Generalized Nash Equilibria for SaaS/PaaS Clouds[*][†]

Jonatha Anselmi[‡] Danilo Ardagna[§] Mauro Passacantando[¶]

Abstract. Cloud computing is an emerging technology that allows to access computing resources on a pay-per-use basis. The main challenges in this area are the efficient performance management and the energy costs minimization. In this paper we model the service provisioning problem of Cloud Platform-as-a-Service systems as a Generalized Nash Equilibrium Problem and show that a potential function for the game exists. Moreover, we prove that the social optimum problem is convex and we derive some properties of social optima from the corresponding Karush-Kuhn-Tucker system. Next, we propose a distributed solution algorithm based on the best response dynamics and we prove its convergence to generalized Nash equilibria. Finally, we numerically evaluate equilibria in terms of their efficiency with respect to the social optimum of the Cloud by varying our algorithm initial solution. Numerical results show that our algorithm is scalable and very efficient and thus can be adopted for the run-time management of very large scale systems.

Keywords. Game Theory; Cloud Computing; Generalized Nash Equilibrium Problem.

2000 MSC: 91A10; 91A80.

1 Introduction

Cloud computing is an emerging paradigm that aims at streamlining the on-demand provisioning of flexible and scalable services accessible through the Internet [24]. The main idea is to supply users with on-demand access to computing or storage resources and charge fees for their usage. In these models, users pay only for the resources they use and they can access software

[*]Research partially supported by grant SA-2012/00331 of the Department of Industry, Innovation, Trade and Tourism (Basque Government) and grant MTM2010-17405 (Ministerio de Ciencia e Innovación, Spain) which sponsored a one-month visit of Danilo Ardagna to BCAM.

[†]This paper has been published in European Journal of Operational Research, vol. 236 (1), pp. 326–339.

[‡]Basque Center for Applied Mathematics (BCAM), 14 Mazarredo, 48009 Bilbao, Spain. E-mail: anselmi@bcamath.org

[§]Dipartimento di Elettronica e Informazione, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milan, Italy. E-mail: danilo.ardagna@polimi.it

[¶]Dipartimento di Informatica, Università di Pisa, Largo B. Pontecorvo 3, 56127 Pisa, Italy. E-mail: mpassacantando@di.unipi.it applications (Software as a Service – SaaS), tools for the development and deployment of Cloud-based applications (Platform as a Service – PaaS) or even low level hardware physical resources (Infrastructure as a Service – IaaS).

In the SaaS paradigm, applications are available over the Web. 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 also development tools. 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, Amazon, and Microsoft) are offering Cloud computing services such as Google's App Engine and Amazon's Elastic Compute Cloud (EC2) or Microsoft Windows Azure. Large data centers provide the infrastructure behind the Cloud, and virtualization technology (which allows the execution of multiple virtual machines on the same physical machine) 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. On the other hand, Cloud providers can 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 operate in a new and dynamic world, characterized by continuous changes in the environment and in the system and performance requirements that must be satisfied. Continuous changes occur without warning and in an unpredictable manner, and are outside the control of the Cloud provider. Therefore, advanced solutions need to be developed that manage the Cloud system in a dynamically adaptive fashion, while continuously providing service and performance guarantees. Recent studies [14, 17, 24] have shown that the main challenges for Cloud systems are the reduction of costs and the improvements of performance levels.

Information Technology (IT) analysts state that at the end of 2012, up to 40% of the budgets of Cloud service centers are devoted to energy costs [18, 28]. Service centers investment grew by 22.1% during 2012 and it is expected it will further grow by another 14.5% in 2013 [25]. Energy efficiency is therefore one of the main focal points on which resource management should be concerned. In addition, providers need to comply with Service Level Agreement (SLA) contracts that determine the revenues gained and penalties incurred on the basis of the level of performance achieved. Quality of Service (QoS) guarantees have to be satisfied despite workload fluctuations, which could span several orders of magnitude within the same business day [17, 18].

The recent development of Cloud systems and the rapid growth of the Internet have led to a remarkable usage of game-theoretic tools. Problems arising in the IT industry, such as quality of service or resource allocation, pricing, and load shedding, can not be handled with classical optimization approaches because each player can be affected by the actions of all players, not only by his own actions. In this setting, a natural modeling framework involves seeking an equilibrium or stable operating point for the system, provided that it exists. 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 improve his objective function by changing unilaterally his strategy. A survey of different modeling and solution concepts of networking games, as well as a number of different applications in telecommunications and wireless networks, based on game theory, can be found in [2].

In Cloud computing, game theory methods have been used to provide load balancing and resource allocation solutions. A number of papers consider centralized and decentralized load balancing strategies in a system with parallel servers (see [7, 29] and the references therein). The requests arrive as a Poisson process, and the service time of incoming jobs is assumed to be known. For such system, 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 in terms of Price of Anarchy (PoA) [30] of the decentralized scheme relative to the global optimal (centralized) one.

In [41], the authors propose a pricing mechanism for resource allocation 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 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 modeled as a game. Then, 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 [41] the problem of the resource allocation is considered for a single virtualized server among competing user requests, while in this paper we consider the Cloud data center at a higher granularity (i.e., VMs).

In this paper we take the perspective of SaaS providers which host their applications at a PaaS provider. Each SaaS provider wants to maximize its profit while complying with QoS requirements of their end-users, which determine the revenues and the 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 resources supplied by the PaaS. The profit maximization is challenging since on-line services see dynamic workloads that fluctuate over multiple time scales [17, 22]. Resources have to be allocated flexibly at run-time according to workload fluctuations. Furthermore, each SaaS behaves selfishly and competes with others SaaSs for the use of resources supplied by the PaaS. The PaaS, in its turn, wants to maximize the revenues obtained providing the resources. The profits are given by the revenues from the SLA with the SaaSs minus the energy costs sustained for running physical servers.

To capture the behavior of SaaSs and PaaS in this conflicting situation, we will model the run-time service provisioning problem as a *Generalized Nash Equilibrium Problem (GNEP)* (see e.g. [16, 19, 23, 26, 37]), which is an extension of the classical Nash equilibrium problem [35], in which both the objective function and the feasible region of each player depend on the strategies chosen by the other players. We then use results from game theory to develop efficient algorithms for the run-time management and allocation of PaaS resources to competing SaaSs.

In [10, 11], we have considered a problem similar to the one faced here, analysing the service provisioning problem with on spot resources. With respect to our previous work, in this paper we extend the game-theoretic model considering a new and more realistic pricing model regulating SaaS and PaaS contract. Furthermore, we explicitly model energy costs of the PaaS infrastructure which requires to include additional decision variables and a totally different solution approach.

The remainder of the paper is organized as follows. Section 2 describes the problem under study. In Section 3, we introduce our model based on the concept of Generalized Nash Equilibrium (GNE); moreover, we prove the existence of a potential function for the game and of social optimum solutions. In Section 4, we prove that the social optimum problem is convex and we derive some properties of social optima from the corresponding Karush-Kuhn-Tucker system. In Section 5, we provide a solution algorithm based on the best reply dynamics and we prove its convergence to GNE. The experimental results discussed in Section 6 demonstrate the efficiency of our algorithm by varing its initial solution. Conclusions are finally drawn in Section 7.

2 Problem Description

We consider SaaS providers using Cloud computing facilities according to the PaaS paradigm to offer multiple transactional Web-Services (WSs), each service representing 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 SaaS provider s are denoted by $\mathcal{A}_s$, while $\mathcal{S}$ indicates the set of SaaSs. In the following, we denote by $\mathcal{A}$ the set of all WS applications hosted at the PaaS location, i.e., $\mathcal{A} \overset{\text{def}}{=} \cup_{s\in\mathcal{S}}\mathcal{A}_s$, and we assume $\mathcal{A}_{s_1} \cap \mathcal{A}_{s_2} = \emptyset$, for all $s_1 \neq s_2$.

Each SaaS provider signs with its customers an SLA contract specifying: (i) the QoS levels that it must meet while responding to end-user requests for a given application, and (ii) the corresponding pricing scheme. In particular, we assume SaaSs have to guarantee to their end-users that the average response time when accessing application k is less than or equal to a given threshold R_k^S, while the per-request revenue for the SaaS is α_k (see Figure 1).

In turn, each SaaS provider signs an SLA contract with the PaaS: The PaaS must guarantee that the average response time for WS application k is less than or equal to $R_k^P(\leq R_k^S)$. Furthermore, for the execution of a single request $k \in \mathcal{A}_s$, SaaS s pays a fee β_k to the PaaS. The SaaS per-request revenue α_k is greater than or equal to $m_k\,\beta_k$, where $m_k > 1$ is the SaaS

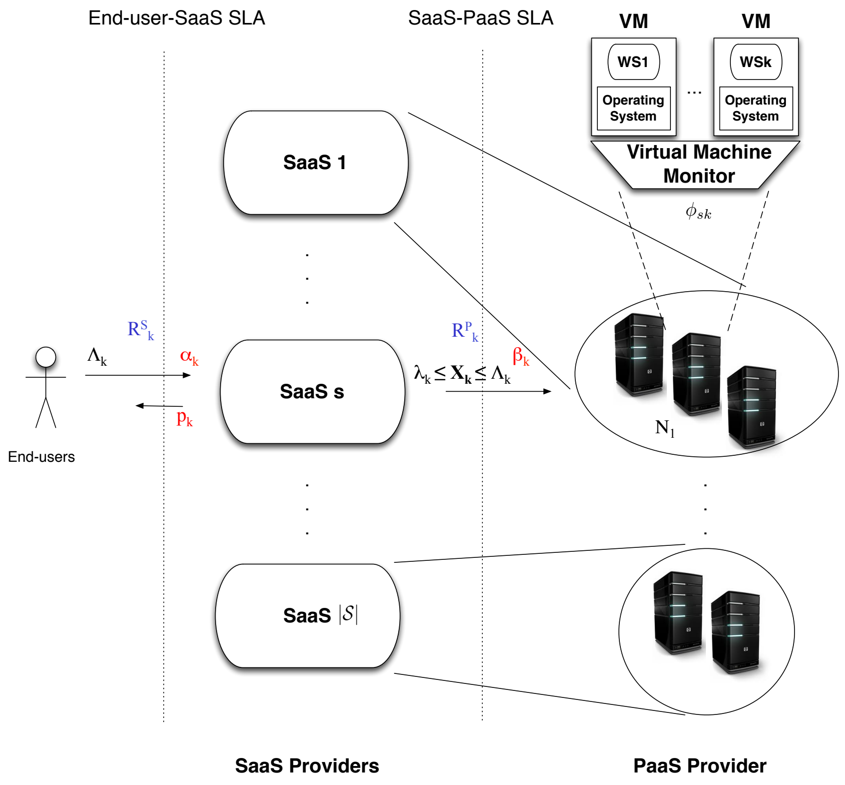

Figure 1: SLA contracts among the players.

margin for the execution of individual requests. Both R_k^P and β_k may vary with the time of the day. Therefore, the SaaS can ask the PaaS provider to reduce its application response time to improve the final end-user experience and thus to increase the loyalty of its customers. However, intuitively, the more stringent are the performance requirements imposed by the SaaS, the higher is the number of resources devoted by the PaaS to the SaaS and overall the higher is the per-request fee β_k payed by the SaaS. In particular, we assume that the average response times R_k^P enters the SaaS payoff function with a linear function, $U_k = \delta_s\,(R_k^S - R_k^P)$. Linear functions are a flexible mechanism to express the discontent of final end-users as a function of response times and have been widely used in the literature (see e.g. [12, 15, 38]).

Current Cloud platforms are based on virtualization technology, which allows a flexible management of the overall intrastructure. Therefore, we assume that applications are executed in virtual machines (VMs), which are dynamically instantiated by the PaaS 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. We also assume that physical servers are dedicated to SaaS providers, i.e. every physical server runs the VMs of the same SaaS. Cloud users usually prefer dedicated servers especially for business critical applications and/or when security requirements are of paramount importance. The use of dedicated servers is becoming widespread in the Cloud market [3]. For example, Amazon has devoted specific data centers to run applications for US government agencies and contractors [5].

For simplicity, we assume that physical servers are homogeneous, having the same processing capacity C. However, our framework can be extended relaxing this latter assumption. In the following, N^P denotes the total number of physical servers used for the dedicated execution of WS applications at the PaaS site.

On each physical servers, we assume that SaaS VMs are replicated with a fixed pattern for fault-tolerance reasons [8] (e.g., for every DBMS instance, two instances of application servers and three instances of web servers are allocated) and if any additional physical server is needed, VMs are replicated on the new server according to this fixed allocation pattern. In the following, we will denote by v_{sk} the proportion of VMs for WS application k hosted by one server dedicated to SaaS s.

The number of physical servers that the PaaS decides to supply to SaaS provider s is denoted by N_s. Since the workload of Internet applications can vary by orders of magnitude within the same business day [17], the physical servers are dynamically allocated by the PaaS provider periodically, e.g., every hour, according to a short-term workload prediction. One hour is also the time unit which is usually adopted by PaaS and IaaS to bill the use of resources to their customers [4]. We denote by Λ_k the prediction for the arrival rate of WS application k.

In this context, each SaaS provider can make the decision of accepting or rejecting a WS application execution request to maximize its own revenue. In other terms, SaaS providers can implement an admission control scheme trading off between the platform costs and the revenues they gain from their customers [1]. We assume that such decisions are taken according to some i.i.d. probabilistic law. The resulting application execution rate (or throughput, acceptance rate) is denoted by $X_k \leq \Lambda_k$. SaaS providers may possibly incur in penalties $p_k \geq 0$ upon rejection of request executions of k. In order to fix the rejection rate above a fixed threshold and guarantee a minimum availability, SaaS s may decide to guarantee a minimum throughput λ_k.

We assume that each physical server runs a Virtual Machine Monitor (VMM) configured in work-conserving mode. The resource allocation mechanism implemented by VMMs can be modeled as a first approximation by the Generalized Processor Sharing (GPS) [36] scheduling (see e.g. [20, 39]). Under GPS, the fraction of the available processing capacity of each server devoted to WS application k at time t is:

$$\frac{\phi_{sk}}{\sum\limits_{l\in\mathcal{K}(t)} \phi_{sl}}, \tag{1}$$

where ϕ_{sk} denotes the CPU *share*, or weight, of application k of the SaaS provider s, and $\mathcal{K}(t) \subseteq \mathcal{A}_s$ is the set of WS applications with waiting requests at time t.

To estimate the per-request mean response time achieved with the GPS mechanism, we adopt the approximation proposed in [8, Formula (18)]. Under the assumption that the requests of a given application are evenly distributed among the physical servers in a probabilistic manner, the mean response time for application k can be approximated by

$$R_k = \frac{N_s}{X_k} \cdot \frac{\rho_s \frac{\rho_{sk}}{\phi_{sk}}}{C \sum_{l\in\mathcal{A}_s} v_{sl} \frac{\rho_{sl}}{\phi_{sl}} - \rho_s \frac{\rho_{sk}}{\phi_{sk}}}, \tag{2}$$

where X_k/N_s is the arrival rate of application-k requests to one physical server of SaaS s (requests are evenly distributed among the N_s physical servers), μ_k denotes the maximum service rate of a capacity-one server for executing a class k request, $\rho_{sk} \overset{\text{def}}{=} X_k/(\mu_k\, N_s)$ is interpreted as the "utilization" of application k requests, and $\rho_s \overset{\text{def}}{=} \sum_{l\in\mathcal{A}_s} v_{sl}\, \rho_{sl}$. Formula 2 renders the mean response time of a virtual machine when the VMM uses the GPS mechanism described above; we point the interested reader to [8] for further details.

It is shown in [8, Theorem 5] that the mean response time of all SaaS-s applications, that is

$$\sum_{k\in\mathcal{A}_s} \frac{X_k}{\sum\limits_{k'\in\mathcal{A}_s} X_{k'}} R_k$$

where R_k is given by (2), is minimized when $\phi_{sk} = \rho_{sk}$, for all $k \in \mathcal{A}_s$. We assume that PaaSs make this choice for the CPU shares, which implies that the average WS application k response time becomes (upon substitution in (2) and noting that $\sum_{l\in\mathcal{A}_s} v_{sl} = 1$)

$$R_k = \frac{N_s}{X_k} \cdot \frac{\sum_{l\in\mathcal{A}_s} v_{sl} X_l/\mu_l}{C\, N_s - \sum_{l\in\mathcal{A}_s} v_{sl} X_l/\mu_l}. \tag{3}$$

Such choice of the weights induce a fair load-balancing among the number of on-going WS execution requests. In fact, (3) implies that the number on-going WS execution requests of application k, which is $R_k\, X_k/N_s$ by Little's law [32], is independent of k.

Remark 1. *In the remainder of the paper, we use* (3) *as model of the response time of application k requests on PaaS s. However, part of the results presented below immediately follow by only using the convexity of R_k. This lets us stipulate that they hold also for a wider class of response time functions.*

Finally, we denote by c the time unit cost for a physical server when it is turned on, including its power consumption and cooling overhead [28].

Table 1 summarizes the notation used in this paper for quick reference. Note that, the SaaS SLA contract parameters (e.g., R_k^S or p_k) are usually public available and stored in public registries [13]. Requests arrival rates and resource demands (i.e., Λ_k, μ_k) can be determined by SaaS and PaaS by prediction methods or monitoring [6, 9]. Vice versa, platform parameters (e.g., servers time unit costs c and the overall number of servers available N) are known only by the PaaS. Hence, the PaaS has or can determine easily the full knowledge of the system parameters.

System Parameters	
$\mathcal{S}$	Set of SaaS providers
$\mathcal{A}_s$	Set of applications offered by SaaS s
$\mathcal{A}$	$= \cup_{s \in \mathcal{S}} \mathcal{A}_s$
C	Processing capacity of physical servers
α_k	Revenue for the SaaS provider for single request execution of WS application k
m_k	SaaS margin for the execution of individual requests
p_k	SaaS penalty for application k requests rejection
δ_s	Utility function slope for SaaS provider s
R_k^S	Upper bound on the average response time of WS application k guaranteed to the SaaS customers
λ_k	Minimum arrival rate guaranteed for WS application k
Λ_k	Prediction (maximum) arrival rate for WS application k
v_{sk}	Proportion of VMs running WS application k hosted by a physical server dedicated to SaaS s
μ_k	Maximum service rate of a capacity 1 server for executing a class k request
c	Time unit cost for a physical server when it is turned on
N^P	Overall number of physical servers available to the PaaS
PaaS Decision Variables	
β_k	Revenue for the PaaS provider for single request execution of WS application k
N_s	Number of physical servers dedicated to SaaS s
SaaS Decision Variables	
X_k	Throughput for application k
R_k^P	Upper bound for the average response time of WS application k

Table 1: Parameters and decision variables

3 Game-theoretic Model

As discussed in Section 1, the goal of the PaaS is to maximize its revenues obtained from the execution of SaaS applications minus the cost incurred with the use of the physical servers.

On the other hand, the goal of each SaaS provider is to maximize its payoff which considers the profits obtained from the execution of incoming requests, the costs incurred for the use of the platform and the penalties associated with request rejections. Each SaaS payoff function finally includes also a linear term which takes into account, for each application-k request, the impact of the response time R_k^P on the end-user loyalties and expresses the discontent of final end-users experiencing large response times.

As it will be detailed in the following, the behavior of the PaaS and the SaaSs in this conflicting situation can be modeled as a GNEP. Section 3.1 formulates the PaaS resource allocation problem, while Section 3.2 formalizes SaaS providers optimization problems. The Generalized Nash equilibria of the game which is originated with this setting are defined in Section 3.3 and a potential function for the game is described in Section 3.4.

3.1 Game formulation from the PaaS side

The PaaS optimization problem is:

$$\max_{\beta_k, N_s} \sum_{k \in \mathcal{A}} \beta_k \, X_k - \sum_{s \in \mathcal{S}} c \, N_s \tag{4}$$

subject to:

$$\beta_k \leq \frac{\alpha_k}{m_k}, \qquad \forall\, k \in \mathcal{A}, \tag{5}$$

$$\sum_{s \in \mathcal{S}} N_s \leq N^P, \tag{6}$$

$$\frac{N_s \sum_{l \in \mathcal{A}_s} v_{sl} X_l / \mu_l}{C\, N_s - \sum_{l \in \mathcal{A}_s} v_{sl} X_l / \mu_l} \leq R_k^P \, X_k, \qquad \forall\, s \in \mathcal{S},\ \forall\, k \in \mathcal{A}_s, \tag{7}$$

$$\sum_{l \in \mathcal{A}_s} \frac{v_{sl} X_l}{\mu_l} < C\, N_s, \qquad \forall\, s \in \mathcal{S}. \tag{8}$$

The first and second terms of the payoff function are the revenues for the execution of end-user requests and the costs of using physical servers, respectively. Constraint family (5) guarantees each SaaS with a margin for the execution of end-user requests ($m_k > 1$). Constraint (6) entails that the total number of servers adopted is lower than the one available. Constraints (7) guarantee that the average response time for requests execution satisfies the SLA between PaaS and SaaSs. Finally, constraint family (8) guarantees that physical servers resources are not saturated.

In the formulation of the PaaS problem, we have not imposed variables N_s to be integer, as in reality they are. In fact, requiring variables to be integer makes the solution much more difficult. Therefore, we consider the continuous relaxation of the problem. However, 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. This is a common assumption adopted in the literature [43] and is also intuitive for large scale data centers including thousands of servers.

We note that the objective function of the PaaS provider is linear and increasing with respect to variables β_k which are bounded above by the constants α_k/m_k. Hence $\beta_k = \alpha_k/m_k$ in every optimal solution of the PaaS provider for any strategy chosen by the SaaS providers. Thus the PaaS revenue for the execution of end-user requests is equal to $\sum_{k\in\mathcal{A}} \alpha_k\, X_k/m_k$, which depends only on variables X_k chosen by SaaS providers and is independent from PaaS variables N_s, hence this term can be deleted from the PaaS objective function. In other words, the PaaS optimization problem consists in minimizing costs of using physical servers. From now on, we assume that the PaaS optimization problem has the following form:

$$\max_{N_s} \; \Theta_p \overset{\text{def}}{=} -\sum_{s\in\mathcal{S}} c\, N_s \tag{9}$$

subject to:

$$\sum_{s\in\mathcal{S}} N_s \le N^P, \tag{10}$$

$$\frac{N_s \sum_{l\in\mathcal{A}_s} v_{sl} X_l/\mu_l}{C N_s - \sum_{l\in\mathcal{A}_s} v_{sl} X_l/\mu_l} \le R_k^P X_k, \qquad \forall\; s\in\mathcal{S},\ \forall\; k\in\mathcal{A}_s, \tag{11}$$

$$\sum_{l\in\mathcal{A}_s} v_{sl}\, X_l/\mu_l < C\, N_s, \qquad \forall\; s\in\mathcal{S}. \tag{12}$$

We note that the problem (9)–(12) is easy to solve because constraints (11)-(12) can be rearranged to obtain linear constraints with respect to N_s so that each variable N_s can be optimized separately. Hence the PaaS optimal solution is

$$N_s = \max_{k\in\mathcal{A}_s} \frac{R_k^P\, X_k \sum_{l\in\mathcal{A}_s} v_{sl}\, X_l/\mu_l}{C\, R_k^P\, X_k - \sum_{l\in\mathcal{A}_s} v_{sl}\, X_l/\mu_l}, \qquad \forall\; s\in\mathcal{S}, \tag{13}$$

provided that the sum of the right-hand sides does not exceed the upper bound N^P; otherwise there is no feasible solution.

3.2 Game Formulation from the SaaS Side

The SaaS s optimization problem is:

$$\max_{X_k, R_k^P} \; \Theta_s \overset{\text{def}}{=} \sum_{k\in\mathcal{A}_s} \left[\alpha_k \left(1-\frac{1}{m_k}\right) X_k - p_k\,(\Lambda_k - X_k) + \delta_s\,(R_k^S - R_k^P)\, X_k\right] \tag{14}$$

subject to:

$$\lambda_k \leq X_k \leq \Lambda_k, \qquad \forall\, k \in \mathcal{A}_s, \tag{15}$$

$$R_k^P \leq R_k^S, \qquad \forall\, k \in \mathcal{A}_s, \tag{16}$$

$$\frac{N_s \sum_{l \in \mathcal{A}_s} v_{sl} X_l/\mu_l}{C N_s - \sum_{l \in \mathcal{A}_s} v_{sl} X_l/\mu_l} \leq R_k^P\, X_k, \qquad \forall\, k \in \mathcal{A}_s, \tag{17}$$

$$\sum_{l \in \mathcal{A}_s} v_{sl} X_l/\mu_l < C N_s. \tag{18}$$

The first terms of the payoff function compute the net revenues obtained from the execution of end-users' request. The second terms determine the penalties incurred with request rejections. Finally, the last term evaluates the response time R_k^P for WS application k requests according to the utility function established in the SLA between the SaaS and each end-user. Constraint family (15) states that X_k cannot be larger than the actual arrival rate Λ_k and smaller than the minimum admission rate λ_k. Constraint family (16) entails that the response time for application k negotiated with the PaaS is smaller than R_k^S. As in the previous section, constraints (17) and (18) guarantee that the application response time satisfies the SLA and that physical servers are not saturated, respectively.

3.3 Generalized Nash Equilibria

The model resulting from the optimization problems described in the previous sections is a Generalized Nash Equilibrium Problem with joint constraints, where the players are the PaaS and SaaS providers: The strategies of the PaaS provider are $N = (N_s)_{s \in \mathcal{S}}$, the strategies of each SaaS provider s are $X_s = (X_k)_{k \in \mathcal{A}_s}$ and $R_s^P = (R_k^P)_{k \in \mathcal{A}_s}$. Each SaaS provider shares constraints (17) and (18) with the PaaS provider.

In this setting, a Generalized Nash Equilibrium (GNE) is a set of strategies such that no player can improve its payoff function by changing its strategy unilaterally [26], i.e. a GNE is a vector $(\overline{X}, \overline{R}^P, \overline{N})$ such that the following relations hold:

$$\Theta_p(\overline{N}) \geq \Theta_p(N), \qquad \forall\, N \text{ s.t. } (\overline{X}, \overline{R}^P, N) \text{ satisfies constraints (10)–(12)}, \tag{19}$$

and for all $s \in \mathcal{S}$

$$\Theta_s(\overline{X}_s, \overline{R}_s^P) \geq \Theta_s(X_s, R_s^P), \qquad \forall\, (X_s, R_s^P) \text{s.t. } (X_s, R_s^P, \overline{N}) \text{ satisfies constraints (15)–(18).} \tag{20}$$

3.4 Potential function

Since (i) the payoff function Θ_p of PaaS provider depends only on his own strategies N_s, (ii) the payoff function Θ_s of each SaaS provider s only depends on his own strategies X_s and R_s^P, and (iii) constraints (17) and (18) are shared among the players, we can conclude that this game is a generalized potential game [27, 34] where the potential function is simply the sum of the players payoff functions. This potential function represents a social welfare for the game.

Proposition 1. *The function*

$$\Pi(X, R^P, N) \overset{\text{def}}{=} \sum_{s\in\mathcal{S}} \sum_{k\in\mathcal{A}_s} \left[\alpha_k \left(1 - \frac{1}{m_k}\right) X_k - p_k\,(\Lambda_k - X_k) + \delta_s\,(R_k^S - R_k^P)\,X_k\right] - \sum_{s\in\mathcal{S}} c\,N_s \tag{21}$$

is a potential for the game.

We denote by Ω the feasible region containing the variables of all the players, i.e.

$$\Omega = \{(X, R^P, N) : \text{ (10), (11), (12) hold and (15)-(16) hold for all } s \in \mathcal{S}\}.$$

Since this is a generalized potential game, we can assume that the potential Π is the payoff function of each player; therefore each global maximizer of Π on the set Ω, called social optimum, is a special Generalized Nash Equilibrium (GNE). In other words, social optima represent the GNE which are optimal from a social point of view.

Proposition 2. *There exists at least one social optimum.*

Proof. First, we prove that Ω is a closed set. For any $(X, R^P, N) \in \Omega$ we have:

$$\begin{aligned}
C\,N_s - \sum_{l\in\mathcal{A}_s} \frac{v_{sl} X_l}{\mu_l} &\geq \frac{N_s \sum_{l\in\mathcal{A}_s} \frac{v_{sl} X_l}{\mu_l}}{R_k^P\,X_k} && \text{[from (11)]}\\
&\geq \frac{N_s \sum_{l\in\mathcal{A}_s} \frac{v_{sl} X_l}{\mu_l}}{R_k^S\,\Lambda_k} && \text{[from (15)-(16)]}\\
&\geq \frac{\left(\sum_{l\in\mathcal{A}_s} \frac{v_{sl} X_l}{\mu_l}\right)^2}{C\,R_k^S\,\Lambda_k} && \text{[from (12)]}\\
&\geq \frac{\left(\sum_{l\in\mathcal{A}_s} \frac{v_{sl} \lambda_l}{\mu_l}\right)^2}{C\,R_k^S\,\Lambda_k} && \text{[from (15)]}
\end{aligned}$$

for all $s \in \mathcal{S}$ and $k \in \mathcal{A}_s$. Therefore we obtain that

$$C\,N_s - \sum_{l\in\mathcal{A}_s} \frac{v_{sl} X_l}{\mu_l} \geq \max_{k\in\mathcal{A}_s} \left\{ \frac{\left(\sum_{l\in\mathcal{A}_s} \frac{v_{sl} \lambda_l}{\mu_l}\right)^2}{C\,R_k^S\,\Lambda_k} \right\} \qquad \forall\ s \in \mathcal{S}, \tag{22}$$

hence in the definition of Ω we can replace constraints (12) with constraints (22), thus Ω is a closed set. Since Ω is also bounded and Π is continuous, the existence of social optima follows from the well-known Weierstrass theorem. $\square$

4 The Social Optimum Problem

The Cloud provisioning game is extremely challenging since the number of SaaS providers $|\mathcal{S}|$ and WS applications $|\mathcal{A}|$ characterizing problem instances of interest in practice are extremely large (i.e., hundreds of SaaS and thousands of applications [17]). In order to identify efficient solution methods, in this section we analyze the social optimum problem determining its main properties.

In particular, we will show that constraints (11) hold as equality in any social optimum and that the social optimum problem is convex. Furthermore, bounds relating the platform capacity, the energy costs and the penalties incurred in case of requests rejection will be identified for interesting system regimes.

Proposition 3. *If* $(\overline{X}, \overline{R}^P, \overline{N})$ *is a social optimum, then the response time constraints* (11) *are all active, i.e.*

$$\frac{\overline{N}_s \sum_{l\in\mathcal{A}_s} v_{sl}\overline{X}_l/\mu_l}{C\,\overline{N}_s - \sum_{l\in\mathcal{A}_s} v_{sl}\overline{X}_l/\mu_l} = \overline{R}_k^P\,\overline{X}_k, \qquad \forall\ s\in\mathcal{S},\ \forall\ k\in\mathcal{A}_s. \tag{23}$$

Proof. Suppose, by contradiction, that there exists an index $k \in \mathcal{A}_s$, for some s, such that

$$\frac{\overline{N}_s \sum_{l\in\mathcal{A}_s} v_{sl}\overline{X}_l/\mu_l}{C\,\overline{N}_s - \sum_{l\in\mathcal{A}_s} v_{sl}\overline{X}_l/\mu_l} < \overline{R}_k^P\,\overline{X}_k.$$

If we slightly decrease the value of $\overline{R}_k^P$, keeping fixed the values of the other variables, we obtain a new feasible vector (X, R^P, N) such that $\Pi(X, R^P, N) > \Pi(\overline{X}, \overline{R}^P, \overline{N})$, which is impossible. $\square$

It follows from Proposition 3 that the variables R_k^P can be expressed as function of X_k and N_s, hence the social optimum problem can be rewritten as follows:

$$\begin{aligned}\max_{X,N}\ &\sum_{s\in\mathcal{S}}\sum_{k\in\mathcal{A}_s}\left[\alpha_k\left(1-\frac{1}{m_k}\right)X_k - p_k(\Lambda_k - X_k) + \delta_s R_k^S X_k\right]\\ &-\sum_{s\in\mathcal{S}}\left[cN_s + \delta_s|\mathcal{A}_s|\frac{N_s\sum_{l\in\mathcal{A}_s} v_{sl}X_l/\mu_l}{CN_s - \sum_{l\in\mathcal{A}_s} v_{sl}X_l/\mu_l}\right]\end{aligned} \tag{24}$$

subject to:

$$\lambda_k \le X_k \le \Lambda_k, \qquad \forall\ k\in\mathcal{A}, \tag{25}$$

$$\frac{N_s\sum_{l\in\mathcal{A}_s} v_{sl}X_l/\mu_l}{C\,N_s - \sum_{l\in\mathcal{A}_s} v_{sl}X_l/\mu_l} \le R_k^S\,X_k, \qquad \forall\ s\in\mathcal{S},\ \forall\ k\in\mathcal{A}_s, \tag{26}$$

$$\sum_{s\in\mathcal{S}} N_s \le N^P, \tag{27}$$

$$\sum_{l\in\mathcal{A}_s} v_{sl}X_l/\mu_l < C\,N_s, \qquad \forall\ s\in\mathcal{S}. \tag{28}$$

Proposition 4. *The social optimum problem* (24)–(28) *is convex.*

Proof. We have to prove that the objective function is concave and all the constraints are convex. To this end, it is sufficient to show that the functions

$$(X_s, N_s) \mapsto \frac{N_s\sum_{l\in\mathcal{A}_s} v_{sl}X_l/\mu_l}{CN_s - \sum_{l\in\mathcal{A}_s} v_{sl}X_l/\mu_l}$$

are convex when constraints (28) are satisfied. Since all these functions have the same structure, we can consider only the function

$$h(x_1,\dots,x_n,y) \overset{\text{def}}{=} \frac{y\sum_{i=1}^n a_i x_i}{b\,y-\sum_{i=1}^n a_i x_i},$$

where variables $x=(x_1,\dots,x_n)$ and y correspond to X_s and N_s respectively, and the constants $a_i>0$ and $b>0$ to v_{sl}/μ_l and C respectively. Therefore, it is sufficient to prove that the function h is convex when $b\,y-\sum_{i=1}^n a_i x_i > 0$ (which corresponds to constraint (28)).

The first derivatives of h are

$$\frac{\partial h}{\partial x_j} = \frac{b a_j y^2}{(b\,y-\sum_{i=1}^n a_i x_i)^2}, \quad j=1,\dots,n, \qquad \frac{\partial h}{\partial y} = \frac{-(\sum_{i=1}^n a_i x_i)^2}{(b\,y-\sum_{i=1}^n a_i x_i)^2};$$

the second derivatives of h are

$$\frac{\partial^2 h}{\partial x_j\,\partial x_\ell} = \frac{2\,b\,a_j\,a_\ell\,y^2}{(b\,y-\sum_{i=1}^n a_i x_i)^3}, \qquad j,\ell=1,\dots,n,$$

$$\frac{\partial^2 h}{\partial x_j\,\partial y} = \frac{-2\,b\,a_j\,y\,\sum_{i=1}^n a_i x_i}{(b\,y-\sum_{i=1}^n a_i x_i)^3}, \qquad j=1,\dots,n,$$

$$\frac{\partial^2 h}{\partial y^2} = \frac{2\,b\,(\sum_{i=1}^n a_i x_i)^2}{(b\,y-\sum_{i=1}^n a_i x_i)^3}.$$

Thus the Hessian matrix of h is

$$\nabla^2 h(x_1,\dots,x_n,y) = \frac{2\,b}{(b\,y-\sum_{i=1}^n a_i x_i)^3}\begin{bmatrix} a_1^2 y^2 & \cdots & a_1 a_n y^2 & -a_1 y\sum_{i=1}^n a_i x_i \\ \vdots & \ddots & \vdots & \vdots \\ a_1 a_n y^2 & \cdots & a_n^2 y^2 & -a_n y\sum_{i=1}^n a_i x_i \\ -a_1 y\sum_{i=1}^n a_i x_i & \dots & -a_n y\sum_{i=1}^n a_i x_i & (\sum_{i=1}^n a_i x_i)^2 \end{bmatrix}.$$

For any vector $u=(u_1,\dots,u_n,u_{n+1})\in\mathbb{R}^{n+1}$ we have

$$\begin{aligned} u^T\,\nabla^2 h(x_1,\dots,x_n,y)\,u &= \frac{2\,b}{(b\,y-\sum_{i=1}^n a_i x_i)^3}\,u^T\begin{bmatrix} a_1\,y^2\,\sum_{i=1}^n a_i u_i - a_1\,y\,u_{n+1}\,\sum_{i=1}^n a_i x_i \\ \vdots \\ a_n\,y^2\,\sum_{i=1}^n a_i u_i - a_n\,y\,u_{n+1}\,\sum_{i=1}^n a_i x_i \\ -y\,(\sum_{i=1}^n a_i x_i)\,(\sum_{i=1}^n a_i u_i) + u_{n+1}\,(\sum_{i=1}^n a_i x_i)^2 \end{bmatrix} \\ &= \frac{2\,b}{(b\,y-\sum_{i=1}^n a_i x_i)^3}\left[y^2\left(\sum_{i=1}^n a_i u_i\right)^2 - 2\,y\,u_{n+1}\left(\sum_{i=1}^n a_i u_i\right)\left(\sum_{i=1}^n a_i x_i\right)\right. \\ &\qquad\left. + u_{n+1}^2\left(\sum_{i=1}^n a_i x_i\right)^2\right] \\ &= \frac{2\,b}{(b\,y-\sum_{i=1}^n a_i x_i)^3}\left[y\sum_{i=1}^n a_i u_i - u_{n+1}\sum_{i=1}^n a_i x_i\right]^2. \end{aligned}$$

Therefore, if $b\,y-\sum_{i=1}^n a_i x_i>0$, then the Hessian matrix of h is positive semidefinite and hence h is convex. □

In the rest of this section we analyze the Karush-Kuhn-Tucker (KKT) system of the social optimum problem and derive some bounds for the social optima relating the platform capacity, the energy costs and the penalties incurred in case of requests rejection.

Since the social optimum problem is convex and the Slater constraint qualification holds, the KKT conditions are necessary and sufficient for the optimality. If we denote $L_k^i \geq 0$, for $k \in \mathcal{A}$ and $i = 1, 2, 3$, the KKT multipliers associated to constraints (25) and (26), and $L^4 \geq 0$, the multiplier associated to constraint (27), then the KKT system is the following:

$$\begin{aligned}
&\alpha_k \left(1 - \frac{1}{m_k}\right) + p_k + \delta_s\, R_k^S \\
&- \frac{C\, v_{sk}\, N_s^2}{\mu_k \left(C\, N_s - \sum_{l \in \mathcal{A}_s} v_{sl} X_l/\mu_l\right)^2} \left(\delta_s\, |\mathcal{A}_s| + \sum_{l \in \mathcal{A}_s} L_l^3\right) \\
&+ L_k^1 - L_k^2 + L_k^3\, R_k^S = 0, \qquad \forall\, s \in \mathcal{S},\ \forall\, k \in \mathcal{A}_s, && (29)\\
&-c + \left(\frac{\sum_{l \in \mathcal{A}_s} v_{sl}\, X_l/\mu_l}{C\, N_s - \sum_{l \in \mathcal{A}_s} v_{sl}\, X_l/\mu_l}\right)^2 \left(\delta_s\, |\mathcal{A}_s| + \sum_{l \in \mathcal{A}_s} L_l^3\right) - L^4 = 0, \qquad \forall\, s \in \mathcal{S}, && (30)\\
&L_k^1\, (X_k - \lambda_k) = 0, \qquad \forall\, k \in \mathcal{A}, && (31)\\
&L_k^2\, (X_k - \Lambda_k) = 0, \qquad \forall\, k \in \mathcal{A}, && (32)\\
&L_k^3 \left(\frac{N_s \sum_{l \in \mathcal{A}_s} v_{sl} X_l/\mu_l}{C\, N_s - \sum_{l \in \mathcal{A}_s} v_{sl} X_l/\mu_l} - R_k^S\, X_k\right) = 0, \qquad \forall\, s \in \mathcal{S},\ \forall\, k \in \mathcal{A}_s, && (33)\\
&L^4 \left(\sum_{s \in \mathcal{S}} N_s - N^P\right) = 0, && (34)\\
&\text{constraints (25)–(28).}
\end{aligned}$$

In the following we will analyze two limiting regimes for the Cloud system. In particular Proposition 5 considers that the system provides very good performance for a provider s (i.e., the constraints (26) are not active for the whole set of WS applications $\mathcal{A}_s$ hosted at the PaaS site), which corresponds to light load conditions for the SaaS s. Vice versa, Proposition 6 considers the case a provider s is under heavy load and the minumum workload is served for every WS applications in $\mathcal{A}_s$. These results are used in Section 5 to identify the initial solution for our resource allocation algorithm, and, as it will be further discussed in Section 6, they allow to achieve the best efficiency in terms of PoA.

In the remainder of the paper, we will denote with $\Psi_{sk} = \alpha_k \left(1 - \frac{1}{m_k}\right) + p_k + \delta_s\, R_k^S$ and we set $\Omega_{sk} = v_{sk}\, C/\mu_k$.

Proposition 5. *If (X, N) is a social optimum solution and there exists a SaaS provider s such that $X_k = \Lambda_k$ and the response time is strictly lower than R_k^S for all WS applications $k \in \mathcal{A}_s$,* *then:*

$$\frac{\sum\limits_{k\in\mathcal{A}_s} v_{sk}\,\Lambda_k/\mu_k}{C-\sqrt{\frac{\delta_s|\mathcal{A}_s|}{\min\limits_{k\in\mathcal{A}_s}\Psi_{sk}/\Omega_{sk}}}} \le N_s \le \frac{\sqrt{c}+\sqrt{\delta_s|\mathcal{A}_s|}}{C\,\sqrt{c}} \sum_{k\in\mathcal{A}_s} v_{sk}\,\Lambda_k/\mu_k. \tag{35}$$

Proof. Since for all $k \in \mathcal{A}_s$ the constraint (26) is not active and $X_k = \Lambda_k$, we obtain $L_k^1 = L_k^3 = 0$. It follows from (29) that

$$\Psi_{sk} - \frac{\Omega_{sk}\,\delta_s\,|\mathcal{A}_s|\,N_s^2}{(C\,N_s - \sum_{l\in\mathcal{A}_s} v_{sl}X_l/\mu_l)^2} = L_k^2 \ge 0 \qquad \forall\; k \in \mathcal{A}_s,$$

hence we have

$$\min_{k\in\mathcal{A}_s}\frac{\Psi_{sk}}{\Omega_{sk}} \ge \frac{\delta_s\,|\mathcal{A}_s|\,N_s^2}{(C\,N_s - \sum_{l\in\mathcal{A}_s} v_{sl}X_l/\mu_l)^2}.$$

Setting $X_l = \Lambda_l$ and taking into account the equilibrium condition $C\,N_s - \sum_{l\in\mathcal{A}_s} v_{sl}\,\Lambda_l/\mu_l > 0$, we obtain the first inequality of (35).

On the other hand, it follows from (30) that

$$\delta_s|\mathcal{A}_s|\left(\frac{\sum_{l\in\mathcal{A}_s} v_{sl}\,X_l/\mu_l}{C\,N_s - \sum_{l\in\mathcal{A}_s} v_{sl}\,X_l/\mu_l}\right)^2 - c = L^4 \ge 0.$$

Setting $X_l = \Lambda_l$ for all $l \in \mathcal{A}_s$, we get

$$\frac{\delta_s|\mathcal{A}_s|}{c}\left(\sum_{k\in\mathcal{A}_s} v_{sl}\,\Lambda_l/\mu_l\right)^2 \ge \left(C\,N_s - \sum_{l\in\mathcal{A}_s} v_{sl}\,\Lambda_l/\mu_l\right)^2,$$

which, together with the equilibrium condition $C\,N_s - \sum_{l\in\mathcal{A}_s} v_{sl}\,\Lambda_l/\mu_l > 0$, implies the second inequality of (35). □

Proposition 6. *If (X, N) is a social optimum solution and there exists a SaaS provider s such that $X_k = \lambda_k$ and the response time is strictly lower than R_k^S for all WS applications $k \in \mathcal{A}_s$, then*

$$N_s \le \min\left\{\frac{1}{C-\sqrt{\frac{\delta_s|\mathcal{A}_s|}{\max\limits_{k\in\mathcal{A}_s}\Psi_{sk}/\Omega_{sk}}}}\;,\;\frac{\sqrt{c}+\sqrt{\delta_s|\mathcal{A}_s|}}{C\,\sqrt{c}}\right\}\sum_{k\in\mathcal{A}_s} v_{sk}\,\lambda_k/\mu_k. \tag{36}$$

Proof. Since for all $k \in \mathcal{A}_s$ the constraint (26) is not active and $X_k = \lambda_k$, we obtain $L_k^2 = L_k^3 = 0$. From (29) we obtain

$$\Psi_{sk} - \frac{\Omega_{sk}\,\delta_s\,|\mathcal{A}_s|\,N_s^2}{(C\,N_s - \sum_{l\in\mathcal{A}_s} v_{sl}X_l/\mu_l)^2} = -L_k^1 \le 0, \qquad \forall\; k \in \mathcal{A}_s,$$

hence we have

$$\max_{k\in\mathcal{A}_s}\frac{\Psi_{sk}}{\Omega_{sk}} \le \frac{\delta_s\,|\mathcal{A}_s|\,N_s^2}{(C\,N_s - \sum_{l\in\mathcal{A}_s} v_{sl}X_l/\mu_l)^2}.$$

Setting $X_l = \lambda_l$ and taking into account the equilibrium condition $C\, N_s - \sum_{l \in \mathcal{A}_s} v_{sl}\, \lambda_l / \mu_l > 0$, we obtain

$$N_s \leq \frac{\sum\limits_{l \in \mathcal{A}_s} v_{sl}\, \lambda_l / \mu_l}{C - \sqrt{\frac{\delta_s |\mathcal{A}_s|}{\max\limits_{k \in \mathcal{A}_s} \Psi_{sk} / \Omega_{sk}}}}. \tag{37}$$

On the other hand, from (30) we get

$$\delta_s |\mathcal{A}_s| \left(\frac{\sum_{l \in \mathcal{A}_s} v_{sl}\, X_l / \mu_l}{C\, N_s - \sum_{l \in \mathcal{A}_s} v_{sl}\, X_l / \mu_l} \right)^2 - c = L^4 \geq 0.$$

Setting $X_l = \lambda_l$ for all $l \in \mathcal{A}_s$, we get

$$\frac{\delta_s |\mathcal{A}_s|}{c} \left(\sum_{k \in \mathcal{A}_s} v_{sl}\, \lambda_l / \mu_l \right)^2 \geq \left(C\, N_s - \sum_{l \in \mathcal{A}_s} v_{sl}\, \lambda_l / \mu_l \right)^2,$$

which, together with the equilibrium condition $C\, N_s - \sum_{l \in \mathcal{A}_s} v_{sl}\, \lambda_l / \mu_l > 0$, implies

$$N_s \leq \frac{\sqrt{c} + \sqrt{\delta_s |\mathcal{A}_s|}}{C\, \sqrt{c}} \sum_{k \in \mathcal{A}_s} v_{sk}\, \lambda_k / \mu_k. \tag{38}$$

Finally, the thesis follows from (37) and (38). □

5 Distributed Solution Method

In the previous Section it has been proved that the social optimum problem is convex and, from a theoretical point of view, it could be solved by the PaaS which has the full knowledge of system parameters (see Section 2). However, computational results demonstrate that only small instances can be solved to optimality with commercial nonlinear optimization packages (see Section 6). To handle representative problem sizes, in this Section we provide a solution algorithm which converges to a GNE. Several different versions of the algorithm we implemented are characterized by a different choice of the initial solution.

Before describing the solution algorithm, we analyze in more details the PaaS and SaaSs problems. In Section 3.1 we have shown that PaaS problem is easy to solve. On the other hand, the SaaS problem (14)–(18) can not be solved analytically, but it can be reformulated as a convex problem. In fact, we can prove similarly to Proposition 3 that constraints (17) are active in any SaaS optimal solution. Hence, the variables R_k^P can be expressed as function of X_k by formula (23) and we can rewrite the SaaS problem as follows:

$$\max_{X_s} \sum_{k \in \mathcal{A}_s} \left[\alpha_k \left(1 - \frac{1}{m_k} \right) X_k - p_k \left(\Lambda_k - X_k \right) + \delta_s\, R_k^S\, X_k \right] - \delta_s |\mathcal{A}_s| \frac{N_s \sum_{l \in \mathcal{A}_s} v_{sl}\, X_l / \mu_l}{C\, N_s - \sum_{l \in \mathcal{A}_s} v_{sl}\, X_l / \mu_l} \tag{39}$$

subject to:

$$\lambda_k \leq X_k \leq \Lambda_k, \qquad \forall\, k \in \mathcal{A}_s, \tag{40}$$

$$\frac{N_s \sum_{l\in\mathcal{A}_s} v_{sl} X_l/\mu_l}{C\,N_s - \sum_{l\in\mathcal{A}_s} v_{sl} X_l/\mu_l} \leq R_k^S\, X_k, \qquad \forall\, k \in \mathcal{A}_s, \tag{41}$$

$$\sum_{l\in\mathcal{A}_s} v_{sl} X_l/\mu_l < C N_s. \tag{42}$$

Finally, it can be proved that this problem is convex following the proof of Proposition 4. Now, we describe the solution algorithm and prove that it converges to a GNE.

Solution Algorithm

1. An initial value for the number of physical servers of each SaaS N_s is identified.
2. If $\sum_{s\in\mathcal{S}} N_s > N^P$, the PaaS provider reduces N_s proportionally to constraint (10) violation, i.e. the PaaS sets
$$\overline{N}_s = \frac{N_s\, N^P}{\sum\limits_{s'\in\mathcal{S}} N_{s'}}, \qquad \forall\, s \in \mathcal{S}; \tag{43}$$
otherwise PaaS provider set $\overline{N}_s = N_s$ for all $s \in \mathcal{S}$.
3. Given $\overline{N}$, each SaaS provider s finds a optimal solution $(\overline{X}_k)_{k\in\mathcal{A}_s}$, of the convex optimization problem (39)–(42) and set
$$\overline{R}_k^P = \frac{1}{\overline{X}_k}\, \frac{\overline{N}_s \sum_{l\in\mathcal{A}_s} v_{sl} \overline{X}_l/\mu_l}{C\,\overline{N}_s - \sum_{l\in\mathcal{A}_s} v_{sl} \overline{X}_l/\mu_l}, \qquad \forall\, k \in \mathcal{A}_s.$$

The algorithm performs three simple steps. An initial estimate for the number of physical servers is identified. If the initial server assignment is unfeasible, then the PaaS reallocates servers among SaaSs. Finally, each SaaS computes the optimal values for $(\overline{X}_s, \overline{R}_s^P)$ accordingly to its objective. We can obtain many different versions of the algorithm according to the way and the players (SaaS or PaaS) that select the initial number of servers. We first prove that the solution algorithm converges to a GNE, then we formulate five alternative methods that will be evaluated in the next Section.

Proposition 7. *If $\overline{N}$ is such that the feasible region* (40)–(42) *of each SaaS provider is non-empty, then the vector $(\overline{X}, \overline{R}^P, \overline{N})$ found by the solution algorithm is a GNE.*

Proof. Since each SaaS determines the best reply $(\overline{X}_s, \overline{R}_s^P)$ to the PaaS strategy $\overline{N}$, we obtain that all the constraints (17) are active. Thus, by formula (13) also the strategy $\overline{N}$ of the PaaS is the best reply to the strategies $(\overline{X}, \overline{R}^P)$ of the SaaS players. Hence $(\overline{X}, \overline{R}^P, \overline{N})$ is a GNE because of relations (19)–(20). □

According to the previous result, the solution algorithm finds a GNE. However, the equilibrium is found in a single best reply dynamic iteration and depends on the arbitrary choice N at step 1. Hence, finding good equilibria for the problem under study could be hard. We have implemented the following heuristic methods for defining a vector N which provides a good equilibrium.

- **Method 1**: Each SaaS randomly selects N_s at step 1. Random initialization at step 1 is considered as a benchmark for the comparison of alternative methods.
- **Method 2**: At step 1, each SaaS sets X_k randomly in the interval $[\lambda_k, \Lambda_k]$ and determines N_s such that physical servers average utilization is $\sum\limits_{l \in \mathcal{A}_s} \frac{v_{sl} X_l}{C\, N_s\, \mu_l} = 0.6$. In other words, the initial number of physical servers is determined according to the utilization thresholds principle which is an heuristic for Cloud systems resource allocation widely used in the literature [21, 40, 44] and adopted in practice by IaaS/PaaS providers [4].
- **Method 3**: At step 1, each SaaS s maximizes the function

$$\sum_{k \in \mathcal{A}_s} \left[\alpha_k \left(1 - \frac{1}{m_k} \right) X_k - p_k(\Lambda_k - X_k) + \delta_s R_k^S X_k \right] - cN_s - \delta_s |\mathcal{A}_s| \frac{N_s \sum_{l \in \mathcal{A}_s} v_{sl} X_l / \mu_l}{CN_s - \sum_{l \in \mathcal{A}_s} v_{sl} X_l / \mu_l}$$

  subject to constraints (40)–(42), considering both X_s and N_s as decision variables.
- **Method 4**: At step 1, the PaaS sets

$$N_s = \frac{\sqrt{c} + \sqrt{\delta_s |\mathcal{A}_s|}}{C\, \sqrt{c}} \sum_{k \in \mathcal{A}_s} v_{sk}\, \Lambda_k / \mu_k,$$

  according to Proposition 5.
- **Method 5**: At step 1, the PaaS sets

$$N_s = \min \left\{ \frac{1}{C - \sqrt{\frac{\delta_s |\mathcal{A}_s|}{\max\limits_{k \in \mathcal{A}_s} \Psi_{sk} / \Omega_{sk}}}} \; , \; \frac{\sqrt{c} + \sqrt{\delta_s |\mathcal{A}_s|}}{C\, \sqrt{c}} \right\} \sum_{k \in \mathcal{A}_s} v_{sk}\, \lambda_k / \mu_k,$$

  according to Proposition 6.

Note that the initialization steps implemented by methods 4 and 5 are performed by the PaaS provider, since they require the knowledge of the time unit cost c of use of the physical servers, which is usually unknown by the SaaS providers. Method 3 can be implemented in practice only if the PaaS shares c with SaaSs. Methods 1–5 are suitable for a distributed implementation and require at most three messages exchange between each SaaS and PaaS (for providing the initial service demand, to communicate the effective number of physical servers, and finally set up the WS applications response time threshold and overall throughput).

6 Numerical Results

The solution algorithm proposed has been evaluated for a variety of system and workload configurations. Section 6.1 is devoted to quantitatively analyze the efficiency of the equilibria achieved by our approach, while the scalability is discussed in Section 6.2. Finally, Section 6.3 illustrates the equilibria properties for a medium size system by varying application performance parameters.

6.1 Equilibria Efficiency

To evaluate the efficiency of our algorithm we have considered a very large set of randomly generated instances. The number of SaaS providers has been varied between 100 and 1,000, the number of applications (evenly shared among SaaSs) between 1,000 and 10,000[1].

The performance parameters of the applications and infrastructural resources costs have been randomly generated uniformly in the ranges reported in Table 2 as in [8, 12, 31], considering also real applications [11], according to commercial fees applied by IaaS/PaaS Cloud providers [4, 33] and the energy costs available in the literature [28]. We have included in the time unit cost c of a physical server also the overhead of the cooling system according to the values reported in [28], varying also the cost of energy per kWh. Furthermore, the penalty values p_k have been set proportional to the revenues for single request execution α_k, $p_k = \gamma_k^1 \alpha_k$, where γ_k^1 has been randomly generated uniformly in the range $[5, 50]$, as in [42], while the upper bounds on the average response time thresholds guaranteed to the SaaS customers were set proportional to the request service demand $1/\mu_k$, i.e., $R_k^S = \gamma_k^2/\mu_k$, where γ_k^2 has been randomly generated uniformly in the range [100, 200], as in [13]. Finally, since request rejection has an important impact on SaaS provider reputation, we set $\lambda_k = 0.95\Lambda_k$.

α_k	[0.01, 1] \$/req	c	[0.03, 0.14] \$/hour	δ_s	[0.1, 1]
μ_k	[10, 1000] req/s	Λ_k	[100, 1000] req/s		
m_k	[1, 2]	v_{sk}	[1, 100]		

Table 2: Performance parameters and time unit cost ranges.

We denote with $\widetilde{x}$ any social optimum and with $\overline{x}$ the GNE found by the solution algorithm using methods 1–5. The efficiency has been measured in terms of the *Price of Anarchy* (PoA) evaluated as

$$PoA = \frac{\Pi(\overline{x})}{\Pi(\widetilde{x})}.$$

The PoA is a measure of the inefficiency due to PaaS and SaaSs selfish behavior with respect to the scenario where the *social optimum* is pursued. The metric is lower or equal to 1 (the greater the better).

[1]We have verified that the performance of our solution is not affected by the applications to SaaSs assignment cardinality (we varied the number of applications per SaaS in the range 1-100), both in terms of Price of Anarchy and execution time. Results are omitted for space limitation.

Random instances have been generated guaranteeing that the constraint (27) is active, which corresponds to the worst case situation for a Cloud system (heavy workload) and for the proposed methods which otherwise can determine the social optimum naively (e.g., if the servers are not saturated, PoA is always equal to 1 for method 3). This has been obtained by solving the social optimum problem with an infinite number of servers and then by setting N^P equal to ρ times the total number of servers actually used. In order to evaluate the robustness of our solution, ρ has been set equal to 0.9, 0.8 and 0.7, which corresponds to increasing workload conditions for the PaaS.

Results are reported in Tables 3–5. The figures reported in each table are the means computed on 10 different runs.

$(\|\mathcal{S}\|,\|\mathcal{A}\|)$	**Method 1**	**Method 2**	**Method 3**	**Method 4**	**Method 5**
(100,1000)	0.41	0.41	0.41	0.45	0.50
(200,2000)	0.39	0.39	0.39	0.45	0.49
(300,3000)	0.40	0.40	0.40	0.47	0.51
(400,4000)	0.40	0.41	0.40	0.48	0.53
(500,5000)	0.40	0.40	0.40	0.45	0.54
(600,6000)	0.40	0.40	0.40	0.49	0.50
(700,7000)	0.39	0.39	0.39	0.44	0.50
(800,8000)	0.40	0.40	0.40	0.47	0.52
(900,9000)	0.39	0.39	0.39	0.44	0.50
(1000,10000)	0.40	0.39	0.40	0.46	0.49
Average	0.40	0.40	0.40	0.46	0.51

Table 3: Solution methods efficiency, $\rho = 0.9$.

$(\|\mathcal{S}\|,\|\mathcal{A}\|)$	**Method 1**	**Method 2**	**Method 3**	**Method 4**	**Method 5**
(100,1000)	0.41	0.41	0.41	0.45	0.50
(200,2000)	0.39	0.39	0.39	0.45	0.48
(300,3000)	0.40	0.40	0.40	0.50	0.51
(400,4000)	0.40	0.41	0.40	0.48	0.54
(500,5000)	0.42	0.42	0.42	0.47	0.56
(600,6000)	0.40	0.40	0.40	0.45	0.51
(700,7000)	0.39	0.39	0.39	0.44	0.53
(800,8000)	0.40	0.41	0.40	0.47	0.52
(900,9000)	0.39	0.39	0.39	0.44	0.50
(1000,10000)	0.41	0.41	0.41	0.47	0.51
Average	0.40	0.40	0.40	0.46	0.52

Table 4: Solution methods efficiency, $\rho = 0.8$.

For every method the PoA does not depend significantly on the system size, neither on the

$(\vert\mathcal{S}\vert,\vert\mathcal{A}\vert)$	**Method 1**	**Method 2**	**Method 3**	**Method 4**	**Method 5**
(100,1000)	0.41	0.41	0.41	0.45	0.50
(200,2000)	0.39	0.40	0.39	0.45	0.48
(300,3000)	0.40	0.41	0.40	0.47	0.51
(400,4000)	0.40	0.41	0.40	0.50	0.54
(500,5000)	0.42	0.42	0.42	0.51	0.57
(600,6000)	0.40	0.40	0.40	0.44	0.51
(700,7000)	0.39	0.40	0.39	0.44	0.50
(800,8000)	0.40	0.41	0.40	0.47	0.56
(900,9000)	0.39	0.40	0.39	0.44	0.50
(1000,10000)	0.40	0.40	0.40	0.46	0.49
Average	0.40	0.41	0.40	0.46	0.51

Table 5: Solution methods efficiency, $\rho = 0.7$.

system workload conditions. Furthermore, methods 1–3 perform similarly, the PoA is around 0.4 (i.e., on average the percentage difference of the sum of the payoff functions with respect to the social optimum is lower than 60%). This means that the heuristic solution based on the utilization threshold proposed in the literature (implemented by method 2) achieves the same results of random initialization (implemented by method 1). Vice versa, methods 4 and 5, based on the analysis of the KKT system of the social optimum problem, allow to improve the PoA by 15% and 25%, respectively.

6.2 Algorithms Scalability

The scalability of our approach has been evaluated performing tests on a VMWare virtual machine based on Ubuntu 11.04 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. KNITRO 8.0 has been used as nonlinear optimization solver, which can exploit the multi-core architecture of our system. We have considered a very large set of randomly generated instances obtained as in the previous Section varying the model parameters according to the ranges reported in Table 2.

Table 6 reports the average execution time required by our methods for problem instances of different sizes. As in the previous Section, the average values reported in the table have always been computed by considering 10 instances with the same size. Results show that the proposed methods are very efficient and can solve problem instances of maximum size in less than one minute. Usually in Cloud systems resource allocation is performed periodically on a hourly basis [1, 14, 17]. Hence, a social optimum solution can be computed directly by the PaaS only for problem instances including 700 SaaSs and 7,000 WS applications which are not very significant in practice.

($\vert\mathbb{S}\vert$,$\vert\mathcal{A}\vert$)	**Social Opt. Solution**	**Method 1**	**Method 2**	**Method 3**	**Method 4**	**Method 5**
(100,1000)	132.80	0.74	0.74	0.80	0.60	0.80
(200,2000)	364.70	3.35	3.30	4.30	3.10	3.30
(300,3000)	632.50	3.31	3.31	5.31	6.31	7.31
(400,4000)	972.90	5.01	5.10	5.20	3.90	4.10
(500,5000)	1551.50	6.53	6.90	7.10	7.80	7.80
(600,6000)	1940,00	7.93	8.32	8.51	9.42	10.18
(700,7000)	2452.10	9.32	9.92	10.42	11.87	12.33
(800,8000)	3224.40	11.44	11.28	11.27	10.07	9.36
(900,9000)	3860.70	14.21	13.60	12.65	11.50	11.11
(1000,10000)	6100.00	41.60	36.60	43.60	44.60	37.60

Table 6: Execution times (s).

6.3 Equilibria Sharing Analysis

The aim of this Section is to analyze how the equilibrium changes by varying the game parameters. The results have been obtained by Method 5 only, since the methods proposed do not differ significantly in terms of execution time and Method 5 performs better in terms of PoA.

In particular, we considered three SaaSs offering five heterogeneous applications each. If not differently stated we set $\alpha_k = 0.5$ \$/req, $\mu_k = 10k$ req/s, $m_k = 1.5$, $c = 0.09$ \$/h, $\Lambda_k = 450$ req/s, and $\delta_s = 0.45$ (i.e., we considered the midpoint of the random intervals reported in Table 2). For the sake of simplicity, we considered the following application to SaaS provider assignment $\mathcal{A}_1 = \{1, \dots, 5\}$, $\mathcal{A}_2 = \{6, \dots, 10\}$, $\mathcal{A}_3 = \{11, \dots, 15\}$.

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. We will investigate how the parameter change will affect: (i) the throughput ratio of WS application 1, i.e., X_1/Λ_1, (ii) the throughput of the remaining applications for the first provider (i.e., $\sum_{k=2}^{5} X_k$) and the total throughput of the remaining providers (i.e., $\sum_{k=6}^{15} X_k$), (iii) the number of servers of the first SaaS provider N_1, and (iv) WS application 1 average response time R_1.

Figures 2–5 show the results we achieved by varying Λ_1 in $[495, 1165]$ req/s: As Λ_1 increases, the number of servers used by the SaaS provider 1 increases also linearly (Figure 4), while R_1 decreases non-linearly improving by around 57% (Figure 5). Vice versa, X_1/Λ_1 ratio is not affected significantly and stays close to 1 (Figure 2), as the overall throughput of the remaining applications (Figure 3).

Figures 6–9 analyze how the GNE changes by varying application 1 maximum service rate μ_1 (the range $[60, 510]$ req/s has been considered). If the maximum service rate increases, the service time required to process each WS application 1 request decreases and the overall capacity required to process WS application 1 decreases accordingly (Figures 8 and 9). Vice versa, application throughput is not affected (see Figures 6 and 7).

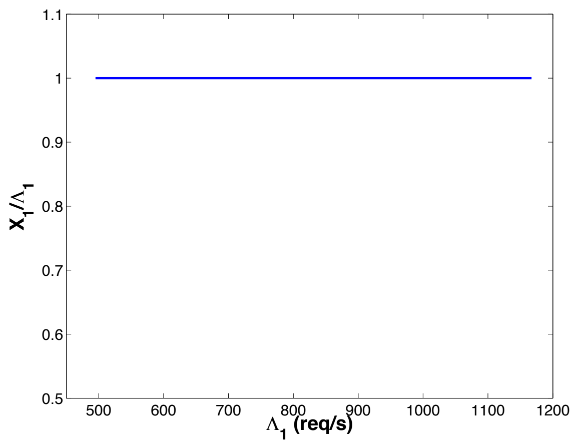

Figure 2: X_1/Λ_1 ratio with varying application 1 incoming workload.

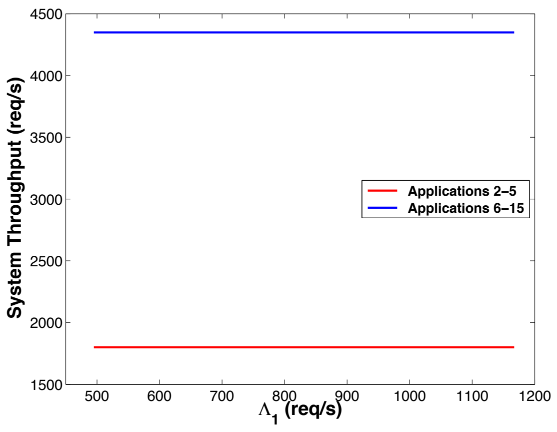

Figure 3: Application 2–5 and 6–15 throughput with varying application 1 incoming workload.

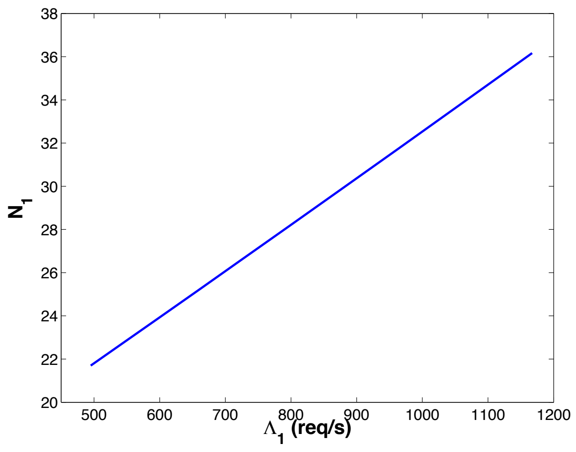

Figure 4: SaaS provider 1 number of servers with varying application 1 incoming workload.

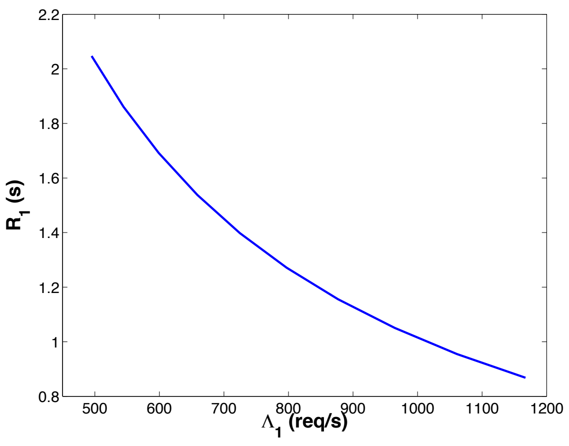

Figure 5: WS application 1 average response time with varying application 1 incoming workload.

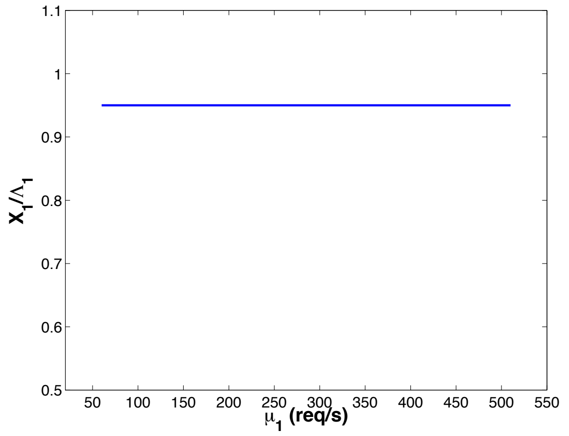

Figure 6: X_1/Λ_1 ratio with varying application 1 maximum service rate.

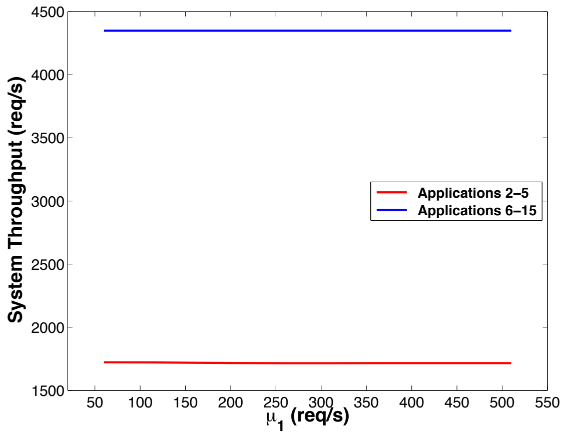

Figure 7: Application 2–5 and 6–15 throughput with varying application 1 maximum service rate.

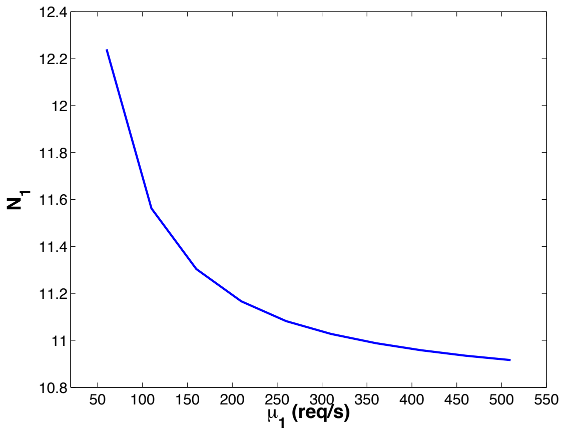

Figure 8: SaaS provider 1 number of servers with varying application 1 maximum service rate.

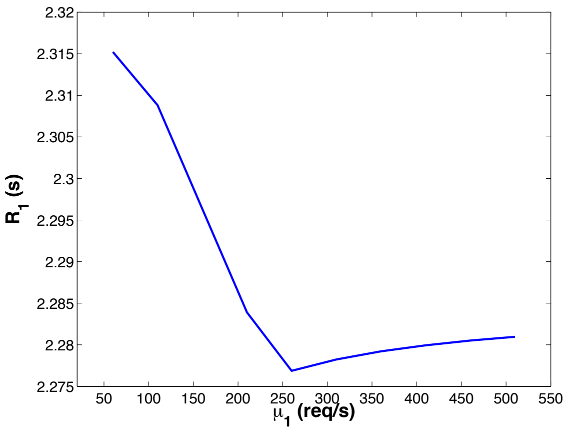

Figure 9: WS application 1 average response time with varying application 1 maximum service rate.

Surprisingly, varying the revenue for application 1 single request execution (α_1 varied in $[0.17, 1.02]$ \$/req) has no impact on the metrics under analysis, see Figures 10–13 (however, recall that WS application rejection penalty p_1 is proportional to α_1).

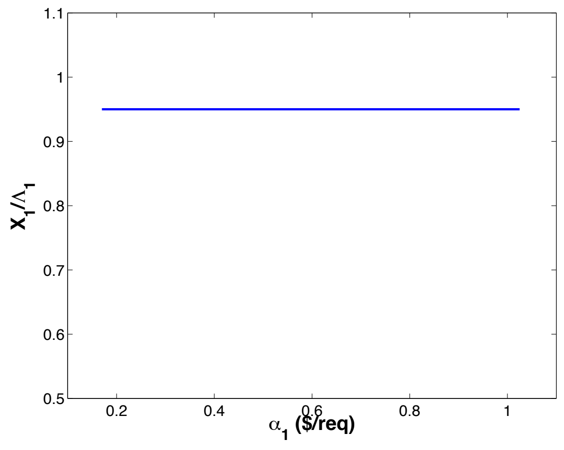

Figure 10: X_1/Λ_1 ratio with varying application 1 per request revenue.

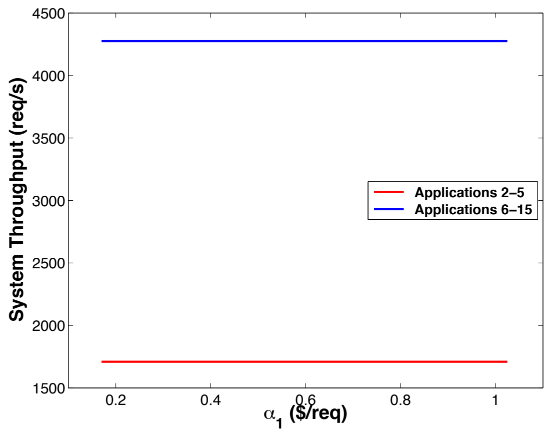

Figure 11: Application 2–5 and 6–15 throughput with varying application 1 per request revenue.

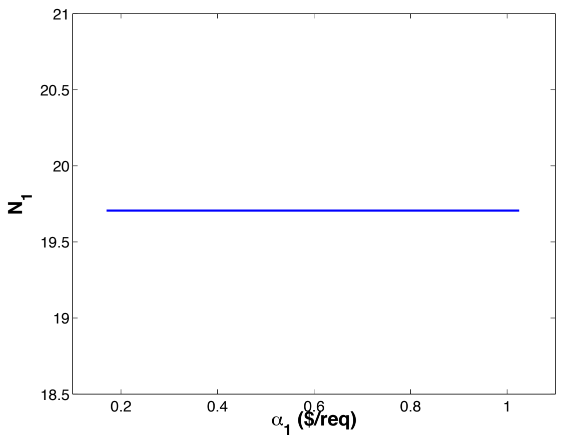

Figure 12: SaaS provider 1 number of servers with varying application 1 per request revenue.

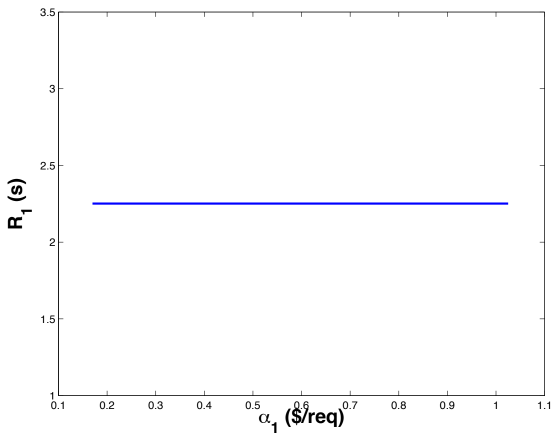

Figure 13: WS application 1 average response time with varying application 1 per request revenue.

Finally, we varied SaaS provider 1 utility function slope δ_1 in the range [0.2, 1]. As δ_1 increases, the metrics under analysis remain almost constant but change abruptly when $\delta_1 = 0.82$ (see Figures 14–17). For that value, the ratio X_1/Λ_1 drops suddenly from 100% to 95% (the minimum achievable value), which is very unintuitive.

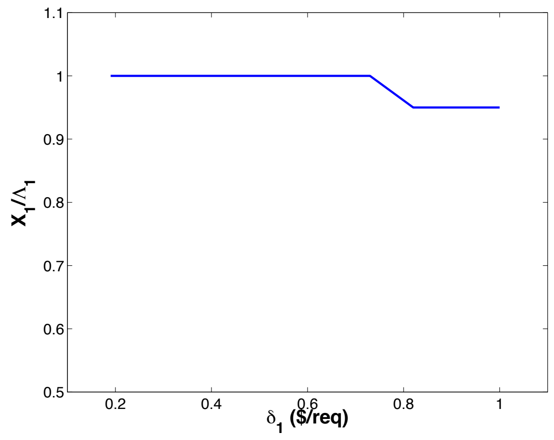

Figure 14: X_1/Λ_1 ratio with varying application 1 per request revenue.

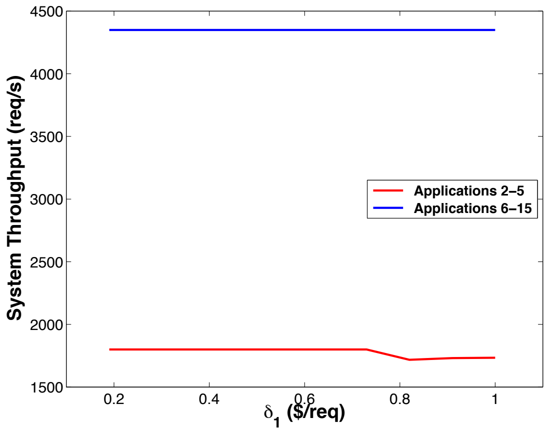

Figure 15: Application 2–5 and 6–15 throughput with varying application 1 per request revenue.

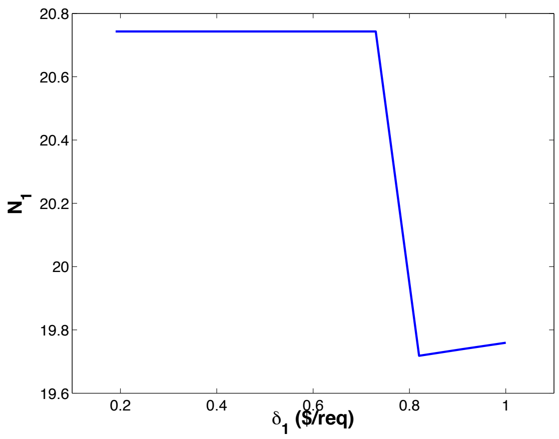

Figure 16: SaaS provider 1 number of servers with varying application 1 per request revenue.

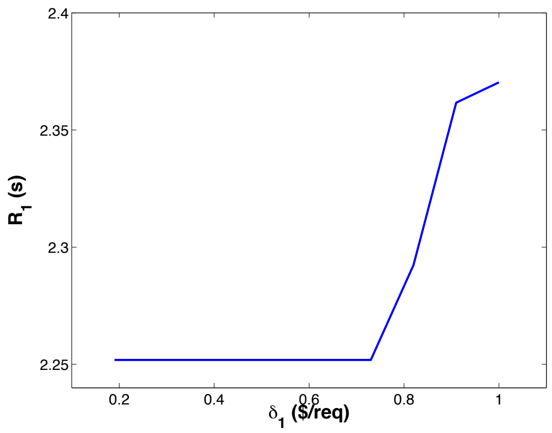

Figure 17: WS application 1 average response time with varying application 1 per request revenue.

7 Conclusions

We proposed a game-theoretic approach for the run-time management of a PaaS provider capacity among multiple competing SaaSs. The cost model consists of a class of utility functions which include revenues and penalties incurred depending on the achieved performance level and the energy costs associated with PaaS resources.

The solution methods proposed have been evaluated for a variety of system and workload configurations and have been demonstrated effective even for very large size problem instances. Systems up to thousands of applications can be managed very efficiently.

Results have shown that solution methods proposed by the literature and the one based on random initialization perform similarly from the PoA point of view. Vice versa, methods 4 and 5 which have been obtained through the analytical analysis of the social optimum problem, allow to improve PoA by 15–25%. Future work will extend the proposed solutions to consider multiple time-scales for performing resource allocation ranging from few minutes to one hour. Finally, the opportunity to allocate SaaS applications on multiple PaaS providers will be also investigated.

Acknoledgements

The authors wish to thank the anonymous referees for the careful reviews and valuable comments, which helped them to improve the presentation form of the paper.

References

[1] J. Almeida, V. Almeida, D. Ardagna, I. Cunha, C. Francalanci, and M. Trubian. Joint admission control and resource allocation in virtualized servers. *Journal of Parallel and Distributed Computing*, 70(4):344–362, 2010.

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

[3] Amazon Inc. Amazon EC2 Dedicated Instances. `http://aws.amazon.com/dedicated-instances/`.

[4] Amazon Inc. Amazon Elastic Cloud. `http://aws.amazon.com/ec2/`.

[5] Amazon Inc. AWS GovCloud (US). `http://aws.amazon.com/govcloud-us/`.

[6] M. Andreolini and S. Casolari. Load prediction models in web-based systems. In *Proceedings of the 1st international conference on Performance evaluation methodolgies and tools*, valuetools '06, New York, NY, USA, 2006. ACM.

[7] J. Anselmi and B. Gaujal. The price of forgetting in parallel and non-observable queues. *Performance Evaluation*, 68(12):1291–1311, 2011.

[8] J. Anselmi and I. M. Verloop. Energy-aware capacity scaling in virtualized environments with performance guarantees. *Performance Evaluation*, 68(11):1207–1221, 2011.

[9] D. Ardagna, S. Casolari, M. Colajanni, and B. Panicucci. Dual time-scale distributed capacity allocation and load redirect algorithms for cloud systems. *Journal of Parallel and Distributed Computing*, 72(6):796 – 808, 2012.

[10] D. Ardagna, B. Panicucci, and M. Passacantando. A game theoretic formulation of the service provisioning problem in cloud systems. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 177–186, New York, NY, USA, 2011. ACM.

[11] D. Ardagna, B. Panicucci, and M. Passacantando. Generalized Nash equilibria for the service provisioning problem in cloud systems. *IEEE Transactions on Services Computing*, in press.

[12] D. Ardagna, B. Panicucci, M. Trubian, and L. Zhang. Energy-aware autonomic resource allocation in multitier virtualized environments. *IEEE Transactions on Services Computing*, 5(1):2–19, 2012.

[13] D. Ardagna and B. Pernici. Adaptive service composition in flexible processes. *IEEE Transactions on Software Engineering*, 33(6):369–384, 2007.

[14] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. Above the clouds: A Berkeley view of cloud computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009.

[15] M. N. Bennani and D. A. Menasce. Resource allocation for autonomic data centers using analytic performance models. In *Proceedings of the Second International Conference on Automatic Computing*, ICAC '05, pages 229–240, Washington, DC, USA, 2005. IEEE Computer Society.

[16] G. Bigi, M. Castellani, M. Pappalardo, and M. Passacantando. Existence and solution methods for equilibria. *European Journal of Operational Research*, 227(1):1–11, 2013.

[17] R. Birke, L. Chen, and E. Smirni. Data centers in the cloud: A large scale performance study. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 336 –343, 2012.

[18] J. Cao, K. Hwang, K. Li, and A. Zomaya. Optimal multiserver configuration for profit maximization in cloud computing. *IEEE Transactions on Parallel and Distributed Systems*, 24(6):1087–1096, 2013.

[19] E. Cavazzuti, M. Pappalardo, and M. Passacantando. Nash equilibria, variational inequalities, and dynamical systems. *Journal of Optimization Theory and Applications*, 114(3):491–506, 2002.

[20] A. Chandra, W. Gong, and P. Shenoy. Dynamic resource allocation for shared data centers using online measurements. In *Proceedings of ACM SIGMETRICS*, pages 300–301, NY, USA, 2003. ACM.

[21] L. Cherkasova and P. Phaal. Session-based admission control: a mechanism for peak load management of commercial web sites. *IEEE Transactions on Computers*, 51(6):669–685, 2002.

[22] A. Croll. Cloud performance from the end user perspective. `http://www.bitcurrent.com/download/cloud-performance-from-the-end-user-perspective/`.

[23] G. Debreu. A social equilibrium existence theorem. *Proceedings of the National Academy of Sciences of the USA*, 38(10):886–893, 1952.

[24] M. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis, and A. Vakali. Cloud computing: Distributed internet computing for IT and scientific research. *IEEE Internet Computing*, 13(5):10–13, 2009.

[25] D. C. Dynamic. Industry census 2012: Emerging data center markets. `https://www.datacenterdynamics.com/blogs/industry-census-2012-emerging-data-center-markets`, 2013.

[26] F. Facchinei and C. Kanzow. Generalized Nash equilibrium problems. *Annals of Operations Research*, 175(1):177–211, 2010.

[27] F. Facchinei, V. Piccialli, and M. Sciandrone. Decomposition algorithms for generalized potential games. *Computational Optimization and Applications*, 50(2):237–262, 2011.

[28] Greenpeace. How clean is your cloud? `http://www.greenpeace.org/international/Global/international/publications/climate/2012/iCoal/HowCleanisYourCloud.pdf`.

[29] M. Haviv and T. Roughgarden. The price of anarchy in an exponential multi-server. *Operations Research Letters*, 35(4):421–426, 2007.

[30] E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. In *Proceedings of the 16th annual conference on Theoretical aspects of computer science*, STACS'99, pages 404–413, Berlin, Heidelberg, 1999. Springer-Verlag.

[31] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang. Power and performance management of virtualized computing environments via lookahead control. In *Proceedings of the 2008 International Conference on Autonomic Computing*, ICAC '08, pages 3–12, Washington, DC, USA, 2008. IEEE Computer Society.

[32] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik. *Quantitative system performance: computer system analysis using queueing network models.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1984.

[33] Microsoft. Windows azure. `http://msdn.microsoft.com/en-us/library/windowsazure/dd163896`.

[34] D. Monderer and L. Shapley. Potential games. *Games and Economic Behaviour*, 14(1):124–143, 1996.

[35] J. Nash. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295, 1951.

[36] A. Parekh and R. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the multiple node case. *IEEE/ACM Transactions on Networking*, 2(2):137–150, 1994.

[37] J. B. Rosen. Existence and uniqueness of equilibrium points for concave n-person games. *Econometrica*, 33(3):520–534, 1965.

[38] B. Urgaonkar and P. Shenoy. Sharc: managing CPU and network bandwidth in shared clusters. *IEEE Transactions on Parallel and Distributed Systems*, 15(1):2–17, 2004.

[39] VMware. ESX server performance and resource management for CPU-intensive workloads. Technical report, VMware, 2005.

[40] A. Wolke and G. Meixner. Twospot: A cloud platform for scaling out web applications dynamically. In E. Di Nitto and R. Yahyapour, editors, *Towards a Service-Based Internet*, volume 6481 of *Lecture Notes in Computer Science*, pages 13–24. Springer Berlin / Heidelberg, 2010.

[41] B. Yolken and N. Bambos. Game based capacity allocation for utility computing environments. In *Proceedings of the 3rd International Conference on Performance Evaluation Methodologies and Tools*, ValueTools '08, pages 1–8, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[42] L. Zhang and D. Ardagna. SLA based profit optimization in autonomic computing systems. In *Proceedings of the 2Nd International Conference on Service Oriented Computing*, ICSOC '04, pages 173–182, New York, NY, USA, 2004. ACM.

[43] Q. Zhang, Q. Zhu, M. Zhani, and R. Boutaba. Dynamic service placement in geographically distributed clouds. In *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*, pages 526–535, 2012.

[44] X. Zhu, D. Young, B. Watson, Z. Wang, J. Rolia, S. Singhal, B. McKee, C. Hyser, D. Gmach, R. Gardner, T. Christian, and L. Cherkasova. 1000 islands: an integrated approach to resource management forvirtualized data centers. *Cluster Computing*, 12(1):45–57, 2009.