Evaluation of Two-Phase Virtual Machine Placement Algorithms for Green Cloud Datacenters

Fabio López-Pires[1*], Benjamín Barán[2,3], Carolina Pereira[2], Marcelo Velázquez[2] and Osvaldo González[2]
[1]*Itaipu Technological Park, Hernandarias, Paraguay*
[2]*National University of the East, Ciudad del Este, Paraguay*
[3]*National University of Asuncion, San Lorenzo, Paraguay*

***Abstract*—Cloud Computing Datacenters represent a power-intensive industry with well-known economical and ecological challenges. This work focusses on Virtual Machine Placement (VMP) problems as a valid alternative to address mentioned challenges. An experimental evaluation of 36 VMP optimization algorithms for power consumption minimization is presented. Algorithms were evaluated under uncertainty of 4 different dynamic parameters, considering 400 experimental scenarios and taking into account an average objective function cost as evaluation criterion. Experimental results indicate that two-phase algorithms considering prediction-based VMPr Triggering and update-based VMPr Recovering methods are best suited for power consumption minimization.**

***Index Terms*—Virtual Machine Placement, Green Cloud Computing, Power Consumption, Infrastructure-as-a-Service**

I. Introduction

Selecting which requested *virtual machines* (VMs) should be hosted at each available *physical machine* (PM) of a cloud computing infrastructure is commonly known as *Virtual Machine Placement* (VMP). This work focusses on the evaluation of different VMP algorithms for Green Cloud Datacenters in a two-phase optimization scheme, based on a previously proposed formulation under uncertainty [1]. In this context, an experimental evaluation of 36 VMP algorithms was performed, considering 80 workloads with 5 datacenters providing *Infrastructure-as-a-Service* (IaaS), totalizing 400 different experimental scenarios [1].

It is worth mentioning that 4 of the 36 VMP algorithms were evaluated in [1]. This work extends the mentioned evaluation in order to include a complete combination of identified resolution alternatives and methods associated to VMP two-phase optimization schemes (see Table I).

The remainder of this work is structured as follows: a simplified VMP formulation is introduced in Section II, while evaluated VMP algorithms are briefly presented in Section III. Section IV summarizes the experimental evaluation as well as the main results. Finally, conclusions and future directions are left to Section V.

**Email address:* fabio.lopez@pti.org.py (Fabio López-Pires)

II. Uncertain VMP Formulation

The VMP problem could be formulated as both online and offline optimization problems [2]. In order to improve the quality of solutions given by online algorithms, the VMP problem could also be formulated as a two-phase optimization scheme, which combines advantages of online and offline formulations for IaaS environments, as previously demonstrated in [1]. As a result, VMP problems are decomposed into two different sub-problems: (i) *incremental VMP* (iVMP) and (ii) *VMP reconfiguration* (VMPr), as shown in Figure 1.

A. Incremental VMP (iVMP)

In this work decisions are performed at each discrete time t, and the iVMP problem could be enunciated as:

Given a complex IaaS environment composed by a set of PMs (H), a set of active VMs which has been 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 $\Delta V(t+1)$ of $V(t)$ into H for the discrete time $t+1$ ($x(t+1)$) without migrations, satisfying constraints and optimizing a given objective function. Clearly, in this iVMP problem, $V(t+1) = V(t) + \Delta V(t+1)$.

1) Input Data for iVMP: The considered formulation [1] receives as input the following information: (i) a set of n available PMs with their specifications; (ii) a set of $m(t)$ requested VMs (already allocated VMs in $V(t)$ plus new requests $\Delta V(t+1)$) and their specifications; (iii) information about the utilization of resources of each active VM at each discrete time t; as well as (iv) current placement at each discrete time t (i.e. $x(t)$).

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

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

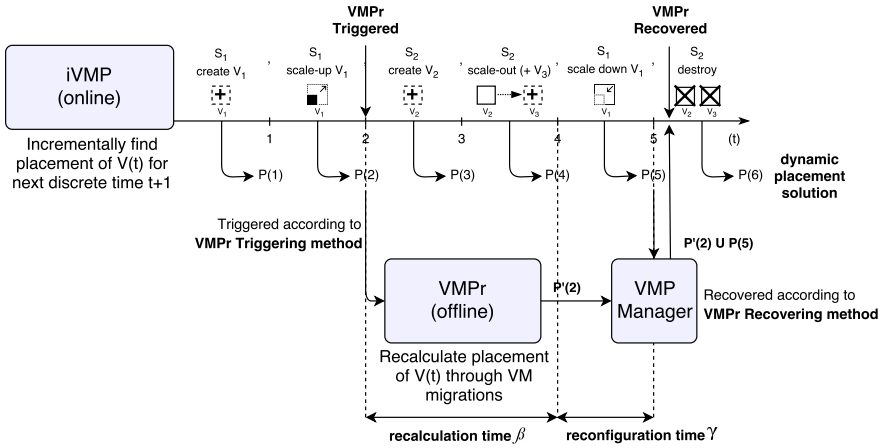

Fig. 1. Two-phase optimization scheme for VMP problems considered in this work, presenting a basic example with a placement recalculation time of $\beta = 2$ (from $t = 2$ to $t = 4$) and a placement reconfiguration time of $\gamma = 1$ (from $t = 4$ to $t = 5$). Authorized by the authors [1].

Formally, the placement for the next time instant $x(t+1)$ is a function of the current placement $x(t)$ and the variation on active VMs at discrete time $t + 1$, i.e.:

$$x(t + 1) = f\left[x(t), \Delta V(t + 1)\right] \tag{2}$$

B. 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 [1]. The formulation of the considered VMPr problem could be enunciated as:

Given a placement of VMs into PMs $(x(t))$, it is sought a placement reconfiguration through migration of VMs among PMs for a discrete time t (i.e. $x'(t)$), satisfying constraints and optimizing an objective function.

A *VMPr Triggering* method defines under which circumstances the VMPr phase should be triggered in the considered two-phase optimization scheme for VMP problems.

1) Input Data for VMPr: It receives the following information as input data: (i) a set of n available PMs with their specifications; (ii) information about the utilization of resources of each active VM at discrete time t; as well as (iii) current placement at discrete time t (i.e. $x(t)$).

2) Output Data for VMPr: The result of the VMPr problem is a placement reconfiguration through migration of VMs among different PMs for the discrete time t, represented by a new placement reconfiguration of $x(t)$, i.e. $x'(t)$.

Given the offline nature of the VMPr phase, $x'(t)$ could be obsolete at the end of a placement recalculation. Here, a *VMPr Recovering* method defines what should be done with cloud service requests arriving during the VMP recalculation time β (see Figure 1).

3) Constraints: Summarizing, during the iVMP process a VM V_j must be allocated to run on a single PM H_i or alternatively located in another federated IaaS provider. Additionally, a PM H_i must have sufficient available resources (CPU, RAM and network resources in this work) to meet the dynamic requirements of all VMs V_j that are allocated to run on H_i.

It is worth remembering that resources of VMs are dynamically used, giving space to re-utilization of idle resources that were already reserved in case of overbooking. Re-utilization of idle resources could represent a 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. Interested readers could refer to a previous work for details [1].

C. Power Consumption Minimization

According to a previous research work [3], the power consumption can be represented by the sum of the power consumption of each PM H_i at each discrete time t:

$$f(x,t) = \sum_{i=1}^{n}((pmax_i - pmin_i) \times Ur_{1,i}(t) + pmin_i) \times Y_i(t) \tag{3}$$

where:

x: evaluated solution of the problem;
$f(x,t)$: total power consumption of PMs at instant t;
$pmax_i$: maximum power consumption of a PM H_i;
$pmin_i$: minimum power consumption of a PM H_i; As suggested in [3], $pmin_i \approx pmax_i * 0.6$;
$Ur_{1,i}(t)$: utilization ratio of resource 1 (in this case CPU) by H_i at instant t;
$Y_i(t) \in \{0,1\}$: indicates if H_i is turned on $(Y_i(t) = 1)$ or not $(Y_i(t) = 0)$ at instant t.

D. Normalization

This work normalizes the objective function cost $f(x,t)$ by calculating $\hat{f}(x,t) \in \mathbb{R}$, where $0 \leq \hat{f}(x,t) \leq 1$ as:

$$\hat{f}(x,t) = \frac{f(x,t) - f(x,t)_{min}}{f(x,t)_{max} - f(x,t)_{min}} \tag{4}$$

where:

$\hat{f}(x,t)$: objective function normalized cost $f(x,t)$ at instant t;
$f(x,t)$: cost of original objective function $f(x,t)$;
$f(x,t)_{min}$: minimum possible cost for $f(x,t)$;
$f(x,t)_{max}$: maximum possible cost for $f(x,t)$.

E. Scenario-based Uncertainty Modeling

In this work, uncertainty is modeled through a finite set of well-defined scenarios S [4], 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 of virtual resources and (iv) utilization of networking virtual resources.

For each scenario $s \in S$, a temporal average value of the objective function $\hat{f}(x,t)$ is calculated as:

$$\overline{f_s(x,t)} = \frac{\sum_{t=1}^{tmax} \hat{f}(x,t)}{t_{max}} \quad (5)$$

where:

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

This work considers the minimization of the average objective function costs for each scenario [4] to select among solutions:

$$F_1 = \frac{\sum_{s=1}^{|S|} \overline{f_s(x,t)}}{|S|} \quad (6)$$

where:

F_1: average $\overline{f_s(x,t)}$ for all scenarios $s \in S$;
$|S|$: cardinality of set S of scenarios.

III. Evaluated Algorithms

An experimental evaluation of 36 different VMP algorithms ($A1$ to $A36$) is presented in a two-phase optimization scheme. Each considered VMP algorithm is composed by: (i) a iVMP resolution alternative, (ii) a VMPr resolution alternative, (iii) a *VMPr Triggering* method and (iv) a *VMPr Recovering* method (see Table I).

The following sub-sections briefly present the considered iVMP and VMPr resolution alternatives as well as the *VMPr Triggering* and *VMPr Recovering* methods.

A. Incremental VMP (iVMP) Algorithms

In this work, the following iVMP heuristic resolution alternatives were evaluated (see Column 3 of Table I): (i) *First-Fit* (FF), (ii) *Best-Fit* (BF), (iii) *Worst-Fit*, (iv) *First-Fit-Decreasing*, (v) *Best-Fit-Decreasing* (BFD) and (vi) a Filter Scheduler implemented in *OpenStack*[1] (OS).

Experimental results obtained by some of the authors already demonstrated that the FFD heuristic outperformed other evaluated algorithms in average [5]. In the FF heuristic, requested VMs are allocated on the first PM with available resource [6]. In FFD and BFD, the list of requested VMs are decreasingly ordered [1]. Considering previous research work [7], a technique inspired in the operation of the real-world IaaS middleware *OpenStack* (OS) is also considered. The OS resolution technique applies filtering and weighting for selecting a PM H_i to host a VM V_j.

[1]//docs.openstack.org/mitaka/config-reference/compute/scheduler

TABLE I
Summary of evaluated VMP algorithms. N/A indicates a not applicable criterion.

Alg.	Decision Approach	iVMP	VMPr	VMPr Triggering	VMPr Recovering
A1	N/A	FF	N/A	N/A	N/A
A2	N/A	BF	N/A	N/A	N/A
A3	N/A	WF	N/A	N/A	N/A
A4	N/A	FFD	N/A	N/A	N/A
A5	N/A	BFD	N/A	N/A	N/A
A6	N/A	OS	N/A	N/A	N/A
A7	D	FF	MMT	T-B	-
A8	D	BF	MMT	T-B	-
A9	D	WF	MMT	T-B	-
A10	D	FFD	MMT	T-B	-
A11	D	BFD	MMT	T-B	-
A12	D	OS	MMT	T-B	-
A13	C	FF	MA	P	$\not C$
A14	C	BF	MA	P	$\not C$
A15	C	WF	MA	P	$\not C$
A16	C	FFD	MA	P	$\not C$
A17	C	BFD	MA	P	$\not C$
A18	C	OS	MA	P	$\not C$
A19	C	FF	MA	P	U-B
A20	C	BF	MA	P	U-B
A21	C	WF	MA	P	U-B
A22	C	FFD	MA	P	U-B
A23	C	BFD	MA	P	U-B
A24	C	OS	MA	P	U-B
A25	C	FF	MA	P-B	$\not C$
A26	C	BF	MA	P-B	$\not C$
A27	C	WF	MA	P-B	$\not C$
A28	C	FFD	MA	P-B	$\not C$
A29	C	BFD	MA	P-B	$\not C$
A30	C	OS	MA	P-B	$\not C$
A31	C	FF	MA	P-B	U-B
A32	C	BF	MA	P-B	U-B
A33	C	WF	MA	P-B	U-B
A34	C	FFD	MA	P-B	U-B
A35	C	BFD	MA	P-B	U-B
A36	C	OS	MA	P-B	U-B

B. VMP Reconfiguration (VMPr) Algorithms

As presented in Column 2 of Table I, the VMPr phase may be considered with centralized (C) or distributed (D) approaches when considering a two-phase optimization scheme for VMP problems. In a centralized decision approach (C), the optimization is globally performed, evaluating the placement of all allocated VMs, while a dis-

tributed approach (D) partially reconfigures VMs allocated in one isolated PM [2].

In the presented experimental evaluation, algorithms $A1$ to $A6$ do not consider any VMPr resolution alternative (that is why Table Iindicates N/A, i.e. not applicable), in order to evaluate two-phase optimization schemes against iVMP alternatives. Additionally, algorithms $A7$ to $A12$ consider a distributed optimization approach, inspired in the *Minimum Migration Time* (MMT) resolution alternative [3]. On the other hand, a centralized optimization approach (C) is considered for algorithms $A13$ to $A36$ based on a *Memetic Algorithm* (MA) (see Column 4 of Table I).

C. Evaluated VMPr Triggering Methods

This work evaluates three VMPr Triggering methods (see Column 5 of Table I): (i) *threshold-based*, (ii) *periodical* and (iii) *prediction-based*. These considered VMPr Triggering methods are briefly introduced next.

1) Threshold-based Triggering (T-B): In this VMPr triggering method, VMPr phases are triggered when defined thresholds are reached. Thresholds are typically defined in terms of utilization of PM resources (e.g. CPU). For this experimental evaluation, a PM V_i is considered to be underloaded (when 10% of CPU utilization) or overloaded (90%). Consequently, a VMPr should be triggered [3].

2) Periodical Triggering (P): A periodical VMPr triggering method consists in triggering VMPr phases in fixed periods of time (e.g. every 10 time instants) [8]. Periodically triggering the VMPr phase could present disadvantages. For example, a reconfiguration could be required before the established time or these triggered reconfiguration phases could be unnecessary. Consequently, optimization opportunities could be wasted or even economical penalties could impact cloud datacenter operation.

3) Prediction-based Triggering (P-B): The considered prediction-based VMPr triggering method uses a *Double Exponential Smoothing* (DES) as a statistical technique for predicting values of the objective functions $\hat{f}(x,t)$, mathematically formulated in Equations (7) to (9):

$$S_t = \alpha \times Z_t + (1-\tau)(S_{t-1} + b_{t-1}) \quad (7)$$

$$b_t = \tau(S_t - S_{t-1}) + (1-\tau)(b_{t-1}) \quad (8)$$

$$\overline{Z}_{t+1} = S_t + b_t \quad (9)$$

where:

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

At each discrete time t, the prediction-based VMPr triggering method predicts the next M values of $\hat{f}(x,t)$ and triggers the VMPr phase if the objective function consistently deteriorates.

D. Evaluated VMPr Recovering Methods

The following VMPr recovering methods are considered: (see column 6 of Table I): (i) *cancellation* and (ii) *update-based*.

1) Canceling Reconfiguration (⊘): Calcavecchia et al. proposed to cancel the VMPr phase whenever a new request is received [8]. This trivial case considers that VMPr phases should only be performed in periods of time with no request.

2) Update-based Recovering (U-B): This VMPr Recovering method is 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 when it is useful [1].

IV. Experimental Evaluation

This section summarizes the experimental environment and the main findings identified in the performed experiments as part of the simulations to evaluate performance of the 36 considered algorithms ($A1$ to $A36$, presented in Table I). The quality of obtained solutions is evaluated in a scenario-based uncertainty model with 400 different experimental scenarios taking into account the average performance metric F_1 defined in Equation (6).

A. Experimental Environment

The 36 evaluated VMP algorithms (see Table I) were implemented using Java programming language. It is worth noting that the presented experimental evaluation is based on simulations, considering the simulation framework presented in [1]. The source code is available online[2].

Taking into account that the VMP formulation considered in this work focusses exclusively in power consumption, adaptions on the existing framework were included. Additionally, code improvements for easily combine all possible iVMP and VMPr resolution alternatives as well as VMPr Triggering and VMPr Recovering methods.

Simulations were performed on a GNU Linux Operating System with an Intel Xeon E3-12000 v6 processor and 32 GB of RAM memory.

The following parameters of the proposed uncertain VMP formulation were considered for the experimental evaluation presented in this work:

- Number of considered resources: $r = 3$;
- Recalculation time for $A13$ to $A36$: $\beta = 2$;
- Recalculation time for $A7$ to $A12$: $\beta = 1$;

Considering previous research by some of the authors, best suitable values for protection factors according to the CPU load scenarios are: (i) $\lambda_{k,1} = 0$ for low CPU loads and (ii) $\lambda_{k,2} = 0.75$ for high CPU loads [5]. Therefore, this work considers those values respectively.

[2]https://github.com/SDDCVMP/framework/tree/power-consumption

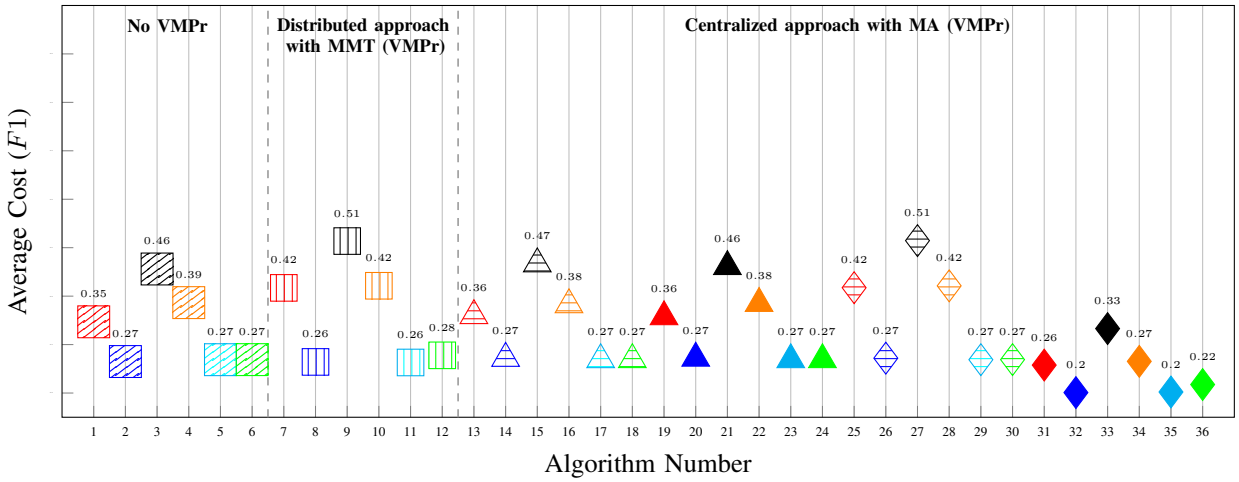

Fig. 2. Average Cost (F1): Average values of $F1$ for each algorithm of Table I. For details on references of this figure, see Table II.

Additionally, 80 different workload traces of requested cloud services ($V(t)$) and their specifications were considered as input data as well as their utilization of resources $U(t)$ at each discrete time t. Workload traces of cloud service requests were generated using a Cloud Workload Trace Generator (CWTG)[3] based on [9] for provider-oriented VMP problems in cloud computing environments. The considered IaaS cloud infrastructure represents datacenters of different sizes. Consequently, the considered workload traces represent different loads of requested CPU resources (e.g. Low ($\leq$ 30%), Medium ($\leq$ 60%), High ($\leq$ 90%), Full ($\leq$ 98%) and Saturated ($\leq$ 120%)) workloads. Each evaluated scenario $s \in S$ is composed by an IaaS datacenter and a workload trace of requested cloud services, totalizing 400 different evaluated scenarios (i.e. 80 workload traces x 5 IaaS datacenters [1]).

Considering the probabilistic nature of the MA resolution alternative, five runs of algorithms $A13$ to $A36$ were performed. Experimental results obtained in this work are summarized in the following sub-sections.

B. Experimental Results

The main goal of the presented experimental evaluation is to explore different alternatives to answer the following research questions ***(RQ)***:

- ***RQ1:*** *Which algorithm presents the best average solution for power consumption minimization with the considered scenarios?.*
- ***RQ2:*** *Under what circumstances should be triggered a VMPr phase? (VMPr Triggering).*
- ***RQ3:*** *What should be done with all the cloud services requested during VMPr recalculation times? (VMPr Recovering).*

[3]https://github.com/DynamicVMP/workload-trace-generator

TABLE II
REFERENCES OF FIGURE 2.

iVMP alternative	**Color**
First-Fit (FF)	red
Best-Fit (BF)	blue
Worst-Fit (WF)	black
First-Fit-Decreasing (FFD)	orange
Best-Fit-Decreasing (BFD)	cyan
OpenStack (OS)	green
VMPr Triggering method	**Pattern**
Not Applicable (N/A)	
Threshold-based	
Periodical	
Prediction-based	
VMPr Recovering method	**Shape**
Not Applicable (N/A)	
Cancellation	
Update-based	

Based on the summarized experimental results presented in Figure 2, the *Main Findings* (MFs) of our evaluation are briefly presented next:

MF1: *In average, $A32$ presents the best quality of solutions, taking into account the F_1 performance metric as evaluation criterion.*

It is worth remembering that $A32$ is based on a centralized decision approach (C) with: (i) BF as iVMP resolution alternative, (ii) MA as VMPr resolution alternative, (iii) a prediction-based (P-B) VMPr Triggering method and (iv) an update-based (U-B) VMPr Recovering method.

For the presented evaluation, the total power consumption of all PMs in the 400 considered experimental scenarios of 1000 time instants is 105,23 [KW], what is 47.34% better than the worst algorithm ($A27$). Clearly, it is worth choosing the right algorithm (savings up to 47.34% were found).

TABLE III
TOP-5 BEST ALGORITHMS CONSIDERING F_1 EVALUATION METRIC FOR SERVER POWER CONSUMPTION. NO OTHER POWER CONSUMPTION, AS TEMPERATURE SUBSYSTEM, IS CONSIDERED.

Ranking	Algorithm	F_1	Power Consumption [KW]
1^{st}	$A32$	0.200	105.2
2^{nd}	$A35$	0.202	107.5
3^{th}	$A36$	0.217	107.6
4^{th}	$A31$	0.257	108.7
5^{th}	$A11$	0.263	143.1

MF2: *The Top-5 best evaluated algorithms (see Table III) consider prediction-based VMPr Triggering and update-based VMPr Recovering methods.*

Clearly, using a VMPr phase is very useful when minimizing energy consumption. To do it, a relevant decision is to choose the right VMPr Triggering and VMPr Recovering methods. Therefore, as a future work we are currently working on including a VMPr phase into commercial implementations of real-world cloud middlewares (e.g. OpenStack), taking into account prediction-based VMPr Triggering and update-based VMPr Recovering methods as well as other novel alternatives.

V. CONCLUSIONS AND FUTURE DIRECTIONS

This work evaluated 36 different VMP algorithms in a previously proposed two-phase optimization scheme in order to identify which one presents the best solutions for power consumption minimization (*Research Question 1*, answered with *Main Finding 1*, i.e. the $A32$ algorithm). Additionally, a main goal of the experimental evaluation is to identify under what circumstances to trigger a placement reconfiguration or a VMPr phase (*Research Question 2*, answered with *Main Finding 2*, i.e. prediction-based Triggering) as well as what to do with VMs requested during VMPr recalculation times (*Research Question 3*, answered with *Main Finding 2*, i.e. update-based Recovering).

It is worth mentioning that power consumption management has become a crucial study issue in the provider-oriented VMP literature [2]. In this context, $A32$ algorithm shows a promising future for green cloud datacenters efficiently managing VMP decisions. A sensibility analysis is proposed as future work, in order to be able to analyze the variability of each algorithm behavior.

Considering that the prediction-based VMPr Triggering method outperformed other evaluated alternatives, future work may include the following *Research Questions* (RQs):

- ***RQ4:*** *which other techniques could be considered more appropriate for VMPr Triggering methods?.*
- ***RQ5:*** *how important is to accurately predict when to trigger a VMPr phase in VMP problems?.*
- ***RQ6:*** *rather than predicting future objective function values, what other parameters could be evaluated for VMPr Triggering methods?.*
- ***RQ7:*** *which machine learning (ML) techniques could be paired with the existing VMPr methods for prediction purposes?.*
- ***RQ8:*** *which other algorithms could be implemented to trigger the VMPr phase paired with the exiting Triggering and Recovering methods?.*
- ***RQ9:*** *is it worth bringing back to the datacenter a VM request that has already been sent to another federated datacenter?. If this is the case, when and how is it worth doing this migration?.*

Finally, alternative VMPr Recovering methods could also be proposed, as well as novel VMPr techniques and formulation parameters, just to cite a few alternatives, towards a practical and efficient management tool for Green Cloud Computing Datacenters.

ACKNOWLEDGMENT

This research work was supported by CONACYT (Paraguay), in the context of the PINV15-781 *Software-defined Datacenters* research project grant.

REFERENCES

[1] F. López-Pires, B. Barán, L. Benítez, S. Zalimben, and A. Amarilla, "Virtual machine placement for elastic infrastructures in overbooked cloud computing datacenters under uncertainty," *Future Generation Computer Systems*, vol. 79, pp. 830–848, 2018.

[2] F. López-Pires and B. Barán, "Cloud computing resource allocation taxonomies." *IJCC*, vol. 6, no. 3, pp. 238–264, 2017.

[3] 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.

[4] 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.

[5] A. Amarilla, S. Zalimben, L. Benítez, F. López-Pires, and B. Barán, "Evaluating a two-phase virtual machine placement optimization scheme for cloud computing datacenters," in *2017 Metaheuristics International Conference (MIC)*, 2017, pp. 99–108.

[6] S. Fang, R. Kanagavelu, B.-S. Lee, C. H. Foh, and K. M. M. Aung, "Power-efficient virtual machine placement and migration in data centers," in *Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing*. IEEE, 2013, pp. 1408–1413.

[7] F. López-Pires, B. Barán, C. Pereira, M. Velázquez, and O. González, "Towards elastic virtual machine placement in overbooked openstack clouds under uncertainty," in *VI Jornadas de Cloud Computing & Big Data (JCC&BD)(La Plata, 2018)*, 2018.

[8] 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.

[9] 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.