An Experimental Comparison of Algorithms for Virtual Machine Placement Considering Many Objectives

Fabio López-Pires[1 2], Benjamín Barán[2],
Augusto Amarilla[2], Leonardo Benítez[2], Rodrigo Ferreira[2], Saúl Zalimben[2]
fabio.lopez@pti.org.py, bbaran@pol.una.py,
{agu.amarilla, benitez.leonardo.py, rodrigofepy, szalimben93}@gmail.com
[1]Itaipu Technological Park
[2]National University of Asunción
Paraguay

ABSTRACT

Cloud computing datacenters provide thousands to millions of virtual machines (VMs) on-demand in highly dynamic environments, requiring quick placement of requested VMs into available physical machines (PMs). Due to the randomness of customer requests, the Virtual Machine Placement (VMP) should be formulated as an online optimization problem. This work presents a formulation of a VMP problem considering the optimization of the following objective functions: (1) power consumption, (2) economical revenue, (3) quality of service and (4) resource utilization. To analyze alternatives to solve the formulated problem, an experimental comparison of five different online deterministic heuristics against an offline memetic algorithm with migration of VMs was performed, considering several experimental workloads. Simulations indicate that First-Fit Decreasing algorithm (A4) outperforms other evaluated heuristics on average. Experimental results prove that an offline memetic algorithm improves the quality of the solutions with migrations of VMs at the expense of placement reconfigurations.

Keywords

Virtual Machine Placement; Cloud Computing; Online Algorithms; Experimental Comparison; Optimization.

1. INTRODUCTION

A main concern of cloud datacenters design is to efficiently manage available resources in order to improve performance and reduce energy consumption of a given computational infrastructure. Most of the time, servers operate in a very low energy-efficiency region (i.e. between 10% and 50% of resource utilization), even considering that workload peaks rarely occur in practice [2]. Several methods were considered for addressing this issue, being the virtualization of resources the most studied for cloud computing datacenters.

LANC '16, October 13-14, 2016, Valparaiso, Chile
© 2016 ACM. ISBN 978-1-4503-4591-0/16/10...$15.00
DOI: http://dx.doi.org/10.1145/2998373.2998374

In cloud computing datacenters, resources are allocated and released dynamically trying to serve requested demands. In this context, deciding the right allocation of virtual machines (VMs) into physical machines (PMs) is known in the specialized literature as Virtual Machine Placement (VMP).

It is relevant to remember that the VMP problem is a NP-Hard combinatorial optimization problem [17]. From an Infrastructure as a Service (IaaS) provider's perspective, the VMP problem must be formulated as an online problem (considering that requests from customers are unknown a priori) and it should be solved with short time constraints. Consequently, solution techniques with low computational complexity are very studied for this VMP environment [13].

The most studied heuristics in the VMP literature are: Best-Fit (BF), Best-Fit Decreasing (BFD), First-Fit (FF), First-Fit Decreasing (FFD) and Worst-Fit (WF) [13].

As presented by López-Pires and Barán in [13], over 60 different objectives have been proposed for VMP problems. The number of considered objective functions may rapidly increase once a complete understanding of the VMP problem is accomplished for practical problems, where many different parameters should be ideally taken into account.

This work focuses in online formulations of a provider-oriented VMP problem in federated-clouds [13] to provide relevant information for design and implementation of resource management systems, more specifically, resource allocation algorithms for VMP problems. Many relevant objectives for IaaS providers are considered in this work. To solve the formulated online problem, five of the most studied heuristics were compared against a Memetic Algorithm (MA) solving an offline formulation of the problem.

Background and Motivation

López-Pires and Barán recently proposed in [9] and [12] offline formulations of VMP problems considering many objectives, proposing novel memetic algorithms for solving the formulated problems. Considering the on-demand model of cloud computing with dynamic resource provisioning and dynamic workloads of cloud applications [14], the resolution of the VMP problem should be performed as fast as possible. Consequently, applying only offline formulations of the VMP problem with meta-heuristics as solution technique, may not be appropriate for these dynamic environments. Therefore, as previously mentioned, solution techniques with low computational complexity (e.g. heuristics) are very studied for solving online formulations of VMP problems [13].

To help IaaS providers in the design and implementation of resource allocation algorithms considering many objective functions, the following research question must be answered: *Which heuristics are most suitable for solving online VMP problems in federated-clouds considering many objectives?*

In this context, Fang et al. presented in [6] a validation of a proposed power-aware algorithm against well-known heuristics such as: BF, FF and WF. Additionally, Jin et al. studied FFD and Dominant Resource First (DRF) algorithms [10], while Anand et al. evaluated in [1] a proposed Integer Linear Programming (ILP) formulation and an FDD algorithm to attend common limitations of ILP algorithms for large instances of NP-Hard optimization problems. On the contrary, this work does not compare novel algorithms against well-known heuristics as presented in the above mentioned works. The main goal of the presented experimental comparison is to analyze the online nature of the VMP problem optimizing many different objectives, identifying advantages and disadvantages of well-known online heuristics against offline alternatives such as the MA proposed in [9].

Experimental results presented by Ihara et al. in [9] recommend the combination of many objective functions into a single objective for IaaS environments, given the requirement of obtaining solutions in a short period of time.

To compare the considered algorithms (see Section 3), a provider-oriented VMP problem formulation is proposed for the optimization of four objective functions: (1) power consumption, (2) economical revenue, (3) quality of service and (4) resource utilization (see Section 2). To combine the above mentioned objectives, a weighted sum method is considered. Experiments were performed to define appropriate weights for each objective function (see Section 4).

The remainder of this paper is structured in the following way: Section 2 summarizes the proposed provider-oriented VMP problem formulation with many objectives. Section 3 describes the considered algorithms to solve the formulated problem, while Section 4 presents experimental workloads, obtained results and main findings of the comparison. Finally, conclusions and future work are left to Section 5.

2. PROBLEM FORMULATION

This section presents a formulation of a VMP problem considering the optimization of the following objective functions: (1) power consumption, (2) economical revenue, (3) quality of service and (4) resource utilization. According to the taxonomy presented in [13], this work focuses on a provider-oriented VMP for federated-cloud deployments, considering two formulation types: (1) online and (2) offline.

An online problem formulation is considered when an algorithm makes decisions on-the-fly without knowing upcoming events (e.g. online heuristics for VMP problems) [4]. On the other hand, if an algorithm has a complete knowledge of the future events of a problem instance, the formulation is called offline (e.g. MAs for VMP problems used in [9] and [12]).

The formulation of the proposed provider-oriented VMP problem is based on [9] and could be enunciated as:

Given a cloud infrastructure composed by a set of PMs (H), a dynamic scenario composed by a set of VMs requested at each discrete time t ($V(t)$) and the current placement of VMs into PMs ($P(t)$), it is incrementally sought a placement of $V(t)$ into H for the next time instant ($P(t+1)$), satisfying the constraints and optimizing the considered objectives.

In online algorithms for solving the proposed VMP problem, placement decisions are performed at each discrete time t, without knowledge of upcoming VM requests. On the other hand, an offline algorithm have knowledge of the complete set of VM requests in order to decide the placement of these VMs into available PMs. Consequently, the current placement is not necessary for offline algorithms.

2.1 Input Data

The proposed formulation of the VMP problem models a cloud computing datacenter, composed by PMs and VMs, receiving the following information as input data:

- a set of n available PMs and their specifications;
- a set of $m(t)$ VMs at each discrete time t and their specifications;
- the current placement at each discrete time t.

The set of PMs is represented as a matrix $H \in \mathbb{R}^{n\times 4}$, as presented in (1). Each PM H_i is represented by its physical resources. This work considers 3 different physical resources (Pr_1-Pr_3): CPU [ECU], RAM memory [GB] and network capacity [Mbps]. It is important to mention that the proposed notation is general enough to include more characteristics associated to physical resources. Finally, the maximum power consumption [W] is also considered, i.e.

$$H = \begin{bmatrix} Pr_{1,1} & Pr_{2,1} & Pr_{3,1} & pmax_1 \\ \dots & \dots & \dots & \dots \\ Pr_{1,n} & Pr_{2,n} & Pr_{3,n} & pmax_n \end{bmatrix} \quad (1)$$

where:

$Pr_{1,i}$:	Processing resources of H_i in [ECU];
$Pr_{2,i}$:	Memory resources of H_i in [GB];
$Pr_{3,i}$:	Network capacity resources of H_i in [Mbps];
$pmax_i$:	Maximum power consumption of H_i in [W];
n:	Number of PMs.

The set of VMs requested by customers at each discrete time t is represented as a matrix $V(t) \in \mathbb{R}^{m\times 5}$, as presented in (2). In this work, each VM V_j requires 3 different virtual resources (Vr_1-Vr_3): CPU [ECU], RAM memory [GB] and network capacity [Mbps]. It is important to mention that the notation could represent any other set of virtual resources such as: Block Storage or even Graphics Processing Unit (GPU). An economical revenue R_j [\$] is associated to each VM V_j as well as a priority level represented as a Service Level Agreement (SLA). The VMs try to lease the requested virtual resources for a fixed period of discrete time.

$$V(t) = \begin{bmatrix} Vr_{1,1} & Vr_{2,1} & Vr_{3,1} & SLA_1 & R_1 \\ \dots & \dots & \dots & \dots & \dots \\ Vr_{1,m(t)} & Vr_{2,m(t)} & Vr_{3,m(t)} & SLA_{m(t)} & R_{m(t)} \end{bmatrix} \quad (2)$$

where:

$Vr_{1,j}$:	Processing requirements of V_j in [ECU];
$Vr_{2,j}$:	Memory requirements of V_j in [GB];
$Vr_{3,j}$:	Network requirements of V_j in [Mbps];
R_j:	Economical revenue for attending V_j in [\$];
SLA_j:	SLA of V_j, where $SLA_j \in \{1, 2, \dots, s\}$, being s the highest priority level;
$m(t)$:	Number of VMs at each discrete time t, then $m(t) \in \{1, \dots, m_{max}\}$;
m_{max}:	Maximum number of acceptable VMs.

Once a VM V_j is powered-off by the customer, its virtual resources are released, so the provider can reuse them. For simplicity, in what follows the index j is not reused; therefore, for this work V_j is not a function of time.

The current placement of VMs into PMs ($P(t)$) is also considered as an input data, taking into account that the placement of requested VMs is performed incrementally at each discrete time t. The placement at each discrete time t is represented as a matrix $P(t) \in \mathbb{R}^{m(t)\times n}$, defined as:

$$P(t) = \begin{bmatrix} P_{1,1}(t) & P_{1,2}(t) & \dots & P_{1,n}(t) \\ \dots & \dots & \dots & \dots \\ P_{m(t),1}(t) & P_{m(t),2}(t) & \dots & P_{m(t),n}(t) \end{bmatrix} \quad (3)$$

where:

$P_{j,i}(t) \in \{0, 1\}$: Indicates if V_j is allocated ($P_{j,i}(t) = 1$) or not ($P_{j,i}(t) = 0$) for execution on a PM H_i (i.e., $P_{j,i}(t) : V_j \to H_i$) at a discrete time t.

2.2 Output

The result of the proposed VMP problem at each discrete time t is a new placement for the next time instant ($P(t+1)$). This is represented by a matrix $P(t+1) \in \mathbb{R}^{m(t+1)\times n}$.

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

where:

$P_{j,i}(t+1) \in \{0, 1\}$: Indicates if V_j is allocated ($P_{j,i}(t) = 1$) or not ($P_{j,i}(t) = 0$) for execution on a PM H_i (i.e., $P_{j,i}(t) : V_j \to H_i$) at instant $t + 1$.

2.3 Constraints

2.3.1 Constraint 1: Unique Placement of VMs

A VM V_j should be located to run on a single PM H_i or alternatively for each V_j such that $SLA_j < s$, it could not be located into any PM. Consequently, this placement constraint is expressed as:

$$\sum_{i=1}^{n} P_{j,i}(t) \leq 1 \quad \forall j \in \{1, \dots, m(t)\} \quad (5)$$

2.3.2 Constraint 2: Assure SLA Provisioning

A VM V_j with the highest level of SLA (i.e. $SLA_j = s$) must be mandatorily allocated to run on a PM H_i. Consequently, this constraint is mathematically expressed as:

$$\sum_{i=1}^{n} P_{j,i}(t) = 1 \quad \forall j \text{ such that } SLA_j = s \quad (6)$$

2.3.3 Constraints 3-5: Physical Resources of PMs

A PM H_i must have sufficient available resources to meet the requirements of all VMs V_j that are allocated to run on H_i. Consequently, these constraints can be formulated as:

$$\sum_{j=1}^{m(t)} Vr_{1,j} \times P_{j,i}(t) \leq Pr_{1,i} \quad (7)$$

$$\sum_{j=1}^{m(t)} Vr_{2,j} \times P_{j,i}(t) \leq Pr_{2,i} \quad (8)$$

$$\sum_{j=1}^{m(t)} Vr_{3,j} \times P_{j,i}(t) \leq Pr_{3,i} \quad (9)$$

$\forall i \in \{1, ..., n\}$, i.e. for all PM H_i.

Physical resources are considered as resources available for VMs, without considering resources for the PM's hypervisor.

2.4 Objective Functions

Each of the considered objective functions must be formulated in a single optimization context (i.e. minimization or maximization) and each objective function cost must be normalized to be comparable and combinable as a single objective. This work normalizes each objective function cost by calculating $\hat{f}_i(x) \in \mathbb{R}$, where $0 \leq \hat{f}_i(x) \leq 1$.

2.4.1 Power Consumption Minimization

Based on Beloglazov et al. [3], this work models the power consumption of PMs considering a linear relationship with the CPU utilization of PMs. This can be represented by the sum of the power consumption of each PM H_i.

$$min\ f_1(x) = \sum_{i=1}^{n} ((pmax_i - pmin_i) \times Ur_{1,i}(t) + pmin_i) \times Y_i(t) \quad (10)$$

where:

$f_1(x)$: Total power consumption of PMs;

$pmin_i$: 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.

2.4.2 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 [14]. Consequently, this work considered a federated-cloud deployment architecture, where a main provider may attend requested resources that are not able to be provided (e.g. a workload peak) by transparently leasing low-price resources at a federated provider [8].

In this work, the maximization of the total economical revenue that a provider receives for attending the requirements of its customers is achieved by minimizing the total costs of leasing resources from alternative datacenters of the cloud federation. Equation (11) represents the mentioned leasing costs, defined by 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. The pricing scheme may depend on the particular agreement between providers of the cloud federation [8]. Consequently, 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 price in markets (R_j). This objective may be formulated as:

$$min\ f_2(x) = \sum_{j=1}^{m(t)} (R_j \times X_j(t) \times \hat{X}_j) \quad (11)$$

where:

$f_2(x)$:	Total costs for not allocating VMs in the main provider;
$X_j(t) \in \{0,1\}$:	Indicates if V_j is allocated for execution on a PM ($X_j(t) = 1$) or not ($X_j(t) = 0$) 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 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 providers. The value of $\hat{X}_j$ depends on the agreement celebrated between federated providers.

2.4.3 Quality of Service Maximization

In this work, the quality of service (QoS) maximization proposes the location of the maximum number of VMs with the highest level of priority associated to the SLA prior to VMs with smaller SLA. In case the main provider allocates a VM in a federated provider the QoS is considered as 0. In order to evaluate this objective in a minimization context, the total of SLA violations is minimized and formulated as:

$$min\ f_3(x) = \sum_{j=1}^{m(t)} (\hat{C}^{SLA_j} \times SLA_j \times X_j(t) \times \hat{Y}_j) \qquad (12)$$

where:

$f_3(x)$:	Total SLA violations figure for a given placement;
$\hat{C}$:	Constant, large enough to prioritize services with a larger SLA over the ones with a lower SLA;
$\hat{Y}_j \in \{0,1\}$:	Indicates if V_j is allocated for its execution on the main provider ($\hat{Y}_j = 0$) or on an alternative datacenter of the cloud federation ($\hat{Y}_j = 1$).

2.4.4 Resources Utilization Maximization

An efficient utilization of resources is a relevant management challenge to be addressed by IaaS providers. This work proposes the resource utilization maximization strategy 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 formulated in Equation (13).

$$min\ f_4(x) = \frac{\sum_{i=1}^{n} \left[1 - \left(\frac{Ur_{1,i}(t) + \cdots + Ur_{r,i}(t)}{r}\right)\right] \times Y_i(t)}{\sum_{i=1}^{n} Y_i(t)} \qquad (13)$$

where:

$f_4(x)$:	Average ratio of wasted resources;
$Ur_{1,i}(t)$:	Utilization ratio of resource 1 (in this case CPU) by H_i at instant t;
$Ur_{r,i}(t)$:	Utilization ratio of resource r (any resource) by H_i at instant t;
r:	Number of considered resources. In this paper 3: CPU, RAM memory and network capacity;

The following sub-section summarizes the main considerations taken into account to combine the four presented objective functions into a single objective.

2.5 Scalarization Method

Experimental results presented in [9] recommend the combination of many normalized objective functions into a single objective (e.g. minimum distance to origin) for IaaS environments. This work considers a weighted sum method to combine many objectives into a single objective.

As previously mentioned, each objective function cost must is normalized to be comparable and combinable as a single objective. This work normalizes each objective function cost by calculating $\hat{f}_i(x) \in \mathbb{R}$, where $0 \leq \hat{f}_i(x) \leq 1$, as defined in [16]:

$$\hat{f}_i(x) = \frac{f_i(x) - f_i(x)_{min}}{f_i(x)_{max} - f_i(x)_{min}} \qquad (14)$$

where:

$\hat{f}_i(x)$:	Normalized cost of objective function $f_i(x)$;
$f_i(x)$:	Cost of objective function $f_i(x)$;
$f_i(x)_{min}$:	Minimum possible cost for $f_i(x)$;
$f_i(x)_{max}$:	Maximum possible cost for $f_i(x)$.

Finally, the four previously presented normalized objective functions are combined into a single objective considering a weighted sum method, expressed as follows:

$$min\ f(x) = \sum_{i=1}^{q} \hat{f}_i(x) \times w_i \qquad (15)$$

where:

$f(x)$:	Single objective function combining each $f_i(x)$;
$\hat{f}_i(x)$:	Normalized cost of objective function $f_i(x)$;
w_i:	Weight of importance associated to $f_i(x)$;
q:	Number of objective functions. In this case 4.

In this work, values of each weight w_i associated to an objective function $f_i(x)$ were obtained empirically by analyzing a large number of experiments to be presented in Section 4.

3. EVALUATED ALGORITHMS

To analyze alternatives to solve the formulated VMP problem (see Section 2), an experimental comparison of five different online deterministic heuristics against an offline memetic algorithm with migration of VMs was performed. This section summarizes the six different algorithms considered in the experimental comparison presented in this work.

3.1 A1: First-Fit (FF)

In the First-Fit (FF) algorithm, requested VMs V_j are allocated on the first PM H_i with available resources (see Section 2.3). Interested readers can refer to [6] for details on FF, BF and WF algorithms applied to VMP problems.

3.2 A2: Best-Fit (BF)

The Best-Fit (BF) algorithm, allocates requested VMs V_j on the first PM H_i with available capacity from a sorted list of PMs in increasing order by a score associated to each PM. The score of a PM H_i is determined as a sum of its ratios of unutilization of resources, as detailed in [6].

3.3 A3: Worst-Fit (WF)

In the Worst-Fit (WF) algorithm, VMs V_j are allocated on the first available PM H_i of a decreasingly ordered list of PMs based on the score of the PM, inversely to the operation of the BF algorithm. For more details, refer to [6].

3.4 A4: First-Fit Decreasing (FFD)

The First-Fit Decreasing (FFD) algorithm operates similarly to the previously presented FF algorithm. The main difference with the FF algorithm is that FFD algorithm sorts the list of requested VMs V_j in decreasing order by requested CPU resources, as described in [7].

3.5 A5: Best-Fit Decreasing (BFD)

The Best-Fit Decreasing (BFD) algorithm has a similar behavior to the BF algorithm. The main difference between the BF algorithm and the BFD one is that the BFD sorts the list of requested VMs V_j in decreasing order by requested CPU resources. Interested readers can refer to [5] for details on the BFD algorithm applied to VMP problems.

3.6 A6: Memetic Algorithm (MA)

A Memetic Algorithm (MA) is considered for an offline alternative to solve the formulated VMP problem taking into account many objectives (see Section 2). This algorithm is based on the one proposed in [9] by Ihara et al.

The considered algorithm may be classified as a metaheuristic, following an evolutionary process to find appropriate solutions. Basically, the considered MA follows this evolutionary behavior: solutions are selected from an evolutionary set of solutions (or population). Crossover and mutation operators are applied as usual, and eventually solutions are repaired, as there may be unfeasible solutions. Improvements of solutions of the evolutionary population may be generated using local optimization operators. Next, a new evolutionary population is selected from the union of the (until that generation) best population and the set of improved solutions. The evolutionary process is repeated until the algorithm meets a stopping criterion (such as a maximum number of generations), returning from the evolutionary population the solution with minimum cost of $f(x)$.

4. EXPERIMENTAL COMPARISON

In this work, an experimental comparison of five of the most studied online deterministic heuristics was performed considering several experimental workloads. The quality of solutions obtained by the evaluated online heuristics was compared to the average performance (in ten runs) of an offline MA that has complete knowledge of future VM requests and the ability of migrating VMs between PMs.

4.1 Experimental Environment

The six evaluated algorithms were implemented using ANSI C programming language. The source code is available online[1], as well as all the considered experimental data. Experiments were performed on a Linux Operating System with an Intel Core i7 2.3 GHz CPU and 12 GB of RAM memory.

Physical resources (matrix H) represent an homogeneous IaaS cloud composed by 10 PMs with the following specifications: 8 [ECU] of CPU, 10 [GB] of RAM memory, 780 [Mbps] of network capacity and 960 [W] of maximum power consumption (see Section 2.1 for notation details).

In this work, requested virtual resources (matrix $V(t)$) were considered using four different workload traces generated using a *Workload Trace Generator* for provider-oriented VMP problems [15] available for research purposes[2].

[1]http://github.com/DynamicVMP/vmpCompetitiveAnalysis

[2]http://wtg.cba.com.py

Table 1: Considered Data for Generated Experimental Workload Traces (W_1 to W_4). Details in [15].

Parameter Description	Input Data
1. Environment	No overbooking, No elasticity
2. Workload Trace Duration [t]	100
3. Number of Cloud Datacenters	1
4. Number of Cloud Services	100
5. Number of VMs per Service	1
6. VMs creation time [t]	W_1: Poisson($\lambda = 10$) W_2: Poisson($\lambda = 50$) W_3: Poisson($\lambda = 70$) W_4: Uniform(0,100)
7. CPU Resources [ECU]	Uniform(1,8)
8. RAM Resources [GB]	Uniform(1,8)
9. Network Capacity [Mbps]	Uniform(10,1000)
10. Revenue of VMs (R_j) [$]	Uniform(0.1,1.5)
11. SLA Level of VMs (SLA_j)	Uniform(1,5)

Each considered workload trace simulates customers requests to allocate a set of 100 VMs following different Probability Distribution Functions (PDFs) for the VM request arrivals. Three workload traces follow a Poisson PDF with different expected values (λ): (W_1): $\lambda = 10$, (W_2): $\lambda = 50$ and (W_3): $\lambda = 70$. In this case, different values of λ may represent workload peaks at different time instants t (see Figure 1). The fourth workload trace (W_4) follows an Uniform PDF, representing a stable workload of VM requests. All parameters considered for generating the experimental workload traces are presented in Table 1.

In order to effectively analyze the described federated-cloud deployment architecture and provider-oriented VMP formulation considering many objectives, experimental workload traces requested more virtual resources than the available ones in the considered main cloud computing datacenter.

Experiments could be summarized as: For each considered workload trace (W_1 to W_4), one run of the following deterministic heuristics was performed in an online context: (1) FF, (2) BF, (3) WF, (4) FFD and (5) BFD. Considering the randomness of the MA compared against the previously mentioned heuristics, ten runs of the mentioned algorithm were performed. The average values of the ten runs were considered for the experimental comparison, as summarized in Table 2, where the offline MA clearly outperforms all 5 online heuristics as expected, given that it uses complete knownledge of VM requests and migration of VMs to optimize the objective function presented in Equation 15.

4.2 Objective Functions Weights

To determine appropriate values for the weights w_i associated to each objective function $f_i(x)$, an exploration of the VMP problem domain was performed. In this context, 1000 feasible solutions (x_1 to x_{1000}) were randomly generated by the MA (A6) considering: H as described in Section 4.1, $V(t)$ as presented in *entorno00-1* (a *benchmark* available online[2]), highest priority of VMs $s = 4$ and $\hat{C} = 1000$.

Obtained values of each objective function $f_i(x)$ were normalized in $\hat{f}_i(x)$, as described in Section 2.5. Consequently, each weight w_i was defined as:

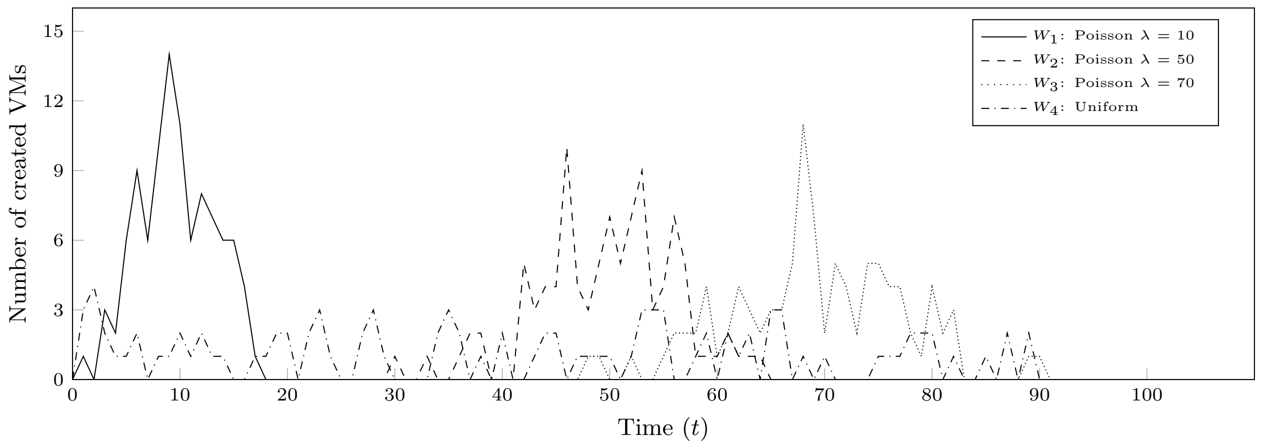

Figure 1: Experimental Workload Traces: Number of created VMs as a function of time t.

$$w_i = \frac{1000}{\sum_{k=1}^{1000} \hat{f}_i(x_k)} \qquad (16)$$

resulting:

w_1: 1.3903; w_2: 2.1379; w_3: 2.7393; w_4: 1.4586.

4.3 Comparison of Online Heuristics

This section summarizes the main findings obtained in the experimental comparison of algorithms for the VMP formulation with many objectives presented in Section 2.

The main goal of the experimental comparison presented in this section is to define *which heuristics are most suitable for solving online formulations of provider-oriented VMP problems in federated-clouds considering many objectives.*

Table 2 summarizes the costs of objective function $f(x)$, that combines the four considered objectives (see Section 2.5), summarizing performed experiments. It worth noting that according to the experimental results presented in Table 2, there is no evaluated heuristic that outperforms all other alternatives in all experimental workload traces. Consequently, Table 2 also presents the average costs of objective function $f(x)$ in the 4 experimental workloads as well as the corresponding ranking. It seems that FFD algorithm is the best heuristic amount the five considered ones closely followed by BFD (logically, MA is first in this ranking).

To better understand the presented comparison, Table 3 details the average normalized costs of each objective function $f_1(x)$ to $f_4(x)$, to analyze particular preferences of IaaS providers for the optimization of the considered objectives (e.g. power consumption could be more important in hours where the electricity price is higher).

Based on the information presented in Tables 2 and 3, the Main Findings (MF) of the experimental comparison performed in this paper are summarized as follows:

MF1: *There is no evaluated heuristic that can clearly be considered as the best alternative for all objective functions, considered simultaneously.*

Additionally, none of the evaluated heuristics performed equally well in all 4 experimental workloads (see Table 2). Consequently, an heuristic performing good enough in average when considering different types of workloads could be sufficiently convenient.

A more detailed evaluation could be performed to obtain information for conjuntural preferences of IaaS providers (see Table 3, where the best heuristic for each objective function and each workload trace is highlighted in the last column).

MF2: *FFD heuristic (A4) was the algorithm that outperforms all other evaluated heuristics taking into account average results in performed experiments.*

According to the average performance of each evaluated heuristic (see Table 2), the following ranking was built: (1st) FDD (A4), (2nd) BFD (A5), (3th) BF (A2), (4th) FF (A1), and (5th) WF (A3), where BFD follows very close the average performance of FDD (with a difference of 1.75%).

MF3: *WF algorithm (A3) is suggested for workloads that can be considered stable (i.e. no workload peaks).*

For experimental workload trace W_4 that considers an Uniform PDF for VM request arrivals, A3 clearly presented the best performance obtaining minimum average results in Table 2. It also performed as the best algorithm for 3 out of the 4 objectives, considering only W_4 (see Table 3).

MF4: *The WF (A3), BFD (A5), FFD (A4) and BFD (A5) algorithms are recommended for $f_1(x)$, $f_2(x)$, $f_3(x)$ and $f_4(x)$ objective functions respectively, when there is a preferred objective function (lexicographic order).*

The BFD algorithm (A5) performed as the best algorithm on average for $f_2(x)$ and $f_4(x)$. For $f_1(x)$ the best alternative seems to be A3 while A4 could be considered as the best for $f_3(x)$ (see Table 3).

MF5: *As expected, the offline MA (A6) outperformed all evaluated online heuristics in all experimental workloads.*

An offline MA was compared to the five evaluated heuristics to experimentally demonstrate that online decisions made along a simulation affects the quality of solutions. Clearly, offline algorithms such as the considered MA present a substantial advantage over online heuristics when considering the quality of solutions for the objective function $f(x)$ (see Table 2). This advantage is presented for the following two main reasons: (1) offline algorithms have a complete knowledge of future events of a problem instance and (2) only A6 considered migrations of VMs between PMs in the comparison, reconfiguring the placement when convenient.

Table 2: Objective Function Costs of Evaluated Algorithms.

Workload \ Algorithm	**A1: FF**	**A2: BF**	**A3: WF**	**A4: FFD**	**A5: BFD**	**A6: MA**
W_1: *Poisson* $\lambda = 10$	3.2927	3.3098	3.5250	3.0205	3.1392	**2.6096**
W_2: *Poisson* $\lambda = 50$	2.4602	2.5112	2.4811	2.4602	2.4555	**2.0001**
W_3: *Poisson* $\lambda = 70$	1.7054	1.6458	1.7054	1.6458	1.6458	**1.3588**
W_4: *Uniform*	3.1875	3.1556	3.0489	3.0907	3.1556	**2.3420**
Average	2.6615	2.6556	2.6901	2.5543	2.5990	**2.0776**
Ranking	5th	4th	6th	2nd	3th	**1st**

Table 3: Objective Functions Costs of Evaluated Heuristics.

Workload	$f_i(x)$ \ Heuristic	**A1: FF**	**A2: BF**	**A3: WF**	**A4: FFD**	**A5: BFD**	**Best Heuristic**
W_1	$f_1(x)$	0.9240	0.9107	**0.8929**	0.9255	0.9208	A3
	$f_2(x)$	0.0372	0.0373	0.0391	**0.0359**	**0.0359**	A4,A5
	$f_3(x)$	0.6104	0.6139	0.6670	**0.5101**	0.5611	A4
	$f_4(x)$	0.1756	0.1933	0.2556	0.1778	**0.1679**	A5
W_2	$f_1(x)$	0.5900	0.5756	**0.5612**	0.5900	0.5903	A3
	$f_2(x)$	**0.0404**	0.0406	0.0420	**0.0404**	**0.0404**	A4,A5
	$f_3(x)$	0.4790	0.4975	**0.4715**	0.4790	0.4790	A3
	$f_4(x)$	0.1652	0.1789	0.2189	0.1652	**0.1618**	A5
W_3	$f_1(x)$	**0.4146**	0.4198	0.4203	0.4198	0.4198	A1
	$f_2(x)$	0.0243	**0.0235**	0.0245	**0.0235**	**0.0235**	A2,A4,A5
	$f_3(x)$	0.3155	**0.3003**	0.3103	**0.3003**	**0.3003**	A2,A4,A5
	$f_4(x)$	0.1457	**0.1296**	0.1497	**0.1296**	**0.1296**	A2,A4,A5
W_4	$f_1(x)$	0.8549	**0.8544**	0.8694	0.8598	**0.8544**	A2,A5
	$f_2(x)$	0.0375	0.0364	**0.0356**	0.0371	0.0364	A3
	$f_3(x)$	0.5311	0.5283	**0.4884**	0.4931	0.5283	A3
	$f_4(x)$	0.3177	0.3034	**0.2919**	0.3188	0.3034	A3
Average	$f_1(x)$	0.6959	0.6901	**0.6859**	0.6988	0.6963	A3
	$f_2(x)$	0.0349	0.0345	0.0353	0.0343	**0.0341**	A5
	$f_3(x)$	0.4841	0.4851	0.4844	**0.4457**	0.4672	A4
	$f_4(x)$	0.2011	0.2014	0.2291	0.1979	**0.1907**	A5

Table 4: VMs Migration Overhead for MA (A6).

Workload \ Migration	# of VMs	Memory in [GB]
Poisson $\lambda = 10$	1493.7	4865.9
Poisson $\lambda = 50$	740.0	2775.1
Poisson $\lambda = 70$	643.6	2415.5
Uniform	1416.7	5329.8

Having a complete knowledge of future VMs requests is considered unrealistic for IaaS environments [4]. Consequently, online algorithms, such as the heuristics evaluated in this work, are a good alternative for VMP problems in IaaS environments. The evaluated offline MA (A6) improves the quality of solutions with migrations of VMs between PMs, at expense of costs associated to placement reconfigurations, as seen in Table 4 where the average VMs migration overhead of each workload is presented as: (1) total number of VM migrations and (2) total RAM memory migrated.

MF6: *As expected, the execution time of all online heuristics are considerably shorter than the offline alternative.*

Considering the low complexity of the evaluated heuristics, these algorithms are able to find promising solutions in short periods of time (a main concern for IaaS providers solving VMP problems) at the expense of not exploring solutions that can potentially result in better quality solutions.

Clearly, any offline alternative that intelligently explore a large space of feasible solutions will result in higher execution times, mainly considering the exponential computational complexity of the VMP problem.

5. CONCLUSIONS AND FUTURE WORKS

This work presented a first experimental comparison of five different online heuristics for solving a proposed provider-oriented VMP problem for the optimization of the following objectives: (1) power consumption, (2) economical revenue, (3) quality of service, and (4) resource utilization. The quality of solutions obtained by online heuristics was compared to the best performance of an offline MA that have complete knowledge of future VM requests and the capacity of VM migration. A previously studied MA [9] was implemented taking into account that obtaining optimal solutions for large instances of the problem could result impracticable [11].

Considering the evaluation of the mentioned algorithms solving several experimental workload traces, six main findings (MF1 to MF6) were identified (see Section 4.3).

The main advantage of the offline MA (A6) over experimentally evaluated online heuristics (A1 to A5) is a better quality of solutions, as summarized in Table 2.

Unfortunately, offline alternatives such as A6 are not appropriate for dynamic environments of VMP problems for IaaS providers, where future VM requests are unknown and placement should be solved in a very short time.

In order to improve the quality of solutions obtained by the evaluated heuristics, the VMP problem could be formulated as a two-phase optimization problem. In this context, VMP problems with many objectives for an IaaS could be decomposed into two different sub-problems: (1) incremental VMP (iVMP) and (2) VMP reconfiguration (VMPr) [18].

A two-phase optimization strategy can be considered, combining both *online* (iVMP) and *offline* (VMPr) algorithms for solving each considered VMP sub-problem. The iVMP problem is considered for dynamic arriving requests when VMs should be created and removed at runtime. Consequently, this sub-problem should be formulated as an *online* problem and solved in a short period of time, where the studied heuristics could be appropriate. On the contrary, the VMPr problem is considered for improving the quality of solutions obtained by the iVMP, considering placement reconfigurations through migration of VMs. This sub-problem could be formulated *offline*, where alternative solution techniques (e.g. meta-heuristics) could be appropriate.

Authors of this work are already working on the mentioned two-phase optimization (iVMP + VMPr), considering more realistic representation of cloud infrastructures and more complex VMP environments with both overbooking and elasticity considerations [15], as well as extending experiments to more workload traces including real-world experimental workloads [4]. Finally, future directions include exploring research questions as: (1) when the VMPr problem should be triggered? and (2) what should the VMPr problem do with requests arriving during reconfiguration?

6. REFERENCES

[1] A. Anand, J. Lakshmi, and S. Nandy. Virtual machine placement optimization supporting performance SLAs. In *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, volume 1, pages 298–305. IEEE, 2013.

[2] L. A. Barroso and U. Hölzle. The case for energy-proportional computing. *IEEE computer*, 40(12):33–37, 2007.

[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*, 28(5):755–768, 2012.

[4] 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*, 24(13):1397–1420, 2012.

[5] D. Dong and J. Herbert. Energy efficient VM placement supported by data analytic service. In *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, pages 648–655. IEEE, 2013.

[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*, pages 1408–1413. IEEE, 2013.

[7] T. Ferreto, C. A. De Rose, and H.-U. Heiss. Maximum migration time guarantees in dynamic server consolidation for virtualized data centers. In *Euro-Par Parallel Processing*, pages 443–454. Springer, 2011.

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

[9] D. Ihara, F. López-Pires, and B. Barán. Many-objective virtual machine placement for dynamic environments. In *Proceedings of the 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing*. IEEE Computer Society, 2015.

[10] H. Jin, D. Pan, J. Xu, and N. Pissinou. Efficient VM placement with multiple deterministic and stochastic resources in data centers. In *Global Communications Conference (GLOBECOM), 2012 IEEE*, pages 2505–2510. IEEE, 2012.

[11] 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*, pages 203–210. IEEE Computer Society, 2013.

[12] 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.

[13] F. López-Pires and B. Barán. A virtual machine placement taxonomy. In *Proceedings of the 2015 IEEE/ACM 15th International Symposium on Cluster, Cloud and Grid Computing*. IEEE Computer Society, 2015.

[14] P. Mell and T. Grance. The NIST definition of cloud computing. *National Institute of Standards and Technology*, 53(6):50, 2009.

[15] J. Ortigoza, F. López Pires, and B. Barán. A taxonomy on dynamic environments for provider-oriented virtual machine placement. In *Proceedings of the 2016 IEEE 4th International Conference on Cloud Engineering*. IEEE Computer Society, 2016.

[16] S. B. Salem, M. Fakhfakh, D. S. Masmoudi, M. Loulou, P. Loumeau, and N. Masmoudi. A high performances cmos ccii and high frequency applications. *Analog Integrated Circuits and Signal Processing*, 49(1):71–78, 2006.

[17] B. Speitkamp and M. Bichler. A mathematical programming approach for server consolidation problems in virtualized data centers. *Services Computing, IEEE Transactions on*, 3(4):266–278, 2010.

[18] Q. Zheng, R. Li, X. Li, N. Shah, J. Zhang, F. Tian, K.-M. Chao, and J. Li. Virtual machine consolidated placement based on multi-objective biogeography-based optimization. *Future Generation Computer Systems*, 2015.