Two-Phase Virtual Machine Placement Algorithms for Cloud Computing: An Experimental Evaluation under Uncertainty

Nabil Chamas
Polytechnic School
National University of Asunción
San Lorenzo, Paraguay
nabilchamasi@gmail.com

Fabio López-Pires
Itaipu Technological Park
National University of Asunción
Hernandarias, Paraguay
fabio.lopez@pti.org.py

Benjamín Barán
Polytechnic School
National University of Asunción
San Lorenzo, Paraguay
bbaran@pol.una.py

***Abstract*—Cloud computing providers must support requests for resources in dynamic environments, considering service elasticity and overbooking of physical resources. Due to the randomness of requests, Virtual Machine Placement (VMP) problems should be formulated under uncertainty. In this context, a renewed formulation of the VMP problem is presented, considering the optimization of four objective functions: (i) power consumption, (ii) economical revenue, (iii) resource utilization and (iv) reconfiguration time. To solve the presented formulation, a two-phase optimization scheme is considered, composed by an online incremental VMP phase (iVMP) and an offline VMP reconfiguration (VMPr) phase. An experimental evaluation of five algorithms taking into account 400 different scenarios was performed, considering three VMPr Triggering and two VMPr Recovering methods as well as three VMPr resolution alternatives. Experimental results indicate which algorithm outperformed the other evaluated algorithms, improving the quality of solutions in a scenario-based uncertainty model considering the following evaluation criteria: (i) average, (ii) maximum and (iii) minimum objective function costs.**

***Index Terms*—Virtual Machine Placement, Cloud Computing, Overbooking, Elasticity, Uncertainty, Incremental VMP, VMP Reconfiguration.**

1. Introduction

Achieving an efficient resource management in cloud computing datacenters presents several research challenges. This work focuses on one of the most studied problems for resource allocation in cloud computing datacenters: the process of selecting which requested *virtual machines* (VMs) should be hosted at each available *physical machine* (PM) of a cloud computing infrastructure, commonly known as *Virtual Machine Placement* (VMP). A complex *Infrastructure as a Service* (IaaS) environment for VMP problems is considered, taking into account service elasticity and overbooking of physical resources [1]. In order to model this complex IaaS environment for VMP problems, cloud services (i.e. a set of inter-related VMs) are considered instead of isolated VMs.

For IaaS customers, cloud computing resources often appear to be unlimited and can be provisioned in any quantity at any required time [2]. Consequently, this work considers a basic federated-cloud deployment architecture for the VMP problem.

It is important to consider that more than 60 objective functions have been proposed for VMP problems [3]. Consequently a multi-objective formulation of the VMP problem is presented, considering the simultaneous optimization of the following objective functions: (i) power consumption, (ii) economical revenue, (iii) resource utilization and (iv) reconfiguration time.

Due to the randomness of customer requests, VMP problems should be formulated under uncertainty. This work presents a scenario-based uncertainty approach for modeling uncertain parameters, evaluating 400 scenarios. At the same time, this work presents a VMP problem taking into account the most complex IaaS environment identified in [1], that considers both types of service elasticity and both types of overbooking of physical resources.

It is worth remembering that VMP is a NP-Hard combinatorial optimization problem [4]. From an IaaS provider perspective, the VMP problem could be formulated as both online and offline optimization problems [3].

It is important to consider that online decisions made along the operation of a cloud computing infrastructure may negatively affect the quality of obtained solutions of VMP problems when comparing to offline decisions [5].

To improve the quality of solutions obtained by online algorithms, VMP problems could be formulated as a two-phase optimization problem, combining advantages of online and offline formulations for IaaS environments [5]. In this context, VMP problems could be decomposed into two different sub-problems: *incremental VMP* (iVMP) and *VMP reconfiguration* (VMPr), as presented in Figure 1.

The iVMP sub-problem is considered for dynamic arriving requests, where VMs could be created, modified and removed at runtime. Consequently, this sub-problem should be formulated as an online problem and solved with short time constraints, where existing heuristics could be reasonably appropriate. Additionally, the VMPr sub-problem is considered for improving the quality of solutions obtained in the iVMP phase, reconfiguring the placement through VM migration. This sub-problem could be formulated offline,

978-1-5386-3057-0/17/$31.00

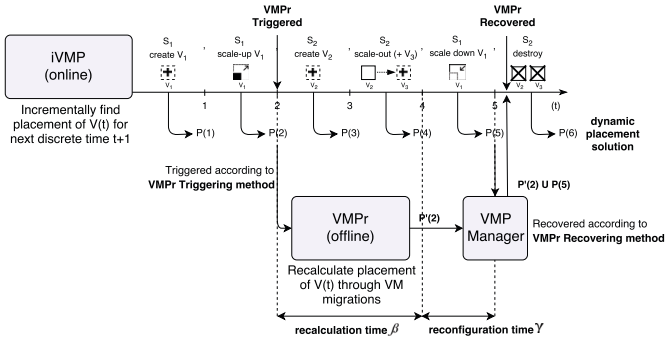

Figure 1. Two-phase optimization scheme for VMP problems considered in this work, presenting a basic example with a placement recalculation time of $\beta = 2$ (from $t = 2$ to $t = 4$) and a placement reconfiguration time of $\gamma = 1$ (from $t = 4$ to $t = 5$).

where alternative solution techniques could result more suitable (e.g. meta-heuristics).

The VMPr phase is triggered according to a given VMPr Triggering method. It is important to notice that the placement recalculated by the VMPr phase is potentially obsolete, due to its offline nature. In fact, while the VMPr is making its calculation, the iVMP still may receive and serve arriving requests, making obsolete the VMPr calculated solution; therefore, the recalculated placement must be recovered accordingly using a VMPr Recovering method, before complete reconfiguration is performed.

This work presents an experimental evaluation of five algorithms considering three VMPr algorithms, three VMPr Triggering algorithms and two VMPr Recovering methods.

The remainder of this paper is structured as follows: the uncertain VMP problem formulation is presented in Section 2, while Section 3 presents details on the design and implementation of evaluated alternatives to solve the presented renewed formulation of the VMP problem. Experimental results are summarized in Section 4. Finally, conclusions and future work are left to Section 5.

2. Uncertain VMP Formulation

This section presents a formulation of the VMP problem under uncertainty considering a two-phase scheme for the optimization of the following objective functions: (i) power consumption, (ii) economical revenue, (iii) resource utilization and (iv) placement reconfiguration time. According to the taxonomy presented in [3], this work focuses on a provider-oriented VMP for federated-cloud deployments, considering a combination of two types of formulations: (i) online (i.e. iVMP) and (ii) offline (i.e. VMPr).

The VMP problem presented in this work takes into account a complex IaaS environment that considers service elasticity, including both vertical and horizontal scaling of cloud services, as well as overbooking of physical resources, including both server and networking resources [1].

2.1. Complex IaaS Environment

The renewed formulation of the VMP problem models a complex IaaS environment, composed by available PMs and VMs requested at each discrete time t, considering the following input data: (i) a set of n available PMs and their specifications (1) and (ii) a set of $m(t)$ VMs requested, at each discrete time t, and their specifications (2).

The set of PMs owned by the IaaS provider is represented as a matrix $H \in \mathbb{R}^{n\times(r+2)}$, as presented in (1). Each PM H_i is represented by r different physical resources. This work considers $r = 3$ physical resources (Pr_1 to Pr_3): CPU [EC2 - Compute Unit (ECU)], RAM [GB] and network capacity [Mbps]. The maximum power consumption [W] is also considered. It is important to mention that the presented notation is general enough to include more characteristics associated to physical resources as *Solid State Drive* (SSD), *Graphical Processing Unit* (GPU) or storage, just to cite a few. Finally, considering that an IaaS provider could own more than one cloud datacenter, PMs notation also includes a datacenter identifier c_i, i.e.

$$H = \begin{bmatrix} Pr_{1,1} & \dots & Pr_{r,1} & pmax_1 & c_1 \\ \dots & \dots & \dots & \dots & \dots \\ Pr_{1,n} & \dots & Pr_{r,n} & pmax_n & c_n \end{bmatrix} \quad (1)$$

where: $Pr_{k,i}$ is the physical resource k on H_i, $1 \leq k \leq r$; $pmax_i$ is the maximum power consumption of H_i in [W]; c_i is the datacenter identifier of H_i, $1 \leq c_i \leq c_{max}$; n is the total number of PMs.

Clearly, the set of PMs H could be modeled as a function of time t, considering PM crashes [6], maintenance or even deployment of new hardware.

In the complex environment considered in this work, the IaaS provider dynamically receives requests of cloud services for placement (i.e. a set of inter-related VMs) at each discrete time t. A cloud service S_b is composed by a set of VMs, where each VM may be located for execution in different cloud computing datacenters according to customer preferences or requirements.

The set of VMs requested by customers at each discrete time t is represented as a matrix $V(t) \in \mathbb{R}^{m(t)\times(r+2)}$, as presented in (2). In this work, each VM V_j requires $r = 3$ different virtual resources ($Vr_{1,j}(t)$-$Vr_{3,j}(t)$): CPU [ECU], RAM memory [GB] and network capacity [Mbps]. Additionally, a cloud service identifier b_j is considered, as well as an economical revenue R_j [$] associated to each VM V_j. As mentioned before, the presented notation could represent any other set of r resources.

The requested VMs try to lease the requested virtual resources for an unknown period of discrete time.

$$V(t) = \begin{bmatrix} Vr_{1,1}(t) & \dots & Vr_{r,1}(t) & b_1 & R_1(t) \\ \dots & \dots & \dots & \dots & \dots \\ Vr_{1,m(t)}(t) & \dots & Vr_{r,m(t)}(t) & b_m(t) & R_{m(t)}(t) \end{bmatrix} \quad (2)$$

where: $Vr_{k,j}(t)$ is the virtual resource k on V_j, $1 \leq k \leq r$; b_j is the service identifier of V_j; $R_j(t)$ is the economical revenue for allocating V_j in [$]; $m(t)$ is the number of VMs at each discrete time t, $1 \leq m(t) \leq m_{max}$.

2.2. Incremental VMP (iVMP)

In online algorithms for solving the presented VMP problem, placement decisions are performed at each discrete

time t. The formulation of the presented iVMP (online) problem is based on [5] and could be enunciated as:
Given a complex IaaS environment composed by a set of PMs (H), a set of active VMs already requested before time t ($V(t)$), and the current placement of VMs into PMs (i.e. $x(t)$), it is sought an incremental placement of $V(t)$ into H for the discrete time $t+1$ ($i.e. x(t+1)$) without migrations, satisfying the problem constraints and optimizing the considered objective functions.

Additionally to the complex IaaS environment, the iVMP problem receives the following input data: (i) information about the utilization of resources of each active VM at each discrete time t (3) and (ii) the current placement at each discrete time t (i.e. $x(t)$) (4).

In most situations, virtual resources requested by cloud services are dynamically used, giving space to re-utilization of idle resources that were already reserved. Information about the utilization of virtual resources at each discrete time t is required in order to model a dynamic VMP environment where IaaS providers consider overbooking of both server and networking physical resources.

Resource utilization of each VM V_j at each discrete time t is represented in matrix $U(t) \in \mathbb{R}^{m(t)\times r}$, as presented (3):

$$U(t) = \begin{bmatrix} Ur_{1,1}(t) & \dots & Ur_{r,1}(t) \\ \dots & \dots & \dots \\ Ur_{1,m(t)}(t) & \dots & Ur_{r,m(t)}(t) \end{bmatrix} \quad (3)$$

where: $Ur_{k,j}(t)$ is the utilization ratio of $Vr_k(t)$ in V_j at each discrete time t.

The current placement of VMs into PMs ($x(t)$) represents VMs requested in the previous discrete time $t-1$ and assigned to PMs; consequently, the dimension of $x(t)$ is based on the number of VMs $m(t-1)$. Formally, the placement at each discrete time t is represented as a matrix $x(t) \in \{0,1\}^{m(t-1)\times n}$, as defined in (4):

$$x(t+1) = \begin{bmatrix} x_{1,1}(t+1) & \dots & x_{1,n}(t+1) \\ \dots & \dots & \dots \\ x_{m(t),1}(t+1) & \dots & x_{m(t),n}(t+1) \end{bmatrix} \quad (4)$$

where: $x_{j,i}(t) \in \{0,1\}$ indicates if V_j is allocated ($x_{j,i}(t) = 1$) or not ($x_{j,i}(t) = 0$) for execution in a PM H_i at a discrete time t (i.e., $x_{j,i}(t) : V_j \rightarrow H_i$).

The result of the iVMP phase at each discrete time t is an incremental placement $\Delta x(t)$ for the next time instant in such a way that $x(t+1) = x(t) + \Delta x(t)$. Clearly, the placement at $t+1$ is represented as a matrix $x(t+1) \in \{0,1\}^{m(t)\times n}$.

2.3. VMP Reconfiguration (VMPr)

An offline algorithm solves a VMP problem considering a static environment where VM requests do not change over time and considers migration of VMs between PMs. The formulation of the presented VMPr (offline) problem is based on [7], [8] and could be enunciated as:
Given a current placement of VMs into PMs ($x(t)$), it is sought a placement reconfiguration through migration of VMs between PMs for the discrete time t (i.e. $x'(t)$), satisfying the constraints and optimizing the considered objective functions.

The VMPr problem receives the current placement at discrete time t (i.e. $x(t)$) (4) as input data.

The result of the VMPr problem is a placement reconfiguration through migration of VMs between PMs for the discrete time t (denoted as $x'(t)$).

2.4. Constraints

2.4.1. Constraint 1: Unique Placement of VMs. A VM V_j should be allocated to run on a single PM H_i or alternatively located in another federated IaaS provider. Consequently, this placement constraint is expressed as:

$$\sum_{i=1}^{n} x_{j,i}(t) \leq 1 \quad (5)$$

$$\forall j \in \{1, \dots, m(t)\}, \text{i.e. for all VM } V_j.$$

2.4.2. Constraints 2-4: Overbooked Resources of PMs. A PM H_i must have sufficient available resources to meet the dynamic requirements of all VMs V_j that are allocated to run on H_i. It is important to remember that resources of VMs are dynamically used, giving space to re-utilization of idle resources that were already reserved. Re-utilization of idle resources could represent higher risk of unsatisfied demand in case utilization of resources increases in a short period of time. Therefore, providers need to reserve a percentage of idle resources as a protection (defined by a protection factor λ_k) in case overbooking is used. These constraints can be formulated as:

$$\sum_{j=1}^{m(t)} x_{j,i}(t) \Big\{ Vr_{k,j}(t) \times Ur_{k,j}(t) + \\ + \lambda_k \Big[Vr_{k,j}(t)(1 - Ur_{k,j}(t)) \Big] \Big\} \leq Pr_{k,i} \quad (6)$$

for every time slot t, $\forall i \in \{1, \dots, n\}$ and $\forall k \in \{1, \dots, r\}$, i.e. for each PM H_i and for each of the r considered resource, *where*: λ_k is the protection factor for $Vr_{k,j} \in$ [0,1]. Note that $\lambda_k = 0$ means full overbooking while $\lambda_k = 1$ means no-overbooking.

2.5. Objective Functions

Although in general some objective functions can be minimized while maximizing other objectives functions, in this work each of the considered objective functions are formulated in a minimization context.

2.5.1. Power Consumption Minimization. Based on Beloglazov et al. [9], this work models the power consumption of PMs considering a linear relationship with the CPU utilization of PMs, without taking into account PMs at alternative datacenters of the cloud federation. The power consumption minimization can be represented by the sum of the power consumption of each PM H_i (see Section 2.1), as defined in (7).

$$f_1(x,t) = \sum_{i=1}^{n}((pmax_i - pmin_i) \times Ur_{1,i}(t) + pmin_i) \times Y_i(t) \quad (7)$$

where: x is the evaluated solution of the problem; $f_1(x,t)$ is the total power consumption of PMs at instant t; $pmin_i$ is the minimum power consumption of a PM H_i. As suggested in [9], $pmin_i \approx pmax_i * 0.6$; $Y_i(t) \in \{0,1\}$: indicates if H_i is turned on $(Y_i(t) = 1)$ or not $(Y_i(t) = 0)$ at instant t.

2.5.2. Economical Revenue Maximization. For IaaS customers, cloud computing resources often appear to be unlimited and can be provisioned in any quantity at any required time t [2]. Consequently, this work considers a basic federated-cloud deployment architecture, where a main provider may support requested resources that are not able to be provided (e.g. a workload peak) by transparently leasing low-price resources from alternative datacenters owned by federated providers [10]. This leasing costs should be minimized in order to maximize economical revenue.

Equation (8) represents the mentioned leasing costs, defined as the sum of the total costs of leasing each VM V_j that is effectively allocated for execution on any PM of an alternative datacenter of the cloud federation. A provider must offer its idle resources to the cloud federation at lower prices than offered to customers in the actual cloud market for the federation to make sense. The pricing scheme may depend on the particular agreement between providers of the cloud federation [10]. For simplicity, this work considers that the main provider may lease requested resources (that are not able to provide) from the cloud federation at 70% $(\hat{X}_j = 0.7)$ of its market price $(R_j(t))$. This Leasing Cost $(LC(t))$ may be formulated as:

$$LC(t) = \sum_{j=1}^{m(t)}(R_j(t) \times X_j(t) \times \hat{X}_j) \quad (8)$$

where: $LC(t)$ is the total leasing costs at instant t; $\hat{X}_j$ indicates if V_j is allocated on the main provider $(\hat{X}_j = 0)$ or on an alternative datacenter of the cloud federation $(\hat{X}_j = 0.7)$.

It is important to note that $\hat{X}_j$ is not necessarily a function of time. The decision of locating a VM V_j on a federated provider is considered only in the placement process, with no possible migrations between different IaaS providers.

Additionally, overbooked resources may incur in unsatisfied demand of resources at some periods of time, causing Quality of Service (QoS) degradation, and consequently Service Level Agreement (SLA) violations with economical penalties. This economical penalties should be minimized for an economical revenue maximization. Based on the workload independent QoS metric presented in [9], formalized in SLAs, this work presents (9) to represent total economical penalties for SLA violations, defined as the sum of the total proportional penalties costs for unsatisfied demand of resources.

$$EP(t) = \sum_{j=1}^{m(t)}\left(\sum_{k=1}^{r} Rr_{k,j}(t) \times \Delta r_{k,j}(t) \times X_j(t) \times \phi_k\right) \quad (9)$$

where: $EP(t)$ is the total economical penalties at instant t; $Rr_{k,j}(t)$ is the economical revenue for attending $Vr_{k,j}(t)$; $\Delta r_{k,j}(t)$ is the ratio of unsatisfied resource k at instant t; $\Delta r_{k,j}(t) = 1$ means no unsatisfied resource; $\Delta r_{k,j}(t) = 0$ means resource k is unsatisfied in 100%; ϕ_k is the penalty factor for resource k, $\phi_k \geq 1$.

In this work, the maximization of the total economical revenue that an IaaS provider receives is achieved by minimizing the total costs of leasing resources from alternative datacenters of the cloud federation as well as the total economical penalties for SLA violations, as presented in (10), i.e.

$$f_2(x,t) = LC(t) + EP(t) \quad (10)$$

where: $f_2(x,t)$ is the total economical expediture of the main IaaS provider at instant t.

2.5.3. Resources Utilization Maximization. An efficient utilization of resources is a relevant resource management challenge to be addressed by IaaS providers. This work presents a maximization of the resource utilization by minimizing the average ratio of wasted resources on each PM H_i (i.e. resources that are not allocated to any VM V_j). This objective function is presented in (11).

$$f_3(x,t) = \frac{\sum_{i=1}^{n}\left[1 - \left(\frac{\sum_{k=1}^{r} Ur_{k,i}(t)}{r}\right)\right] \times Y_i(t)}{\sum_{i=1}^{n} Y_i(t)} \quad (11)$$

where: $f_3(x,t)$ is the average ratio of wasted resources at instant t.

2.5.4. Reconfiguration Time Minimization. Performance degradation may occur when migrating VMs between PMs [11]. Logically, it is desirable that the time of placement reconfiguration by VM migration is kept to a minimum possible. As explained in [11], the time that a VM takes to be migrated from one PM to another could be estimated as the ratio between the total amount of RAM memory to be migrated and the capacity of the network channel.

Inspired in [11], once a placement reconfiguration is accepted in the VMPr phase, all VM migrations are assumed to be performed in parallel through a management network exclusively used for these actions, increasing 10% CPU utilization in VMs being migrated. Consequently, the minimization of the (maximum) reconfiguration time could be achieved by minimizing the maximum amount of memory to be migrated from one PM H_i to another $H_{i'}$ $(i \neq i')$.

Equation (12) is presented to minimize the maximum amount of RAM memory that must be moved between PMs.

$$f_4(x,t) = \max(MT_{i,i'}) \quad \forall i, i' \in \{1,\ldots,n\} \quad (12)$$

where: $f_4(x,t)$ is the network traffic overhead for VM migrations at instant t; $MT_{i,i'}$ is the total amount of RAM memory to be migrated from PM H_i to $H_{i'}$.

2.6. Normalization and Scalarization Methods

As a consequence of experimental results obtained in a previous work by the authors [7] for VMP problems optimizing multiple objective functions, even in a many-objective optimization context for cloud computing datacenters, instead of calculating a whole Pareto set approximation, a scalarization method (e.g. minimum distance to origin) is suggested to combine all considered objective functions into a single objective function, therefore solving the studied problem considering a Multi-Objective problem solved as Mono-Objective (MAM) approach [3]. Consequently, each of the considered objective function must be formulated in a single optimization context (in this case, minimization) and each objective function cost must be normalized to be comparable and combinable as a single objective. This work normalizes objective functions cost by calculating $\hat{f}_i(x,t)$.

$$\hat{f}_i(x,t) = \frac{f_i(x,t) - f_i(x,t)_{min}}{f_i(x,t)_{max} - f_i(x,t)_{min}} \tag{13}$$

where: $\hat{f}_i(x,t)$ is the normalized cost of objective function $f_i(x,t)$ at instant t; $f_i(x,t)$ is the cost of original objective function; $f_i(x,t)_{min}$ and $f_i(x,t)_{max}$ are the minimum and maximum possible cost for $f_i(x,t)$ respectively.

Finally, the presented normalized objective functions are combined into a single objective considering a minimum Euclidean distance to the origin, expressed as:

$$F(x,t) = \sqrt{\sum_{i=1}^{q} \hat{f}_i(x,t)^2} \tag{14}$$

where: $F(x,t)$ is a unique function combining each $\hat{f}_i(x,t)$ at instant t; q is the number of objective functions. In this work $q = 4$.

2.7. Scenario-based Uncertainty Modeling

In this work, uncertainty is modeled through a finite set of well-defined scenarios S [12], where the following uncertain parameters are considered: (i) virtual resources capacities (vertical elasticity), (ii) number of VMs that compose cloud services (horizontal elasticity), (iii) utilization of CPU and RAM memory virtual resources and (iv) utilization of networking virtual resources (both relevant for overbooking). For each scenario $s \in S$, a temporal average value of the objective function $F(x,t)$ presented in (14) is calculated as:

$$\overline{f_s(x,t)} = \frac{\sum_{t=1}^{t_{max}} F(x,t)}{t_{max}} \tag{15}$$

where: $\overline{f_s(x,t)}$ is the temporal average of combined objective function for all discrete time instants t in scenario $s \in S$; t_{max} is the duration of a scenario in time instants.

As previously described, when parameters are uncertain, it is important to find solutions that are acceptable for any (or most) considered scenario $s \in S$. This work considers minimization of the following criteria to select among solutions from different evaluated alternatives as: (i) average [12], (ii) maximum [12] and (iii) minimum objective function costs:

$$F_1 = \overline{F(x,t)} = \frac{\sum_{s=1}^{|S|} \overline{f_s(x,t)}}{|S|} \tag{16}$$

$$F_2 = \max_{s \in S}(\overline{f_s(x,t)}) \tag{17}$$

$$F_3 = \min_{s \in S}(\overline{f_s(x,t)}) \tag{18}$$

where: F_1 is the average $\overline{f_s(x,t)}$ for all scenarios $s \in S$ [12]; F_2 represents the maximum $\overline{f_s(x,t)}$ considering all scenarios $s \in S$ while F_3 is the minimum $\overline{f_s(x,t)}$ considering all scenarios $s \in S$.

3. Evaluated Algorithms

This work evaluates five algorithms, presented in Table 1. Algorithm 1 (*A1*) is inspired in [13], considering a centralized decision approach while Algorithm 2 (*A2*) is inspired in [9] following a distributed decision approach. Additionally, Algorithm 3 (*A3*) considers a centralized decision approach implementing the presented prediction-based VMPr Triggering and update-based VMPr Recovering methods. Algorithm 4 (*A4*) is inspired in [14], considering a centralized decision approach. Algorithm 5 (*A5*), also inspired in [14], considers a centralized decision approach implementing the presented prediction-based VMPr Triggering and update-based VMPr Recovering methods. In this context, *A1*, *A2* and *A4* consider original VMPr Triggering and VMPr Recovering methods proposed on each original research work [9], [13].

3.1. Incremental VMP (iVMP) Algorithm

In experimental results previously obtained by the authors in [5], the First-Fit Decreasing (FFD) heuristic outperformed other evaluated heuristics in average; consequently, the mentioned heuristic was the only one considered in this work for the iVMP problem in all evaluated algorithms (*A1* to *A5*), as summarized in Table 1. This way, this paper focuses in its main contribution: the VMPr phase. Further studies on alternative heuristics for the iVMP phase is left as future work.

In the First-Fit (FF) heuristic, requested VMs $V_j(t)$ are allocated on the first PM H_i with available resources (see Section 2.4.2). The considered FFD heuristic operates similarly to FF heuristic, with the main difference that FFD heuristic sorts the list of requested VMs $V_j(t)$ in decreasing order by revenue $R_j(t)$ (first see details in Algorithm 1).

Taking into account the particularities of the presented complex IaaS environment, the FFD heuristic presents some modifications when comparing to the one presented in [5], mainly considering the cloud service request types previously described in Section 2.1. In fact, Algorithm 1 shows that cloud service destruction, scale-down of VM resources and cloud services scale-in are processed first, in order to release resources for immediate re-utilization (steps 1-3 of Algorithm 1). At step 4, requests from $V(t)$ are sorted by a given criterion as revenue ($R_j(t)$) in decreasing order (of

Algorithm 1: First-Fit Decreasing (FFD) for iVMP.

Data: H, $V(t)$, $U(t)$, $x(t)$ (see notation in Section 2.1)
Result: Incremental Placement $x(t+1)$

```
/* removed cloud services                                   */
process cloud services destruction from V(t);
/* vertical elasticity                                      */
process scale-down of VMs resources from V(t);
/* horizontal elasticity                                    */
process cloud services scale-in from V(t);
/* sort VMs by revenue                                      */
sort VMs by revenue (R_j(t)) in decreasing order;
/* vertical elasticity                                      */
process scale-up of VMs resources from V(t);
/* horizontal elasticity                                    */
process cloud services scale-out from V(t);
/* created cloud services                                   */
foreach unprocessed V_j in V(t) do
    while V_j is not allocated do
        foreach H_i in H do
            if H_i has enough resources to host V_j then
                allocate V_j into H_i and break loop;
            end
        end
        if V_j is still not allocated then
            allocate V_j in another federated provider;
        end
    end
end
update x(t+1) with processed requests;
return x(t+1)
```

course, other criterion may be considered, as CPU [5]), where scale-up of VM resources and cloud services scale-out are firstly processed (steps 5-6), in order to consider elastic cloud services more important than non-elastic ones. Next, unprocessed requests from $V_j(t)$ include only cloud service creations that are allocated in decreasing order (steps 7-18). Here, a V_j is allocated in the first H_i with available resources (see (6)) after considering previously sorted $V(t)$. If no H_i has sufficient resources to host V_j, it is allocated in another federated provider. Finally, the placement $x(t+1)$ is updated and returned (steps 19-20).

Algorithm 2: Ant Colony Optimization (ACO) for VMPr.

Data: H, $U(t)$, $x(t)$ (see notation in Section 2.1)
Result: Recalculated Placement $x'(t)$

```
/* initializes empty migration plan                         */
x'(t) = ∅;
Set pheromone value on all VM-PM pairs to τ_max;
foreach q in nCycles do
    foreach a in nAnts do
        /* initializes empty ant score                      */
        Score_ant = 0; x'(t)_tmp = ∅; x'(t)_a = ∅;
        while | x'(t)_tmp | <| VMs | do
            Compute probability to be migrated;
            Choose (v, p) randomly;
            Add (v, p) to x'(t)_tmp;
            Update PMs used capacities;
            /* calculates new score                         */
            Score_tmp = f(x'(t)_tmp);
            /* replace if new score is better               */
            if Score_tmp <Score_ant then
                Score_ant = Score_tmp;
                x'(t)_a = x'(t)_a ∪ {(v, p)};
            end
        end
    end
    Choose the best migration plan as x'(t)_cBest;
    /* replaces x'(t) when a new best local
       score is found                                       */
    if f(x'(t)_cBest) <f(x'(t)); x'(t) = x'(t)_cBest;
    /* updates pheromore based on the new best
       migration plan found                                 */
    foreach (v, p) ∈ V * P do
        τ_v,p = (1 − ρ) * τ_v,p + Δτ^best_v,p;
        if τ_v,p >τ_max ; τ_v,p = τ_max;
        if τ_v,p <τ_min ; τ_v,p = τ_min;
    end
end
return x'(t)
```

3.2. VMP Reconfiguration (VMPr) Algorithms

Previous research work by the authors focused on developing VMPr algorithms considering centralized decisions such as the offline *Memetic Algorithms* (MAs) presented in [7], [8], [15] and the *Ant Colony Optimization* (ACO) presented in [14]. In this work, the considered VMPr algorithms for centralized decision approaches are: (i) *A1* and *A3* (based

TABLE 1. SUMMARY OF EVALUATED ALGORITHMS AS WELL AS THEIR CORRESPONDING VMPR TRIGGERING AND RECOVERING METHODS. N/A INDICATES A NOT APPLICABLE CRITERION.

Algorithm	Approach	iVMP	VMPr	VMPr Triggering	VMPr Recovering
A1	Centralized	FFD	MA	Periodically	Cancellation
A2	Distributed	FFD	MMT	Threshold-based	N/A
A3	Centralized	FFD	MA	*Prediction-based*	*Update-based*
A4	Centralized	FFD	ACO	Periodically	Cancellation
A5	Centralized	FFD	ACO	*Prediction-based*	*Update-based*

on MA) as well as (ii) *A4* and *A5* (based on ACO). Details on each algorithm can be found in [7] and [14].

Additionally, a distributed decision approach is also considered in the experimental evaluation performed in this work (*A2*). For this purpose, the most representative related work was considered [9]: the Minimum Migration Time (MMT) algorithm. The considered MMT is presented in Algorithm 4. For original explanation see [9].

The main difference between the MA (see Algorithm 3) and the ACO (see Algorithm 2) with the MMT algorithm (see Algorithm 4) is the considered decision approach, where the MA and ACO performs a centralized decision that globally reconfigures the placement of VMs while the MMT algorithm performs a distributed decision partially reconfiguring VMs allocated in only one PM at a time.

3.3. Evaluated VMPr Triggering Methods

In this work, a VMPr Triggering method defines when the VMPr phase should be triggered in a two-phase optimization scheme for VMP problems. Considering VMPr Triggering methods, this work evaluated two main approaches: (i) periodical and (ii) threshold-based, mixing them with the VMPr algorithms (see Table 1). This work also presents a prediction-based approach for a novel VMPr Triggering method, statistically analyzing the objective function costs and proactively detecting requirements for triggering the VMPr phase. The following sub-sections describe the VMPr Triggering methods presented in this work as part of a two-phase optimization scheme for VMP problems in a complex IaaS environment.

3.3.1. Periodical Triggering. Several studied works considered to periodically triggering the VMPr phase. This work considers the VMPr Triggering method described in [13], triggering the VMPr phase every 10 discrete time instants.

Periodically triggering the VMPr could present disadvantages when defining a fixed reconfiguration period (e.g. every 10 time instants). For example, a reconfiguration could be required before the established time, where optimization opportunities could be wasted or even economical penalties could impact the cloud datacenter operation. In certain cases the reconfiguration may not be necessary and triggering the VMPr could represent profitless use of resources.

Algorithm 3: Memetic Algorithm (MA) for VMPr.

Data: H, $U(t)$, $x(t)$ (see notation in Section 2.1)
Result: Recalculated Placement $x'(t)$
initialize set of candidate solutions Pop_0;
Pop'_0 = repair infeasible solutions of Pop_0;
Pop''_0 = apply local search to solutions of Pop'_0;
$x'(t)$ = select best solution from $Pop''_0 \cup x(t)$ considering (14);
$u = 0; Pop_u = Pop''_0$;
while *stopping criterion is not satisfied* **do**
  Pop_u = selection of solutions from $Pop_u \cup x'(t)$;
  Pop'_u = crossover and mutation on solutions of Pop_u;
  Pop''_u = repair infeasible solutions of Pop'_u;
  Pop'''_u = apply local search to solutions of Pop''_u;
  $x'(t)$ = select best solution from Pop'''_u considering (14);
  increment number of generations u;
end
return $x'(t)$

Algorithm 4: Minimum Migration Time (MMT) for VMPr running at PM H_i.

Data: H, $U(t)$, $x(t)$, H_i (see notation in Section 2.1)
Result: Recalculated Placement $x'(t)$
/* H_i has exceeded upper threshold */
if H_i *is overloaded* **then**
  sort VMs V_j allocated into H_i in increasing order by RAM;
  while H_i *is overloaded* **do**
    schedule migration of V_j from H_i to $H_{i'}$ using FFD;
  end
end
/* H_i does not reach the lower threshold */
if H_i *is underloaded* **then**
  schedule migration of all V_j from H_i to $H_{i'} \neq H_i$ if possible;
end
update $x'(t)$ considering scheduled migrations;
return $x'(t)$

3.3.2. Threshold-based Triggering. Another very studied VMPr Triggering method considers a threshold-based approach, where thresholds are defined in terms of utilization of PM resources (e.g. CPU). Thresholds indicate when a PM H_i is considered to be underloaded or overloaded, and consequently, a VMPr should be triggered. This work considers a threshold-based VMPr Triggering method based on [9], fixing utilization thresholds for overloaded and underloaded PM detection, to 10% and 90% respectively.

The above described threshold-based VMPr Triggering method makes isolated reconfiguration decisions at each PM without a complete knowledge of global optimization objectives, giving place to a distributed decision approach, as the algorithm *A2* implemented in this work.

3.3.3. Proposed Prediction-based Triggering. Considering the main identified issues related to the studied VMPr Triggering methods, this work presents a novel prediction-based VMPr Triggering method, statistically analyzing the global objective function $F(x,t)$ that is optimized (see (14)) and proactively detecting situations where a VMPr triggering is potentially required for a placement reconfiguration.

The prediction-based VMPr Triggering method considers Double Exponential Smoothing (DES) [16] as a statistical technique for predicting values of the objective function $F(x,t)$, as formulated next in (19) to (21):

$$S_t = \alpha \times Z_t + (1-\tau)(S_{t-1} + b_{t-1}) \quad (19)$$

$$b_t = \tau(S_t - S_{t-1}) + (1-\tau)(b_{t-1}) \quad (20)$$

$$\overline{Z}_{t+1} = S_t + b_t \quad (21)$$

Algorithm 5: Update-based VMPr Recovering.

Data: $x(t)$, $x'(t-\beta)$ (see notation in Section 2.1)
Result: Recovered Placement $x'(t)$
1 remove VMs V_j from $x'(t-\beta)$ that are no longer running in $x(t)$
2 adjust resources from $x'(t-\beta)$ that changed in $x(t)$
3 add VMs V_j from $x(t)$ that were not considered in $x'(t-\beta)$
4 **if** $x'(t-\beta)$ *is better than* $x(t)$ **then** ;
5 **return** $x'(t-\beta)$;
6 **else return** $x(t)$;

where: α is the smoothing factor, $0 \leq \alpha \leq 1$; τ is the trend factor, $0 \leq \tau \leq 1$; Z_t is the known value of $F(x,t)$ at discrete time t; S_t represents the expected value of $F(x,t)$ at discrete time t; b_t is the trend of $F(x,t)$ at discrete time t; $\overline{Z}_{t+1}$ is the value of $F(x,t+1)$ at discrete time t.

At each discrete time t, the VMPr Triggering method predicts the next N values of $F(x,t)$ and effectively triggers the VMPr phase in case $F(x,t)$ is predicted to consistently increase, considering that $F(x,t)$ is being minimized.

3.4. Evaluated VMPr Recovering Methods

It is important to consider that the placement reconfiguration calculated in the VMPr phase is potentially obsolete, considering the offline nature of the VMPr problem formulation. In this work, a VMPr Recovering method defines what should be done with cloud service requests arriving during the VMPr recalculation time β. The iVMP may receive cloud service requests during the β discrete time that the VMPr performed the calculation of an improved placement (see Figure 1). Consequently, the calculated new placement must be recovered according to the considered VMPr Recovering method before the reconfiguration is performed. The mentioned issue is mainly associated to centralized decision approaches, such as the ones presented in [7], [13].

3.4.1. Canceling Reconfiguration. Calcavecchia et al. studied in [13] a very basic VMPr Recovering method, canceling the VMPr whenever a new request is received. In this case, the VMPr is only performed in periods with no requests, that could be considered unpractical for IaaS providers, taking into account the highly dynamic environment of cloud computing markets and particularly the complex IaaS environment presented in this work.

3.4.2. Proposed Update-based Recovering. This work presents a VMPr Recovering method based on updating the placement reconfiguration calculated in the VMPr phase, according to the changes that happened during the placement recalculation time, applying operations to update the potentially obsolete placement (see Algorithm 5).

The update-based VMPr Recovering method receives the placement reconfiguration calculated in the VMPr phase (corresponding to the discrete time $t-\beta$) and the current placement $x(t)$ as input data (see Algorithm 5).

Considering that any VM V_j could be destroyed, or a cloud service could be scaled-in (horizontal elasticity) during the β discrete times where the calculation of the placement reconfiguration was performed, these destroyed VMs are removed from $x'(t-\beta)$ (step 1). Next, any resource from a VM V_j could be adjusted due to a scale-up or scale-down (vertical elasticity). Consequently, these resource adjustments are performed in $x'(t-\beta)$ (see step 2). Additionally, new VMs V_j could be created, or a cloud service could be scaled-out (horizontal elasticity), during the calculation of $x'(t-\beta)$. In the example of Figure 1 cloud service S_2 is created ($+V_2$) and additionally a scale-out of the mentioned cloud service was performed ($+V_3$) during the recalculation time β. These VMs are added to $x'(t-\beta)$ using an FFD heuristic (step 3), the same heuristic used in the iVMP phase. Finally, if the partially recalculated placement $x'(t-\beta)$ is better than the current placement $x(t)$, $x'(t-\beta)$ it is accepted (step 5) and the corresponding management actions are performed (i.e. mainly migration of VMs between PMs). In case $x'(t-\beta)$ is not better than the current placement $x(t)$, no change is performed and the VMPr phase finishes without any further consequence.

4. Experimental Evaluation

The following sub-sections summarize the experimental environment as well as the main findings identified in the experiments performed as part of the simulations to validate the two-phase optimization scheme for VMP problems. The quality of solutions obtained by the evaluated algorithms in a scenario-based uncertainty model with 400 different scenarios was compared considering the following evaluation criteria among solutions: (i) average, (ii) maximum and (iii) minimum objective function costs, defined in (16) to (18).

4.1. Experimental Environment

The five evaluated algorithms (see Table 1) previously presented in Section 3 were implemented using Java programming language. The source code is available online[1], as well as all the considered input data and experimental results. Experiments were performed on a GNU Linux Operating System with an Intel(R) Xeon(R) E5530 at 2.40 GHz CPU and 16 GB of RAM memory. The following parameters of the presented uncertain VMP formulation were considered for the experimental evaluation presented in this work (see details in Section 2):

- Number of considered resources: $r = 3$;
- Recalculation time for $A1$, $A3$, $A4$ and $A5$: $\beta = 2$;
- Recalculation time for $A2$: $\beta = 1$;
- Protection factor for each resource k: $\lambda_k = 0.5$;
- Penalty factor for each resource k: $\phi_k = 1$.

As input data, available PMs (see (1)) include 4 different types of PMs. Considering the available PM types, 5 different IaaS datacenters were considered (DC_1 to DC_5).

Additionally, 80 different workload traces[1] of requested cloud services ($V(t)$) and their specifications (see (2)) were considered as input data as well as their utilization of

1. http://github.com/DynamicVMP/DynamicVMPFramework

resources $U(t)$ at each discrete time t (see (3)). Requested VMs were considered according to instance types offered by Amazon Elastic Compute Cloud (EC2).

It is important to remember that in this work, the following parameters are considered to be uncertain: (i) virtual resources capacities (vertical elasticity), (ii) number of VMs that compose cloud services (horizontal elasticity), (iii) utilization of CPU and RAM memory virtual resources and (iv) utilization of networking virtual resources (both relevant for overbooking). Consequently, two different Probability Distribution Functions (PDFs) were considered to represent each parameter behavior (i.e. Uniform and Poisson). Workload traces of cloud service requests were generated using a Cloud Workload Trace Generator (CWTG) for provider-oriented VMP problems [17], and are available online[2].

Considering the scenario-based uncertainty modeling approach presented in this work, each evaluated scenario $s \in S$ is composed by an IaaS and a workload trace of requested cloud services, totalizing 400 different evaluated scenarios (i.e. 80 workload traces $\times$ 5 IaaS datacenters).

For simplicity, only one cloud computing datacenter is considered in this work. Experiments are summarized in what follows: ten runs of algorithms *A1*, *A3*, *A4* and *A5* were performed for the 400 considered scenarios, taking into account the randomness of the MA and ACO considered for solving the VMPr phase. Average obtained results are presented in Table 2. The same table also shows results of one run of the determinist *A2* algorithm, performed with the same 400 scenarios.

4.2. Experimental Results

Table 2 presents values of the considered evaluation criteria, i.e. F_1, F_2 and F_3 costs (see (16) to (18)), summarizing results obtained in performed simulations. The evaluation criteria are presented separately for each of the five considered IaaS cloud datacenter (DC1 to DC 5). Also, the considered IaaS cloud datacenters represent datacenters of different sizes and consequently, and the considered workload traces represent different load of requested CPU resources (e.g. Low ($\leq$ 30%), Medium ($\leq$ 60%), High ($\leq$ 90%), Full ($\leq$ 98%) and Saturate ($\leq$ 120%)) workloads.

2. http://github.com/DynamicVMP/workload-trace-generator

TABLE 2. SUMMARY OF EVALUATION CRITERIA IN EXPERIMENTAL RESULTS FOR EVALUATED ALGORITHMS. VALUES IN BOLD INDICATE BEST RESULTS.

Criterion	\	DC_1	DC_2	DC_3	DC_4	DC_5	*Ranking*
F_1	*A1*	0.839	0.944	1.003	1.007	1.039	2^{nd}
	A2	0.841	0.946	1.002	1.003	1.037	3^{rd}
	A3	0.779	**0.866**	**0.926**	**0.918**	**0.963**	1^{st}
	A4	0.773	0.956	1.029	1.034	1.111	5^{th}
	A5	**0.769**	0.946	1.024	1.031	1.096	4^{th}
F_2	*A1*	1.029	1.138	1.208	1.224	1.252	2^{nd}
	A2	1.036	1.143	1.214	1.226	1.256	3^{rd}
	A3	1.006	**1.102**	**1.163**	**1.183**	**1.205**	1^{st}
	A4	**0.936**	1.280	1.552	1.568	1.771	5^{th}
	A5	0.938	1.307	1.496	1.508	1.658	4^{th}
F_3	*A1*	0.691	0.798	0.830	0.806	0.839	3^{rd}
	A2	0.690	0.763	0.808	0.785	0.820	2^{nd}
	A3	**0.546**	**0.613**	**0.653**	**0.586**	**0.648**	1^{st}
	A4	0.633	0.809	0.848	0.861	0.894	5^{th}
	A5	0.645	0.809	0.842	0.855	0.853	4^{th}

Based on the information presented in Table 2, the *Main Findings* (MFs) of the experimental evaluation performed in this work are summarized as follows:

MF1: *Algorithm $A3$ that considered the presented VMPr Triggering and VMPr Recovering methods outperformed all other evaluated algorithms in most experiments, taking into account the considered evaluation criteria.*

In summary, $A3$ obtained better results (minimum cost) for the three considered evaluation criteria (see Table 2).

When considering average objective function costs (F_1) as evaluation criterion, $A3$ outperformed the other algorithms except in the DC1, in which A5 got a slightly better result. Additionally, when considering maximum objective function costs (F_2) as evaluation criterion, $A3$ obtained the best results except in DC1, being outperformed by A4 in 7%. Finally, when considering minimum objective function costs (F_3), $A3$ got the best results on each datacenter.

Figure 2 illustrates the temporal average cost of the single combined objective function for all scenarios $s \in S$, denoted as $\overline{f_s(x,t)}$ in (15).

MF2: *The presented $A3$ algorithm outperformed other evaluated algorithms in most of the considered scenarios, when considering average values of the single combined objective function on each scenario $s \in S$.*

As presented in Figure 2, $A3$ outperformed the other 4 algorithms in most of the considered scenarios, while performing near to the other evaluated alternatives in the few scenarios where it does not perform as the best alternative. $A3$ was the best algorithm in 80% of the 400.

Summarizing, according to the performed experimental evaluation, the algorithm that considered the presented prediction-based VMPr Triggering, the update-based VMPr Recovering methods, and the MA based on [7], [8], [15] for the VMP reconfiguration phase ($A3$), is the clear alternative for solving the uncertain VMP problem in a two-phase optimization scheme, considering the simulation results presented in this section.

5. Conclusions and Future Work

A renewed formulation of an uncertain VMP problem was presented, for the optimization of the following four objective functions: (i) power consumption, (ii) economical revenue, (iii) resource utilization as well as (iv) placement reconfiguration time (see Section 2.5) in a context of Multi-Objective problem solved as Mono-Objective (MAM) [7]. Clearly, the formulation may consider any other objectives functions with almost no change.

Additionally, a scenario-based uncertainty approach for modeling relevant uncertain parameters considering the complex IaaS environment in the two-phase optimization scheme for VMP problems was presented. The considered

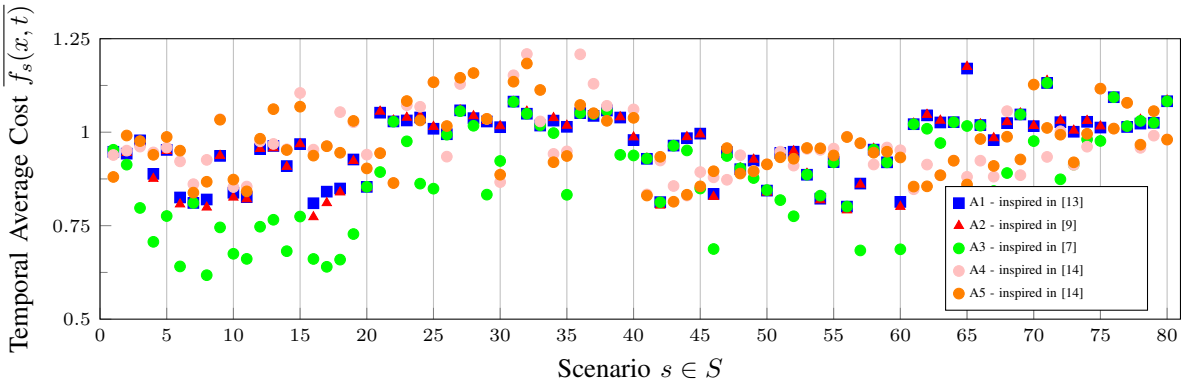

Figure 2. Temporal average cost: Average values of $\overline{f_s(x,t)}$ in DC_1 to DC_5 per each scenario $s \in S$.

uncertain parameters were: (i) virtual resources capacities (vertical elasticity), (ii) number of VMs that compose cloud services (horizontal elasticity), (iii) utilization of CPU and RAM memory virtual resources and (iv) utilization of networking virtual resources (both relevant for overbooking).

Experimental results in simulations suggested that the best algorithm for solving the presented uncertain VMP problem in a two-phase optimization scheme is the one considering the prediction-based VMPr Triggering, the update-based VMPr Recovering methods and a MA ($A3$ algorithm).

Several future works were also identified. First, a formulation of a VMP problem considering a dynamic set of PMs $H(t)$, to consider PM crashes, maintenance or even deployment of new generation hardware is proposed as a future work.

Experimenting with geo-distributed datacenters is also left as a future work, taking into account that simulations presented considered only one cloud computing datacenter.

Finally, other VMPr algorithms can be proposed and mixed with the presented VMPr Triggering and VMPr Recovering methods resulting in a potentially better variety of new algorithms in a two-phase optimization scheme.

References

[1] J. Ortigoza, F. López-Pires, and B. Barán, "A taxonomy on dynamic environments for provider-oriented virtual machine placement," in *2016 IEEE International Conference on Cloud Engineering (IC2E)*, April 2016, pp. 214–215.

[2] P. Mell and T. Grance, "The nist definition of cloud computing," *National Institute of Standards and Technology*, 2009.

[3] F. López-Pires and B. Barán, "A virtual machine placement taxonomy," in *Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on*. IEEE Computer Society, May 2015, pp. 159–168.

[4] B. Speitkamp and M. Bichler, "A mathematical programming approach for server consolidation problems in virtualized data centers," *Services Computing, IEEE Transactions on*, pp. 266–278, 2010.

[5] F. López-Pires, B. Barán, A. Amarilla, L. Benítez, R. Ferreira, and S. Zalimben, "An experimental comparison of algorithms for virtual machine placement considering many objectives," in *9th Latin America Networking Conference (LANC)*, 2016, pp. 75–79.

[6] P. Sv, W. Li, E. Wadbro, J. Tordsson, E. Elmroth *et al.*, "Continuous datacenter consolidation," in *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 2015, pp. 387–396.

[7] D. Ihara, F. López-Pires, and B. Barán, "Many-objective virtual machine placement for dynamic environments," in *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*. IEEE, 2015, pp. 75–79.

[8] F. López-Pires and B. Barán, "A many-objective optimization framework for virtualized datacenters," in *Proceedings of the 2015 5th International Conference on Cloud Computing and Service Science*, 2015, pp. 439–450.

[9] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755–768, 2012.

[10] M. Gahlawat and P. Sharma, "Survey of virtual machine placement in federated clouds," in *Advance Computing Conference (IACC), 2014 IEEE International*, Feb 2014, pp. 735–738.

[11] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[12] M. A. Aloulou and F. Della Croce, "Complexity of single machine scheduling problems under scenario-based uncertainty," *Operations Research Letters*, vol. 36, no. 3, pp. 338–342, 2008.

[13] N. M. Calcavecchia, O. Biran, E. Hadad, and Y. Moatti, "Vm placement strategies for cloud scenarios," in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*. IEEE, 2012, pp. 852–859.

[14] E. Feller, C. Morin, and A. Esnault, "A case for fully decentralized dynamic VM consolidation in clouds," in *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*. IEEE, 2012, pp. 26–33.

[15] F. López-Pires and B. Barán, "Multi-objective virtual machine placement with service level agreement: A memetic algorithm approach," in *Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*. IEEE Computer Society, 2013, pp. 203–210.

[16] J. Huang, C. Li, and J. Yu, "Resource prediction based on double exponential smoothing in cloud computing," in *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, April 2012, pp. 2056–2060.

[17] J. Ortigoza, F. López-Pires, and B. Barán, "Workload generation for virtual machine placement in cloud computing environments," in *2016 XLII Latin American Computing Conference (CLEI)*, Oct 2016, pp. 1–9.