaS and fluctuates periodically depending on the IaaS provider time of the day energy costs and also on the supply and demand from SaaS for on spot VMs [8, 23]. Indeed, SaaS providers compete among them for the use of on spot VMs and specify the maximum cost $\sigma^U_k$ for each application $k$ they are willing to pay per instance hour. The on spot cost $\sigma_k$ is fixed by the IaaS provider which can also decide to do not allocate any spot instance to a SaaS. On the other hand, each SaaS provider is guaranteed to have access up to $f^U_k$ flat VMs he reserved by applying to the one-time payment. The number of on spot VMs available at the IaaS cloud service center is denoted by $s^U$.

For example in the Amazon case on spot costs are available via the EC2 API [8] or by third party sites [36]. On spot costs fluctuate according to the time of the day and on the Cloud site region, and could be less or greater than the time unit cost $\varphi$ for flat VMs. Finally, we denote with $\delta$ the cost for on demand instances. With the current pricing models, $\delta$ is strictly greater than $\varphi$ and we assume $\delta > \sigma^U_k$ for all $k$. Indeed, since the IaaS provider can arbitrarily terminate on spot instances from a SaaS resource pool [8], no one is willing to pay for a less reliable resource a time unit cost higher than on demand instances which provide a higher availability level.

On spot instances have been traditionally adopted to support batch computing intensive workloads [8]. However, since nowadays IaaS providers allow specifying autonomic policies which can dynamically allocate VM instances in few minutes as a reaction to failures, we advocate the use of on spot instances also to support the execution of traditional WS applications.

3. GENERALIZED NASH EQUILIBRIUM APPROACH

Hereafter we introduce the Generalized Nash Equilibrium Problem arising in the Cloud computing system under study and we prove the existence of at least an equilibrium.
## 3.1 Problem Formulation

The goal of flat SaaS provider $p$ is to determine every hour the number of flat VMs to be devoted for the execution of all WS applications $k \in A_p$, in order to maximize its profits and, at the same time, as so to to satisfy the prediction $\Lambda_k$ for the arrival rate of the WS application $k$. Let us denote with $\mu_k$ the maximum service rate for the requests of application $k$ on a VM of capacity one. If the workload is evenly shared among the VMs, then the average response time for the execution of application $k$ requests is given by:

$$E[R_k] = \frac{1}{C\mu_k - \frac{\Lambda_k}{f_k + d_k + s_k}},$$

under the assumption that VMs are not saturated, i.e. guaranteeing the equilibrium conditions for the $M/G/1$ queues $C\mu_k (f_k + d_k + s_k) - \Lambda_k > 0$.

The average per time unit revenues for application $k$ requests are given by $V_k \Lambda_k = (v_k + m_k E[R_k]) \Lambda_k$.

Considering the infrastructural costs to access flat, on demand, and on spot VM instances the goal of a SaaS provider is to maximize its profits given by:

$$\sum_{k \in A_p} \left( v_k \Lambda_k + \frac{m_k A_k (f_k + d_k + s_k)}{C\mu_k (f_k + d_k + s_k) - \Lambda_k} + \right.$$

$$\left. - \varphi f_k - \delta d_k - \sigma_k s_k \right).$$

With this setting in mind, the problem of the generic SaaS provider $p$ becomes:

$$\max_{\sigma} \theta_p = \sum_{k \in A_p} \frac{m_k A_k (f_k + d_k + s_k)}{C\mu_k (f_k + d_k + s_k) - \Lambda_k} +$$

$$- \sum_{k \in A_p} (\varphi f_k - \delta d_k - \sum_{k \in A_p} \sigma_k s_k)$$

$$\sum_{k \in A_p} \varphi f_k \leq f_p^U \quad \forall \ k \in A_p \quad (2)$$

$$f_k + d_k + s_k > \frac{\Lambda_k}{C\mu_k} \quad \forall \ k \in A_p \quad (3)$$

$$\sum_{k \in A} s_k \leq s^U \quad \forall \ k \in A_p \quad (4)$$

$$f_k, d_k, s_k \geq 0 \quad \forall \ k \in A_p \quad (5)$$

Note that the terms $\sum_{k \in A_p} v_k \Lambda_k$ can be dropped in the SaaS objective function since they are independent of the decision variables.

**Constraint (2)** entails that the flat VMs allocated to applications are less or equal to the one available, while constraint (3) guarantees that resources are not saturated. Finally constraint (4) guarantees that the on spot VMs allocated to competing SaaS providers are lower than the one available at the IaaS cloud service center $s^U$.

We would like to remark that, in the formulation of the problem, we have not imposed variables $f_k, d_k, s_k$ to be integer, as in reality they are. In fact, requiring variables to be integer makes the solution much more difficult. However, preliminary experimental results have shown that if the optimal values of the variables are fractional and they are rounded to the closest integer solution, the gap between the solution of the real integer problem and the relaxed one is very small, justifying the use of a relaxed model. We therefore decide to deal with continuous variables, actually considering a relaxation of the real problem.

On the other side, the IaaS provider’s goal is to determine the time unit cost $\sigma_k$ for on spot VM instances for all applications $k \in A_p$ and every SaaS provider $p$, in order to maximize its total revenue:

$$\max_{\sigma} \theta_I = \sum_{k \in A} (\varphi f_k + \delta d_k + \sigma_k s_k)$$

$$\sigma_k^L \leq \sigma_k \leq \sigma_k^U \quad \forall \ k \in A.$$

where $A$ denotes the set of indexes of all WS applications (i.e., $A = \cup_p A_p$, $A_p \cap A_q = \emptyset$ if $p_1 \neq p_2$).

Note that the on spot instance lower bound $\sigma_k^L$ is fixed according to the time of the day and includes the energy costs for running a single VM instance for one hour and the amortized cost of the hosting physical machine [23]. For the sake of clarity, the notation adopted in this paper is summarized in Table 1.

If the maximum time unit costs of an application is lower than the minimum set by the IaaS, i.e. $\sigma_k^L < \sigma_k^U$, formally the SaaSs and IaaS problems have no solution. In that case we can set $s_k = 0$ and consider a simplified problem where the capacity allocation for application $k$ is limited to determine the number of flat and on demand instances. Hence in the following we will always assume that $\sigma_k^L \leq \sigma_k^U$ for all $k$. Note that, if the on spot instances are terminated by the IaaS provider, then the SaaS can dynamically start the execution of an equal number of on demand instances.

In this framework, SaaS providers and the IaaS provider are making decisions at the same time, and the decisions of a SaaS depend on those of the others SaaS and the IaaS. Vice versa, IaaS objective function depends on SaaS decisions. In this setting, we cannot analyze decision in isolation, but we must ask what a SaaS would do, taking into account the decision of the IaaS and other SaaSs. To capture the behavior of SaaSs and IaaS in this conflicting situation (game) in which what a SaaS or the IaaS (the players of the game) does directly affects what others do, we consider the Generalized Nash equilibrium concept [16], which is broadly used in Game Theory and other fields. We remind the reader that the generalized Nash equilibrium problem (GNEP) differs from the classical Nash equilibrium problem (NEP) since, not only the objective functions of each player (called pay-off functions) depend upon the strategies chosen by all the other players, but also each player’s strategy set may depend on the rival players’ strategies. In our setting the constraint of each problem involving other player’s variables (joint constraint) comes from (4).

Following the Nash equilibrium concept, SaaS and IaaS providers adopt a strategy such that none of them can improve its revenue by changing its strategy unilaterally (i.e., while the other players keep their strategies unchanged). The service provisioning problem results therefore in a GNEP where the players are the SaaS providers and the IaaS provider, the strategy variables of SaaS provider $p$ are $f_k, d_k,$ and $s_k$, for $k \in A_p$, while the strategy variables of the IaaS are the costs for on spot VMs, $\sigma_k$, for all $k \in A$. Within this setting, the IaaS’s strategy is simple. In fact, if SaaS provider decide not to use on spot VMs for application $k$, that is $s_k = 0$, the value of the IaaS payoff does not depend on the choice for $\sigma_k$, that can be any feasible value. Whereas, if
\(s_k \neq 0\), regardless its value, the best response of the IaaS is to play \(s_k = \sigma_k^p\). When one player has a strategy that yields a higher revenue, no matter which choice the other players makes, that player is said to have a dominant strategy, and he will play that strategy in each of the Nash equilibria. Therefore, whenever \(s_k \neq 0\), the IaaS will play its dominant strategy \(\sigma_k^p = \sigma_k^p\).

Another important feature of the derived GNEP is that it satisfies the Convexity Assumption: the payoff functions of both SaaS providers and IaaS, are concave in its own strategy \(x\), \(p\), and \(\sigma\) and the set of strategies are convex. Moreover, even if the decision of a SaaS depends on the decisions of the other SaaSs and the IaaS, the only constraint of each problem involving other player’s variables (coming from (4) in each SaaS problem), is the same for all players: we refer to this special class of GNEP as jointly convex GNEP [18].

### 3.2 Existence of equilibria

Using the model introduced in the previous sections, we now prove that there exists an equilibrium for service provisioning on the Cloud. The proof is based on the equivalence between generalized Nash equilibria and fixed points of the best-response mapping, and on the Kakutani’s fixed point theorem [24].

To simplify the discussion we introduce the following notations. Let \(x^p = (f_k, d_k, s_k)_{k \in A_p}\) denotes the strategies vector of SaaS provider \(p\), \(x = (x^p)_{p=1}^P\), \(x^{-}\) the vector formed by the strategies of all SaaS providers different from \(p\) and \(\sigma = (\sigma_k)_{k \in A}\). Moreover we indicate by \(X_p(x^{-})\) the set of strategies for provider \(p\), and \(X_I\) the set of strategies of the IaaS provider.

**Theorem 1** There exists at least one generalized Nash equilibrium for the game.

**Proof** Let consider any SaaS provider \(p\). For any feasible strategy \(x^{-}\) of the other SaaS providers we have that \(X_p(x^{-})\) contains the set:

\[
X_p^1 := \{x^p \geq 0 : \sum_{k \in A_p} f_k \leq f_k^U, \quad s_k = 0 \quad \forall k \in A_p\}
\]

Moreover, for any feasible strategy \(\sigma\) of IaaS provider we have the following relations:

\[
\Theta_p^I(x^p) \leq \Theta_p^I(x^p, \sigma) \leq \Theta_p(x^p) \leq \Theta_p^I(x^p) \leq \Theta_p^I(x^p, \sigma),
\]

where \(\Theta_p^I(x^p)\) is:

\[
\sum_{k \in A_p} \left[ \frac{m_k \Lambda (f_k + d_k + s_k)}{C \mu_k (f_k + d_k + s_k) - \Lambda_k} - \varphi f_k - \delta d_k - \sigma_k^U s_k \right]
\]

and \(\Theta_p^I(x^p)\) is:

\[
\sum_{k \in A_p} \left[ \frac{m_k \Lambda (f_k + d_k + s_k)}{C \mu_k (f_k + d_k + s_k) - \Lambda_k} - \varphi f_k - \delta d_k - \sigma_k^L s_k \right].
\]

Each summand is like \(\frac{m \Lambda (f + d + s)}{C \mu (f + d + s) - \Lambda} - \varphi f - \delta d - \sigma\), and its concavity follows by direct evaluation of the Hessian:

\[
\begin{bmatrix}
m \Lambda (f + d + s) & 1 & 1 \\
C \mu (f + d + s) - \Lambda & 1 & 1 \\
\end{bmatrix}
\]

The eigenvalues are zero and \(s > \frac{\Lambda}{C \mu}\), which is negative since \(m < 0\) and \(f + d + s \geq \frac{\Lambda}{C \mu}\). We remark that if \(f_k + d_k + s_k = \frac{\Lambda_k}{\mu_k}\) for some \(k \in A_p\), then \(\Theta_p^I(x^p) \to -\infty\) and if \(d_k \to +\infty\) then \(\Theta_p^I(x^p) \to -\infty\) as well.

If we denote by \(M^p := \max_{x^p \in X_p^1} \Theta_p^I(x^p)\), then the set:

\[
\overline{X}_p := \{x^p : \Theta_p^I(x^p) \geq M^p\}
\]

is nonempty, convex and compact. Therefore, for any feasible \(x^{-}\) and \(\sigma\) we obtain that:

\[
\max_{x^p \in X_p(x^{-}, \sigma)} \Theta_p(x^p, \sigma) \geq \max_{x^p \in X_p^1} \Theta_p^I(x^p) = M^p
\]

thus:

\[
\arg \max_{x^p \in X_p(x^{-}, \sigma)} \Theta_p(x^p, \sigma) \subseteq \overline{X}_p,
\]

that is the sets of best responses of player \(p\) to the strategies of the rivals are uniformly bounded by \(\overline{X}_p\).

Finally, we consider the convex compact set

\[
\overline{X} := \overline{X}_1 \times \ldots \overline{X}_p \times \overline{X}_I
\]

and the best response set-valued mapping \(B\) defined as follows:

\[
B(x, \sigma) := \arg \max_{y^p \in \overline{X}_p(x^{-}, \sigma)} \Theta_1(y^1, \sigma) \times \ldots \times \arg \max_{y^p \in \overline{X}_p(x^{-}, \sigma)} \Theta_P(y^P, \sigma)
\]

From the above discussion it follows that the set valued map \(B : \overline{X} \rightrightarrows \overline{X}\) has nonempty and convex values, and its graph is closed by continuity of payoff functions. Therefore, by Kakutani’s theorem there exists a fixed point of \(B\), that is a strategy \((x, \sigma) \in B(x, \sigma)\), which is a generalized Nash equilibrium of the game.

### 4. A SINGLE APPLICATION CASE STUDY

In order to gain insight into the properties of the equilibria in our setting, let us focus on the case of a single SaaS provider with a single application class.

In the following we will also assume that the IaaS is over-provisioned and there is no an upper bound \(s^U\) on the number of on spot VM instances available and, hence, the constraint (4) is relaxed. Indeed, it is not reasonable that a single SaaS will be able to saturate the on spot instances capacity available in a real system. In that case, each player’s strategy (SaaS and IaaS) belong to a set which is fixed and does not depend on the rival players’ strategies: hence the GNEP reduces to a NEP which is much more simple to solve.

The aim of the SaaS provider, given the IaaS strategy \(\sigma\), is to choose \(f\), \(d\), and \(s\) that maximize the payoff:

\[
\Theta_S = \frac{m \Lambda (f + d + s)}{C \mu (f + d + s) - \Lambda} - \varphi f - \delta d - \sigma
\]

over the set \(X_S = \{(f, d, s) \in \mathbb{R}_+^3 : f + d + s > \frac{\Lambda}{C \mu}, \ f \leq f^U\}\). The aim of the IaaS provider, given the SaaS strategies \((f, d, s)\), is to choose \(\sigma\) that maximize the payoff:

\[
\Theta_I = \varphi f + \delta d + \sigma
\]

over the set \(X_I = \{\sigma \in \mathbb{R} : \sigma^L \leq \sigma \leq \sigma^U\}\).

An analytic study can be obtained, as shown in Appendix A, writing down the KKT conditions for the SaaS and the IaaS optimization problems and concatenating them. We
can observe that the Nash equilibria depend on the application workload conditions and on the relation between the cost \( \varphi \) of the flat VMs and the upper bound \( \sigma^U \) for on spot instances cost. The obtained equilibria and the corresponding value of the payoff functions \( \Theta_S \) and \( \Theta_I \) are reported in Table 2.

The first important remark we get is that, in general, there is no a unique Nash equilibrium (e.g., when \( \varphi = \sigma^U \)). However, if multiple equilibria exist they are equivalent, that is they are characterized by the same values of the payoffs. Furthermore, assuming \( \delta > \sigma^U \) for all \( k \) we always have \( d = 0 \) (i.e., the SaaS provider always adopt flat or on spot VMs which are always cheaper than on demand instances).

When \( \varphi \geq \sigma^U \) (Table 2, first and second rows) then it is more convenient for the SaaS provider to use only on spot instances which number can be obtained by a closed formula under any workload conditions. Remind that the value of \( \sigma^L \) is set by the IaaS provider, and SaaS provider can not set the value of \( \sigma^U \). In these hypotheses, the SaaS provider is under heavy load conditions and requests cannot be executed by using only the flat VMs. The SaaS provider will use all of its flat VMs and will buy on spot VMs at the maximum cost. Again, the number of on spot VMs can be determined by a closed formula (see Table 2, third row). Since \( \varphi < \sigma^U \), in these conditions the payoff \( \Theta_S \) is greater than the value obtained when \( \varphi \geq \sigma^U \) (Table 2 first and second rows). Hence these conditions are more profitable for the SaaS provider which determines the optimal trade-off between the cost of the flat and on spot VMs and the revenues which can be achieved by the SLA contract. With the same arguments we can derive that, vice versa, the payoff \( \Theta_I \) is lower than the value obtained when \( \varphi \geq \sigma^U \), hence these conditions are less profitable from the IaaS point of view even if on spot instances are sold at the maximum cost.

Let us now examine the case when \( \varphi < \sigma^U \) and \( f^U > \frac{A}{\sigma^U} \). The SaaS provider is under light load conditions and, therefore, the incoming requests could be executed by using only the flat VMs. Under these hypotheses, the equilibria are set according to the marginal value of the payoff function of the SaaS player evaluated in \( f = f^U \) and \( s = 0 \), that is \( \frac{-m^2 A^2}{(c_1 + \frac{1}{\sigma^U \mu^2})^2} \). This marginal value represents the change in the revenues of SaaS per unit change in \( s \), i.e., the increase in the SaaS provider revenues deriving from SLA contracts obtained providing better performance to end-users by adopting an additional on spot VM. We identify two cases:

- If \( \varphi < \sigma^U < \frac{-m^2 A^2}{(c_1 + \frac{1}{\sigma^U \mu^2})^2} \) (see Table 2, fourth row), the marginal value is greater than the cost of individual on spot instance sold at the maximum cost (\( \sigma^U \)), therefore it is convenient for the SaaS provider to buy on spot instances. Note that under these conditions, the payoff \( \Theta_S \) is greater than the value obtained when \( \varphi \geq \sigma^U \) as in the previous case. Vice versa, for the IaaS provider these circumstances are less profitable.

- If \( \varphi < \sigma^U \) and \( \frac{-m^2 A^2}{(c_1 + \frac{1}{\sigma^U \mu^2})^2} \leq \sigma^U \) (see Table 2, fifth and sixth row), then the marginal value is less or equal to the cost of on spot VMs and hence the SaaS provider has no incentives to buy additional on spot VMs. Also in these cases, the equilibrium value of the payoff of SaaS provider is greater than the one obtained when \( \varphi \geq \sigma^U \), while this again does not happen for the IaaS provider. Finally, it is worth noticing that if \( \frac{-m^2 A^2}{(c_1 + \frac{1}{\sigma^U \mu^2})^2} < \varphi \) the SaaS provider adopts a number of VMs lower than \( f^U \). Hence, in that case the costs incurred by using a larger number of flat VMs cannot be counterbalanced by the revenues which can be obtained by improving application performance.

### Table 1: Parameters and decision variables.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_k )</td>
<td>Number of flat VMs used for application ( k )</td>
</tr>
<tr>
<td>( d_k )</td>
<td>Number of on demand VMs used for application ( k )</td>
</tr>
<tr>
<td>( s_k )</td>
<td>Number of on spot VMs used for application ( k )</td>
</tr>
<tr>
<td>( \sigma_k )</td>
<td>Time unit cost for on spot VMs used for application ( k )</td>
</tr>
</tbody>
</table>

### Decision Variables

- \( f \) Set of applications of the \( p \) SaaS providers
- \( d \) Set of applications of all the SaaS providers
- \( f^U \) Maximum number of flat computational resources IaaS can provide for provider \( p \)
- \( s^U \) Maximum number of on spot computational resources IaaS can provide for all the SaaS providers
- \( \Lambda \) Capacity of computational resources
- \( \mu_k \) Maximum service rate of a capacity 1 server for executing class \( k \) application
- \( m_k \) Application \( k \) utility function slope
- \( \varphi \) Time unit cost for flat VMs
- \( \delta \) Time unit cost for on demand VMs
- \( \sigma^L_k \) Minimum time unit cost for on spot VMs used for application \( k \), set by the IaaS provider
- \( \sigma^U_k \) Maximum time unit cost for on spot VMs used for application \( k \), set by the SaaS provider

### System Parameters

- \( n \) number of SaaS providers
- \( A_p \) Set of applications of the \( p \) SaaS provider
- \( A \) Set of applications of all the SaaS providers
- \( f^U \) Maximum number of flat computational resources IaaS can provide for provider \( p \)
- \( s^U \) Maximum number of on spot computational resources IaaS can provide for all the SaaS providers
- \( \Lambda \) Capacity of computational resources
- \( \mu_k \) Maximum service rate of a capacity 1 server for executing class \( k \) application
- \( m_k \) Application \( k \) utility function slope
- \( \varphi \) Time unit cost for flat VMs
- \( \delta \) Time unit cost for on demand VMs
- \( \sigma^L_k \) Minimum time unit cost for on spot VMs used for application \( k \), set by the IaaS provider
- \( \sigma^U_k \) Maximum time unit cost for on spot VMs used for application \( k \), set by the SaaS provider

5. GENERAL SOLUTION METHOD

In the previous sections, we showed that the service provisioning problem in a Cloud Computing environment rep-
SaaS Equilibrium and value | IaaS Equilibrium and value
---|---
$\varphi > \sigma^U$ | $f = 0 \quad d = 0 \quad s = \frac{\Lambda}{C_p} \left(1 + \sqrt{\sigma^U} + \sqrt{-m}\right)$
$\Theta_S = -\frac{\Lambda}{C_p} (\sqrt{\sigma^U} + \sqrt{-m})^2$ | $\sigma = \sigma^U$
$\sigma^U$ | $f = f^U \quad d = 0 \quad s \geq 0$
$\Theta_S = -\frac{\Lambda}{C_p} (\sqrt{\sigma^U} + \sqrt{-m})^2$ | $\sigma = \sigma^U$
$\varphi < \sigma^U$ | $f = f^U \quad d = 0 \quad s = \frac{\Lambda}{C_p} \left(1 + \sqrt{\sigma^U} + \sqrt{-m}\right)$
$\Theta_S = -\frac{\Lambda}{C_p} (\sqrt{\sigma^U} + \sqrt{-m})^2 + f^U (\sigma^U - \varphi)$ | $\sigma = \sigma^U$

| $f^U < \frac{\Lambda}{C_p}$ | $f = f^U \quad d = 0 \quad s = \frac{\Lambda}{C_p} \left(1 + \sqrt{\sigma^U} + \sqrt{-m}\right)$
$\Theta_S = -\frac{\Lambda}{C_p} (\sqrt{\sigma^U} + \sqrt{-m})^2 + f^U (\sigma^U - \varphi)$ | $\sigma = \sigma^U$

| $\varphi < \sigma^U$ | $f = f^U \quad d = 0 \quad s = 0$
$\Theta_S = -\frac{\Lambda}{C_p} (\sqrt{\varphi} + \sqrt{-m})^2$ | max$(\sigma^L, \varphi) \leq \sigma \leq \sigma^U$
$\Theta_T = \frac{\Lambda}{C_p} (\sqrt{-m} \varphi + \varphi)$ | $\Theta_T = \frac{\Lambda}{C_p} (\sqrt{-m} \varphi + \varphi)$

| $\varphi < \sigma^U$ | $f = f^U \quad d = 0 \quad s = 0$
$\Theta_S = \frac{m \Lambda^2}{(C_p f^U - \Lambda)^2} \varphi$ | max$(\sigma^L, \varphi) \leq \sigma \leq \sigma^U$
$\Theta_T = \frac{m \Lambda^2}{(C_p f^U - \Lambda)^2} \varphi$ | $\Theta_T = \frac{m \Lambda^2}{(C_p f^U - \Lambda)^2} \varphi$

We remark that the GNEP has usually multiple or even infinitely many solutions and it is not true that any solution of the jointly convex GNEP is also a solution of the VI. A solution of the jointly convex GNEP which is also a solution of the VI is called a variational equilibrium. Our approach is to calculate, among all the equilibria, a variational equilibrium. In fact, a variational equilibrium is more socially stable than other equilibria (see [14]) and it is a valuable target for an algorithm: there are plenty of methods for solving VI problems. In our setting we have:

$$F = \begin{bmatrix}
\frac{m_1 \Lambda^2}{(C \mu_1 (f_1 + d_1 + s_1) - \Lambda^2)^2} + \varphi \\
\frac{m_1 \Lambda^2}{(C \mu_1 (f_1 + d_1 + s_1) - \Lambda^2)^2} + \delta \\
\frac{m_1 \Lambda^2}{(C \mu_1 (f_1 + d_1 + s_1) - \Lambda^2)^2} + \sigma_1 \\
\cdots \\
\cdots \\
\cdots \\
\cdots \\
\frac{m_1 \Lambda^2}{(C \mu_1 (f_1 + d_1 + s_1) - \Lambda^2)^2} + \sigma_1 \\
\end{bmatrix}$$

whose Jacobian is:

$$JF = \begin{bmatrix}
a_{11} & a_{11} & a_{11} \\
a_{11} & a_{11} & a_{11} \\
\cdots & \cdots & \cdots \\
0 & a_{1|A|} & a_{1|A|} \\
0 & a_{1|A|} & a_{1|A|} \\
0 & a_{1|A|} & a_{1|A|} \\
0 & a_{1|A|} & a_{1|A|} \\
\end{bmatrix}$$

where the generic term $a_{ij} = -\frac{m_1 \Lambda^2}{(C \mu_1 (f_1 + d_1 + s_1) - \Lambda^2)^2} > 0$, and $B$ is a matrix of dimension $3|A| \times |A|$. Its symmetric part is:
Therefore, in the feasible set X, its non-zero eigenvalues (i.e., \(3 \lambda_1, \cdots, 3 \lambda_3\)) are positive (being \(m < 0\) and \(f + d + s > \frac{\lambda_3}{\lambda_1}\)), and hence F results to be monotone \(^3\) (not strictly). Hence we can calculate a variational equilibrium of the GNEP associated to the service provisioning problem by solving a monotone VI. Concerning the solution methods for monotone VI, among the simplest ones, there are the projection type methods. These methods found on the well known fixed point reformulation of a VI:

\(x^* \in X\) solves \(VI(X, F)\) iff

\[ x^* = \text{Proj}_X(x - \alpha F(x^*)) \] for any \(\alpha > 0,\)

where \(\text{Proj}_X\) denotes the orthogonal projection onto \(X\).\(^4\)

The important feature of these methods is that there is not need of using derivative of \(F\) and \(\beta\) they do not involve any computation besides the function evaluation and the projection onto \(X\). When the projection is easily computable, as is our case, projection methods are extremely simple and cheap. Among the projection type methods we consider the hyperplane projection method [22], where two projection per iteration are executed. The method is simple and admits a geometric interpretation. Given the current point \((x^t, \sigma^t)\), first we compute \(\text{Proj}_X((x^t, \sigma^t) - \beta_t F(x^t, \sigma^t))\) and then we search on the line segment between those points, for \((y^t, \delta^t)\) such that the hyperplane

\[ \{(x, \sigma) : (F(y^t, \delta^t), (x, \sigma) - (y^t, \delta^t)) = 0\} \]

strictly separates \((x^t, \sigma^t)\) for any solution of the problem.

The next iterate \((x^{t+1}, \sigma^{t+1})\) is computed by projecting \((x^t, \sigma^t)\) onto the hyperplane and then onto \(X\). The scheme is formally stated in Figure 1. We set as initial solution of the algorithm \(x^0\) the one obtained by the best reply among SaaS according to the Gauss-Siedel scheme [18] and setting \(\sigma^0 = \sigma^U\) for all \(k\).

We emphasize that constructing the hyperplane requires a single projection onto the feasible set and employs an Armijo-type line search. During the line search, moreover, no projection onto \(X\) are required, but only function evaluations. Regarding the choice of the parameters, following [22] we take \(\beta_{t+1} = \text{median}(\beta, \theta \alpha_t \beta_t)\), where \(\theta > 1\) but not too large, e.g., \(\theta = 2\).

The algorithm for the VI solution can be executed efficiently by the IaaS, under the assumption that the SaaSs provide to the IaaS also the prediction of the incoming workload \(\Lambda_t\) for the next hour. Indeed, for the problem under analysis the SaaS utility function slopes are advertised to the cloud end user and hence are known and every SaaS has to state its bit \(\sigma^U_k\) to the IaaS.

\(^3\)F is monotone on \(X\) if, for any \(x, y \in X\), it holds that \(<F(x) - F(y), x - y> \geq 0.\)

\(^4\)\(\text{Proj}_X(z) = \arg\min\{\|w - z\|, w \in X\} \quad \forall z \in \mathbb{R}^n.\)

**STEP 0.** Select parameters \(\xi \in (0, 1), \beta, \beta_t \) s.t. \(0 < \beta \leq \beta_t\) and a sequence \(\{\beta_t\} \subset [\beta, \beta_t]\). Let \((x^0, \sigma^0) \in X\) and set \(t = 0\).

**STEP 1.** Compute \((x^t, \sigma^t) = (x^t, \sigma^t) - \beta_t F(x^t, \sigma^t)\)

**STEP 2.** If \((x^t, \sigma^t) = \text{Proj}_X((x^t, \sigma^t))\) then STOP

**STEP 3.** Set \(j(t)\) the minimum \(j \in \{1, 2, 3\} \) such that the hyperplane

\[ \langle F(y^t, \delta^t), (x, \sigma) - (y^t, \delta^t) \rangle = \xi \| (x, \sigma) - \text{Proj}_X((x^t, \sigma^t)) \|^2. \]

Let \(\alpha_t = 2^{-j(t)}\) and

\[ \langle (y^t, \delta^t), (x^t, \sigma^t) \rangle = \gamma_t \left( F(y^t, \delta^t), (x^t, \sigma^t) - (y^t, \delta^t) \right). \]

**STEP 4.** Set \(t = t + 1\) and go to STEP 1.

**Figure 1:** Algorithm for VI.

### 6. EXPERIMENTAL RESULTS

The resource management algorithm proposed has been evaluated for a variety of system and workload configurations. The application performance parameters have been varied as considered in the literature (see e.g. [20, 9] and references therein). Cloud providers time unit costs have been varied according to the commercial fees currently adopted [8]. Section 6.1 is devoted to quantitatively analyse the single application case study presented in Section 4. Section 6.2 illustrates the variational equilibria properties on a medium size system. Finally, the scalability of the algorithm reported in Figure 1 is discussed in Section 6.3.

#### 6.1 Single Application Analysis

For the numerical analysis reported in this Section we set \(\Lambda = 10\) req/sec, \(C = 1\), \(\mu = 1\) req/sec, and \(m = -1\). Figures 2 and 3 report the plots of the SaaS and IaaS payoff functions \(\Theta_S\) an \(\Theta_I\) where we set \(\sigma^U = 0.095\), and we varied \(\varphi\) and \(f^U\) under the assumption that \(f^U \leq \Lambda/(C\mu)\) (i.e., which corresponds to rows 1-3 of Table 2). Under this hypothesis the SaaS provider is under heavy load conditions since he cannot serve the overall incoming workload by using only his flat resources. The plots show that under the condition \(\varphi > \sigma^U\) the payoff functions are constant, while when \(\varphi < \sigma^U\) \(\Theta_S\) (\(\Theta_I\)) increases (decreases) linearly with \(f^U\).

Figures 4 and 5 plot \(\Theta_S\) and \(\Theta_I\) as function of \(\sigma^U\) (Table 2 rows 4-6) under light workload conditions for the SaaS \((\Lambda < C\mu f^U)\) where we set \(\varphi = 0.038\) and \(f^U = 50\). In this case the behaviour of the payoff functions changes crossing the marginal value which with the considered setting is equal to 0.06258: When \(\sigma^U\) is greater than the marginal value both \(\Theta_S\) and \(\Theta_I\) are constant, while increase for lower values of \(\sigma^U\). Indeed, the SaaS provider acquires additional on spot instances to profitably serve incoming end user requests, while the IaaS obtains higher revenues selling on spot instances.
6.2 Equilibria Sharing Analysis

The aim of this Section is to analyse how the on spot VMs are shared among competing SaaS changing the game parameters. The analysis results have been obtained by the algorithm described in Section 5. In particular we considered two SaaS offering five heterogeneous applications each. If not differently stated we set $s_U = 40$, $C = 1$, $\varphi = 0.03$, $\delta = 0.1$, $f_p = 20$ ($p \in \{1, 2\}$), $\Lambda_k = 1$ req/s, $\mu_k = k$ req/s, $m_k = -1$, $\sigma^L_k = 0.038$, and $\sigma^U_k = 0.09$ for all $k \in \{1, 10\}$. In the following we will vary one parameter at the time for the first application $k = 1$, while the parameters of the remaining ones will be held fixed. Figures 6-9 show how the number of resources devoted to the first application (in terms of flat, on demand, and on spot instances) and the overall capacity allocated to the remaining classes change as a function of the varying parameter. In particular, in Figure 6 the incoming workload $\Lambda_1$ varies between 1 and 14 req/s. As the Figure shows, all of the on spot instances ($s_U = 40$) available at the IaaS are always used but, as the workload increases, they are migrated from the other applications to application 1. In order to profitably sustain the workload, the number of flat instances used is also increased, but on demand VMs are not used until $\Lambda_1$ reaches 11 req/s. When $\Lambda_1$ is further increased the system starts allocating on demand VMs which are more expensive but are needed to serve the incoming requests. In general the resource allocation trends are linear with $\Lambda_1$, the discontinuities in the plots are due to the fact that the equilibrium is not unique and hence the same performance and revenues can be obtained with multiple values of $(f_k, d_k, s_k)$.

Figure 7 shows the resource sharing at the equilibrium changing the slope of application 1 utility function (which has been varied in the range $[-15, -1]$). As in the previous analysis, the on spot capacity is migrated to application 1 which becomes more sensible to response time variations and hence requires additional capacity. However, in this case the adoption of on demand instances is never profitable.

Figure 8 analyses how the variational equilibrium changes by varying application 1 maximum service rate (the range $[0.05, 1]$ req/s has been considered). If the maximum service rate increases the service time required to process each application 1 request decreases and the overall capacity required to process application 1 decreases accordingly. Hence, in this case on spot instances are migrated from application 1 to the other classes and on demand instances are used only when application 1 requests are very CPU intensive ($\mu_1 < 0.1$ req/s).

Finally, Figure 9 shows how the equilibrium changes by varying the maximum time unit cost for application 1 ($\sigma^U_1$ has been varied in the range $[0.1, 1]$; we set $\varphi = 0.038$, while for the remaining classes $\sigma^L_k = 0.018$ and $\sigma^U_k = 0.028$). As $\sigma^U_1$ increases the number of on spot VMs allocated to
application 1 decreases since the IaaS set \( \sigma_1 = \sigma_1^U \) and the SaaS provider can use in a more cost efficient way the on spot VMs to serve his remaining applications, while application 1 is supported by flat instances. Also in this scenario on demand VMs are never used and the trends are linear. This is very unintuitive, since increasing the maximum time unit cost one is willing to pay for a given application implies that the number of on spot instances devoted to the same application is reduced.

Figure 6: Resource allocation with varying application 1 incoming workload.

Figure 7: Resource allocation with varying application 1 utility function slope.

6.3 Scalability Analysis

To evaluate the scalability of our resource allocation algorithm we have considered a very large set of randomly generated instances. All tests have been performed on VMWare virtual machine based on Ubuntu 9.10 server running on an Intel Nehalem dual socket quad-core system with 32 GB of RAM. The virtual machine has a physical core dedicated with guaranteed performance and 4 GB of memory reserved. MINOS 5.51 has been used as non linear optimization solver.

The number of SaaS provider has been varied between 10 and 80, the number of applications (evenly shared among SaaSs) between 100 and 800.

The performance parameters of Web applications and infrastructural resources costs have been randomly generated uniformly in the ranges reported in Table 3 as in other literature approaches [29, 2, 9] and according to commercial fees applied by IaaS cloud providers [8].

Table 4 reports, for problem instances of different sizes, the average computational time in seconds as well as the average number of iterations performed by the algorithm reported in Figure 1 from the initial best reply solution (the means are computed on ten different runs). Since problems with a size comparable with real systems [35] including thousands of VM instances and hundreds of SaaS providers can be solved in less than one hour, our approach can be used to support the run time management of real cloud infrastructures.

7. RELATED WORK

The recent development of Cloud systems and the rapid growth of the Internet have led to a remarkable development in the use of the Game Theory tools. Problems arising in the ICT industry, such as resource or quality of service allocation problem, pricing, and load shedding, can not be handled with classical optimization approaches. Indeed, in a pure optimization approach the goal of, in general, a complex system does not depend on the interrelationships among different users, or players [7]. However, interaction across different players is non-negligible: each player can be affected by the actions of all players, not only her own action. In this setting, a natural modelling framework involves seeking an equilibrium, or stable operating point for the system. More precisely, each player seeks to optimize his own goal, which depends on the strategies of the other players upon his own, and this optimization is performed simultaneously by different players. An equilibrium (in the sense of Nash) is reached when no player can decrease his objective function by changing unilaterally its strategy.
Exe. Time (s)

Table 3: Performance parameters and time unit cost ranges.

<table>
<thead>
<tr>
<th>System</th>
<th>Exe. Time (s)</th>
<th>N-It.</th>
</tr>
</thead>
<tbody>
<tr>
<td>N-SaaS,N-Appl.</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>10,100</td>
<td>322</td>
<td>20</td>
</tr>
<tr>
<td>20,200</td>
<td>748</td>
<td>32</td>
</tr>
<tr>
<td>30,300</td>
<td>1051</td>
<td>47</td>
</tr>
<tr>
<td>40,400</td>
<td>1627</td>
<td>51</td>
</tr>
<tr>
<td>50,500</td>
<td>1922</td>
<td>37</td>
</tr>
<tr>
<td>60,600</td>
<td>2127</td>
<td>42</td>
</tr>
<tr>
<td>70,700</td>
<td>2966</td>
<td>53</td>
</tr>
<tr>
<td>80,800</td>
<td>3450</td>
<td>38</td>
</tr>
</tbody>
</table>

Table 4: VI algorithm average execution time and number of iterations.

<table>
<thead>
<tr>
<th>System</th>
<th>Exe. Time (s)</th>
<th>N-It.</th>
</tr>
</thead>
<tbody>
<tr>
<td>N-SaaS,N-Appl.</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>10,100</td>
<td>322</td>
<td>20</td>
</tr>
<tr>
<td>20,200</td>
<td>748</td>
<td>32</td>
</tr>
<tr>
<td>30,300</td>
<td>1051</td>
<td>47</td>
</tr>
<tr>
<td>40,400</td>
<td>1627</td>
<td>51</td>
</tr>
<tr>
<td>50,500</td>
<td>1922</td>
<td>37</td>
</tr>
<tr>
<td>60,600</td>
<td>2127</td>
<td>42</td>
</tr>
<tr>
<td>70,700</td>
<td>2966</td>
<td>53</td>
</tr>
<tr>
<td>80,800</td>
<td>3450</td>
<td>38</td>
</tr>
</tbody>
</table>

In [3] the authors consider centralized and decentralized load balancing strategies in a system with multiple and heterogeneous processor sharing servers. Each server has an associated service capacity and a holding cost per unit time. The requests arrive as a Poisson process, and the service time of incoming jobs is assumed to be known. For such systems, the load balancing problem is investigated in two different scenarios: (i) a centralized setting leading to a global optimization problem, in which a dispatcher decides where each job will get service so as to minimize the weighted mean number of jobs in the system, and (ii) a distributed non-cooperative setting leading to a non-cooperative game transformed into a standard convex optimization problem. The paper studies structural properties of both strategies, and the efficiency loss (or price of anarchy, PoA, which measures the inefficiency of a decentralized scheme) of the decentralized scheme relative to the global optimal (centralized) one.

In [42] the authors propose a pricing mechanism for allocation capacity in a utility computing system among competing end-users requests. The fixed available service capacity is allocated among the different flows proportionally to their monetary bids. The paper studies the resulting equilibrium point, establishes convergence of a best-response algorithm, and bounds the efficiency loss (PoA) of this distributed mechanism. More precisely: end-users requests are represented as job flows in a controlled queueing system. These jobs arrive to the system through a fixed, random process, are stored in a buffer, and then are serviced by the resource in a first come, first served manner. The service rate is set through a proportional share mechanism. Within this framework, the interactions between end-users are modelled as a game. Then, using ideas from Game Theory, authors show that the equilibrium can be reached in a distributed, asynchronous manner. The paper also reports the sensitivity analysis with respect to the variation of problem’s parameters (e.g., load intensity and relative importance of the competing user requests). Differently from our point of view, in [42] the problem of the capacity allocation is considered for a single virtualized server among competing user requests, while in this paper we consider the infrastructure data center at a higher granularity (i.e., VMs).
8. CONCLUSIONS
We proposed a game theory based approach for the run time management of a IaaS provider capacity among multiple competing SaaSs. The model includes infrastructural costs and revenues deriving from cloud end-users which depend on the achieved level of performance of individual requests. Future work will validate of our solution by performing experiments in real cloud environments. Furthermore, a comparison with the heuristic solutions adopted by SaaS and IaaS providers for the run time cloud management will be also performed.

9. REFERENCES


### Appendix A

Since each player has to solve a convex optimization problem, we know that Nash equilibria of the game are the solutions of the KKT system associated to the game. If we denote with $\alpha_d$, $\alpha_f$, $\alpha_V$, $\alpha_s$, $\lambda_d$, and $\mu$, the multipliers relative to constraints $d \geq 0$, $0 \leq f \leq f_U$, $s \geq 0$, and $\lambda_d \leq \sigma \leq \sigma_U$, respectively, then the KKT system of the game is as follows:

\[
\begin{align*}
-\delta - \frac{m \Lambda^2}{(C \mu (d + f + s) - \Lambda)^2} + \alpha_d &= 0 \\
-\varphi - \frac{m \Lambda^2}{(C \mu (d + f + s) - \Lambda)^2} + \lambda_f - \lambda_U &= 0 \\
-\sigma - \frac{m \Lambda^2}{(C \mu (d + f + s) - \Lambda)^2} + \alpha_s &= 0 \\
\quad + s + \alpha_{\sigma} - \sigma_U &= 0 \\
\quad + \alpha_d - \lambda_d &= 0 \\
\quad + \lambda_f - f &= 0 \\
\quad + \alpha_s - s &= 0 \\
\quad + \alpha_U (\sigma - \sigma_U) &= 0 \\
\quad + \alpha_V (\sigma_U - \sigma) &= 0 \\
\quad + \alpha_d, \alpha_f, \alpha_V, \alpha_s, \alpha_{\sigma}, \sigma_U &\geq 0 \\
(f, s) &\in X, \quad \sigma \in X_I
\end{align*}
\]

Since $\alpha_d$ and $\alpha_U$ can not be both positive, and $\alpha_{\sigma} = s + \alpha_s \geq \alpha_s$, we obtain $\lambda_f = 0$ and $\alpha_d = s$. Moreover from the first and third equation we obtain $\alpha_d \geq \alpha_s - \alpha_s = \delta - \sigma \geq \delta - \sigma_U > 0$, thus $d = 0$. From the second and third equation we have:

\[
0 = -\varphi + \alpha_s - \lambda_d - \alpha_f - \lambda_f \leq -\varphi + \alpha_f + \lambda_f - \lambda_f \leq -\varphi + \sigma_U + \lambda_f - \lambda_f.
\]

Let us consider three cases: $\varphi > \sigma_U$, $\varphi = \sigma_U$, and $\varphi < \sigma_U$.

1. If $\varphi > \sigma_U$, then $\alpha_f > \alpha_f$. Hence we deduce that $\alpha_f = 0$, $\alpha_f > 0$, $f = 0$, $s > 0$, $\sigma = \sigma_U$, $\alpha_s = 0$. Moreover we find $s = \frac{\Lambda}{C \mu} \left( \sqrt{1 - \frac{m \Lambda^2}{(C \mu f - \Lambda)^2}} \right)$.

2. If $\varphi = \sigma_U$, then we prove by contradiction that $\alpha_f = 0$. In fact, if $\alpha_f > 0$, then we obtain $\alpha_f = 0$, $f = 0$, $s > 0$, $\sigma = \sigma_U$, $\alpha_s = 0$. Since $\alpha_f \geq \alpha_f$, we have $\alpha_f = 0$. Therefore we find $f + s = \frac{\Lambda}{C \mu} \left( 1 + \sqrt{\frac{m \Lambda^2}{(C \mu f - \Lambda)^2}} \right)$. Moreover, we have:

\[
\sigma_U = \varphi = \frac{-m \Lambda^2}{(C \mu (f + s) - \Lambda)^2} = \sigma - \alpha_s \leq \sigma \leq \sigma_U,
\]

thus $\alpha_s = 0$ and $\sigma = \sigma_U$.

3. If $\varphi < \sigma_U$, then similarly to the previous case we obtain $\alpha_f = 0$. We now consider two cases: $f_U \leq \frac{\Lambda}{C \mu}$ and $f_U > \frac{\Lambda}{C \mu}$.

- If $f_U \leq \frac{\Lambda}{C \mu}$, then we have $s > 0$, hence $\sigma = \sigma_U$ and $\alpha_s = 0$. Moreover, $\alpha_f = \sigma_U - \varphi > 0$, thus $f = f_U$ and $s = \frac{\Lambda}{C \mu} \left( 1 + \sqrt{\frac{m \Lambda^2}{(C \mu f - \Lambda)^2}} - f_U \right)$.

- If $f_U > \frac{\Lambda}{C \mu}$, we have to distinguish three cases:

- If $\sigma < \frac{\Lambda}{C \mu} \frac{m \Lambda^2}{(C \mu f - \Lambda)^2}$, then we can prove that $s > 0$. In fact, if $s = 0$ then:

\[
\sigma = \frac{-m \Lambda^2}{(C \mu f - \Lambda)^2} + \alpha_s \geq \frac{-m \Lambda^2}{(C \mu f - \Lambda)^2} \geq \frac{-m \Lambda^2}{(C \mu f - \Lambda)^2} \text{ which is impossible. Hence we have } \alpha_s = 0,
\]

\[
\sigma = \lambda_U, \quad \alpha_f = \alpha_f - \sigma = \sigma_U - \varphi > 0, \quad f = f_U, \quad s = \frac{\Lambda}{C \mu} \left( 1 + \sqrt{\frac{m \Lambda^2}{(C \mu f - \Lambda)^2}} - f_U \right).
\]
If \( \frac{-m \Lambda^2}{(C \mu f - \Lambda)^2} < \varphi \), then we can prove that 
\( \alpha_f = 0 \) and \( s = 0 \). In fact, if \( \alpha_f > 0 \) then \( f = f^U \) and

\[
\varphi = \frac{-m \Lambda^2}{(C \mu (f^U + s) - \Lambda)^2} - \alpha_f < \frac{-m \Lambda^2}{(C \mu (f^U + s) - \Lambda)^2} \leq \frac{-m \Lambda^2}{(C \mu f^U - \Lambda)^2},
\]

which is impossible. Moreover, if \( s > 0 \) then \( \sigma = \sigma^U \) and \( 0 = \alpha_s = \sigma^U - \varphi > 0 \) which is impossible. Hence we obtain \( f = \frac{\Lambda}{\nu^U} \left(1 + \sqrt{-\frac{m}{\varphi}}\right) \) and \( \sigma = \varphi + \alpha_s \geq \varphi \).

If \( \varphi \leq \frac{-m \Lambda^2}{(C \mu f^U - \Lambda)} \leq \sigma^U \), then we can prove that \( s = 0 \) and \( f = f^U \). In fact, if \( s > 0 \) then \( \alpha_s = 0 \), \( \sigma = \sigma^U \), and

\[
\alpha_f = \frac{-m \Lambda^2}{(C \mu (f + s) - \Lambda)^2} - \varphi = \sigma^U - \varphi > 0,
\]

hence \( f = f^U \) and

\[
\sigma^U \geq \frac{-m \Lambda^2}{(C \mu f^U - \Lambda)^2} \geq \frac{-m \Lambda^2}{(C \mu (f^U + s) - \Lambda)^2} = \sigma^U,
\]

which is impossible. Moreover, if \( f < f^U \) then \( \alpha_f > 0 \) and

\[
\varphi = \frac{-m \Lambda^2}{(C \mu f - \Lambda)^2} > \frac{-m \Lambda^2}{(C \mu f^U - \Lambda)^2} \geq \varphi,
\]

which is impossible. Finally, we have \( \sigma = \alpha_s - \frac{-m \Lambda^2}{(C \mu f^U - \Lambda)^2} \geq \frac{-m \Lambda^2}{(C \mu f^U - \Lambda)^2} \).