Many-Objective Optimization for Virtual Machine Placement in Cloud Computing

Fabio López-Pires and Benjamín Barán

Abstract Resource allocation in cloud computing datacenters presents several research challenges, where the Virtual Machine Placement (VMP) is one of the most studied problems with several possible formulations considering a large number of existing optimization criteria. This chapter presents the main contributions that studied for the first time Many-Objective VMP (MaVMP) problems for cloud computing environments. In this context, two variants of MaVMP problems were formulated and different algorithms were designed to effectively address existing research challenges associated to the resolution of Many-Objective Optimization Problems (MaOPs). Experimental results proved the correctness of the presented algorithms, its effectiveness in solving particular associated challenges and its capabilities to solve problem instances with large numbers of physical and virtual machines for: (1) MaVMP for initial placement of VMs (static) and (2) MaVMP with reconfiguration of VMs (semi-dynamic). Finally, open research problems for the formulation and resolution of MaVMP problems for cloud computing (dynamic) are discussed.

1 Introduction

This chapter presents contributions related to the Virtual Machine Placement (VMP) problem from a Many-Objective Optimization perspective. Provider-oriented VMP problems can be enunciated as the process of assigning physical machines (PMs) to host requested virtual machines (VMs) in multi-tenant environments. Depending on particular requirements of a cloud computing infrastructure, the VMP problem could be formulated as several different optimization problems, considering several different objective functions. It is important to notice that these requirements may change over time and be defined as dynamic resource management policies. These

F. López-Pires (✉)
Itaipu Technological Park, Hernandarias, Paraguay
e-mail: fabio.lopez@pti.org.py

B. Barán
National University of Asunción, San Lorenzo, Paraguay
e-mail: bbaran@pol.una.py

S. Chaudhary et al. (eds.), *Research Advances in Cloud Computing*,
DOI 10.1007/978-981-10-5026-8_12

particular considerations open different possible environments and formulations for the VMP problem. As a previous work by the authors, more than 60 different objective functions were identified in the specialized VMP literature [1, 2].

In real-world cloud computing infrastructures, the resolution of VMP problems could require the optimization of several objective functions in practical cases. This particular requirement could be clearly noted taking into account the large number of already studied objective functions, which could be formulated considering different possible approaches for modeling each objective function. In this context, Cloud Service Providers (CSPs) might be faced with the need to simultaneously optimize several conflicting objective functions when solving VMP problems.

It is important to consider that optimization problems simultaneously optimizing more than three objective functions are commonly known as Many-Objective Optimization Problems (MaOPs), as defined in [3]. In this context, there are several current research challenges for the resolution of MaOPs [4, 5].

Many-Objective Optimization is still considered an unexplored domain in resource management of cloud computing infrastructures [6], although there are already a few many-objective formulations proposed for the VMP problem in the specialized literature [7–9], as presented in this chapter.

The following sections present contributions and research challenges for different variants of Many-Objective VMP (MaVMP) problems, such as: (1) MaVMP for initial placement of VMs (static), (2) MaVMP with reconfiguration of VMs (semi-dynamic) and (3) MaVMP for cloud computing environments (dynamic).

2 Many-Objective VMP for Initial Placement of VMs

Considering that no many-objective formulation for the VMP problem was presented in the literature [2, 9], basic static environments such as initial placement of VMs were first studied [8]. This section presents a general many-objective optimization framework which is able to consider as many objective functions as needed when solving a MaVMP problem for initial placement of VMs (see Sect. 2.1). As an example of utilization of the presented framework, a first formulation of a MaVMP problem is presented, considering the simultaneous optimization of the following five objective functions: (1) power consumption, (2) network traffic, (3) economical revenue, (4) quality of service (QoS) and (5) network load balancing.

In the formulation of the MaVMP for initial placement of VMs to be presented in Sect. 2.2, a multilevel priority is associated to each VM, representing a Service Level Agreement (SLA) considered in the placement process, in order to effectively prioritize important VMs (e.g., in peaks situations where the total requested VMs resources are higher than available PMs resources). To solve the formulated MaVMP for initial placement of VMs, an interactive Memetic Algorithm (MA) was proposed (see Sect. 2.3) considering particular challenges associated to the resolution of a MaVMP problem, as the potentially unmanageable number of non-dominated solutions that compose a Pareto set approximation P_{known}.

2.1 Many-Objective Optimization Framework

The general many-objective optimization framework for the VMP problem proposed in [8] considers that as the number of conflicting objectives of a MaVMP problem formulation increases, the total number of non-dominated solutions normally increases (even exponentially in some cases), being increasingly difficult to discriminate among solutions using only the dominance relation [4]. For this reason, it is recommended the utilization of lower and upper bounds associated to each objective function $f_z(x)$, where $z \in \{1, \ldots, q\}$ ($L_z \leq f_z(x) \leq U_z$), to be able to iteratively reduce the number of possible non-dominated solutions of P_{known}.

A formulation of a MaVMP for initial placement of VMs, based on many objective functions and constraints to be detailed in Sect. 2.2, may be written as:

Optimize:

$$y = f(x) = [f_1(x), f_2(x), f_3(x), \ldots, f_q(x)] \quad \text{typically with } q > 3, \tag{1}$$

where for example:

$$\begin{aligned} f_1(x) &= \text{power consumption;} \\ f_2(x) &= \text{inter-VM network traffic;} \\ f_3(x) &= \text{economical revenue;} \\ f_4(x) &= \text{quality of service;} \\ f_5(x) &= \text{network load balancing;} \\ &\vdots \\ f_q(x) &= \text{any other considered objective function.} \end{aligned} \tag{2}$$

subject to constraints as:

$$\begin{aligned} e_1(x) &: \text{unique placement of VMs;} \\ e_2(x) &: \text{assure provisioning of highest SLA;} \\ e_3(x) &: \text{processing resource capacity of PMs;} \\ e_4(x) &: \text{memory resource capacity of PMs;} \\ e_5(x) &: \text{storage resource capacity of PMs;} \\ e_6(x) &: f_1(x) \in [L_1, U_1]; \\ e_7(x) &: f_2(x) \in [L_2, U_2]; \\ e_8(x) &: f_3(x) \in [L_3, U_3]; \\ e_9(x) &: f_4(x) \in [L_4, U_4]; \\ e_{10}(x) &: f_5(x) \in [L_5, U_5]; \\ &\vdots \\ e_r(x) &: \text{any other considered constraint.} \end{aligned} \tag{3}$$

2.2 Problem Formulation

A few articles have already proposed formulations of a pure multi-objective VMP problem (MVMP), considering the simultaneous optimization of at most three objective functions [10, 11]. A previous work of the authors proposed for the first time a MaVMP formulation [8]. This section presents a formulation of a MaVMP problem considering the following five objective functions to be simultaneously optimized: (1) power consumption, (2) network traffic, (3) economical revenue, (4) quality of service and (5) network load balancing. In the presented MaVMP formulation, a multilevel priority is associated to each VM considered in the placement process in order to effectively prioritize VMs. Formally, the presented offline (static) MaVMP problem for initial placement of VMs can be enunciated as [8]:

Given a set of PMs, $H = \{H_1, H_2, ..., H_n\}$, a network topology G (as illustrated in Figure 1) and a set of VMs, $V = \{V_1, V_2, ..., V_m\}$, it is sought a correct placement of the set of VMs V into the set of PMs H satisfying the r constraints of the problem and simultaneously optimizing all q objective functions defined in this formulation (as energy consumption, network traffic, economical revenue, QoS and load balancing in the network), in a pure many-objective context.

2.2.1 Input Data

The presented formulation of the MaVMP problem for initial placement of VMs models a virtualized datacenter infrastructure, composed by PMs, VMs and a network topology that interconnects PMs.

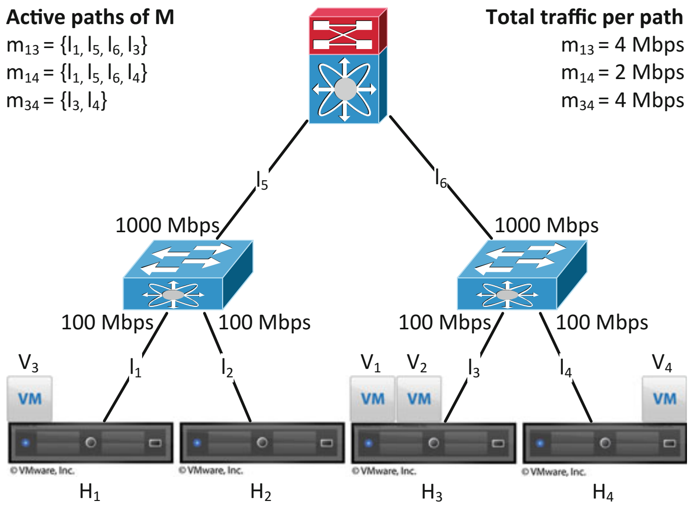

Fig. 1 Example of placement in a virtualized datacenter infrastructure, composed by PMs, a network topology and VMs

The set of PMs is represented as a matrix $H \in \mathbb{R}^{n\times 4}$. Each PM H_i is represented by processing resources of CPU (as ECU),[1] RAM [GB], storage [GB], and a maximum power consumption [W] as:

$$H_i = [Hcpu_i, Hram_i, Hhdd_i, pmax_i], \quad \forall i \in \{1, ..., n\} \tag{4}$$

where

$Hcpu_i$: Processing resources of H_i;
$Hram_i$: RAM memory resources of H_i;
$Hhdd_i$: Storage resources of H_i;
$pmax_i$: Maximum power consumption of H_i;
n: Number of PMs.

It should be mentioned that the proposed notation is general enough to include additional characteristics associated to each PM such as Graphic Processing Units (GPUs) or Network Interface Cards (NICs) just to cite a few.

As shown in the example of Fig. 1, a network topology of a virtualized datacenter is represented as:

G: Network topology;
L: Set of links l_a in G. For simplicity, links are assumed as semi-duplex in what follows;
M: Set of paths for all-to-all PM network interconnections;
K: Capacity set of the communication channels, typically in [Mbps].

The set of VMs requested by customers is represented as a matrix $V \in \mathbb{R}^{m\times 5}$. Each VM V_j requires processing resources of CPU (as ECU) (see Footnote 1), RAM [GB], and storage [GB], providing an economical revenue R_j [\$] to the provider.

A SLA is also assigned to each VM to indicate its priority level. Consequently, a V_j is represented as:

$$V_j = [Vcpu_j, Vram_j, Vhdd_j, R_j, SLA_j], \quad \forall j \in \{1, ..., m\} \tag{5}$$

where

$Vcpu_j$: Processing requirements of V_j;
$Vram_j$: Memory requirements of V_j;
$Vhdd_j$: Storage requirements of V_j;
R_j: Economical revenue for locating V_j;
SLA_j: Service Level Agreement SLA_j of a V_j. If the highest priority level is s, then $SLA_j \in \{1, \dots, s\}$;
m: Number of VMs.

[1] http://aws.amazon.com/ec2/faqs.

The traffic between VMs is represented as a matrix $T \in \mathbb{R}^{m \times m}$. Each V_j requires network communication resources [Mbps] to communicate with other VMs. The network traffic between requested VMs is represented as:

$$T_j = [T_{j1}, T_{j2}, ..., T_{jm}], \quad \forall j \in \{1, ..., m\} \tag{6}$$

where

T_{jk}: Average network traffic between V_j and V_k [Mbps]. Note that it is considered that $T_{jj} = 0$.

Figure 1 presents an example of a virtualized datacenter, composed by 4 PMs ($H = \{H_1, H_2, H_3, H_4\}$) and a network topology considering six physical network links ($L = \{l_1, l_2, l_3, l_4, l_5, l_6\}$). In this example, the set of capacity for each communication channel is $K = \{100, 100, 100, 100, 1000, 1000\}$ [Mbps]. Using shortest path, a path m_{12} between H_1 and H_2 uses links $\{l_1, l_2\}$, i.e., $m_{12} = \{l_1, l_2\}$. Analogously, $m_{13} = \{l_1, l_5, l_6, l_3\}$ and $m_{14} = \{l_1, l_5, l_6, l_4\}$, as shown in Fig. 1. All four requested VMs of Fig. 1 are correctly located into one of the available PMs.

2.2.2 Output Data

A possible solution x indicates a complete placement of each VM V_j into the necessary PMs H_i, considering the many-objective optimization criteria applied. A placement (or solution x to the proposed VMP problem) is represented as a matrix $P = \{P_{ji}\}$ of dimension (m x n), where $P_{ji} \in \{0, 1\}$ indicates if V_j is located ($P_{ji} = 1$) or not ($P_{ji} = 0$) for execution on a PM H_i (i.e., $P_{ji} : V_j \rightarrow H_i$).

2.2.3 Constraint 1: Unique Placement of VMs

A VM V_j should be executed on a single PM H_i or alternatively, it could be not located into any PM if the associated *SLA* is not the highest level of priority s (i.e., $SLA_j < s$). This constraint is mathematically formulated as:

$$\sum_{i=1}^{n} P_{ji} \leq 1 \quad \forall j \in \{1, ..., m\} \tag{7}$$

where

P_{ji}: Binary variable equals 1 if V_j is located on H_i; otherwise, it is 0.

2.2.4 Constraint 2: Assure SLA Provisioning

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

$$\sum_{i=1}^{n} P_{ji} = 1 \quad \forall j \text{ such that } SLA_j = s \tag{8}$$

2.2.5 Constraints 3–5: Physical Resources Capacities of PMs

A PM H_i must be able to meet the requirements of all VMs V_j that are located to run on H_i. In this chapter, it is not considered the overbooking of resources [12]. Consequently, these constraints can be mathematically formulated as:

$$\sum_{j=1}^{m} Vcpu_j \times P_{ji} \leq Hcpu_i \tag{9}$$

$$\sum_{j=1}^{m} Vram_j \times P_{ji} \leq Hram_i \tag{10}$$

$$\sum_{j=1}^{m} Vhdd_j \times P_{ji} \leq Hhdd_i \tag{11}$$

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

2.2.6 Adjustable Constraints

The work presented in [8] proposed the utilization of lower and upper bounds associated to each objective function to reduce the number of possible solutions of the Pareto set approximation P_{known}, when needed by a decision-maker. Consequently, this set of adjustable bounds can be formulated as the following constraints:

$$f_z(x) \in [L_z, U_z], \quad \forall z \in \{1, \ldots, q\} \tag{12}$$

A VMP problem can be defined as a many-objective optimization problem, when considering the simultaneous optimization of more than three objective functions. As a concrete example, this chapter proposes the simultaneous optimization of the following five objective functions.

2.2.7 Objective Function 1: Power Consumption Minimization

Based on [13] formulation, the work presented in [8] also proposes the minimization of power consumption, represented by the sum of the power consumption of each PM H_i:

$$f_1(x) = \sum_{i=1}^{n} ((pmax_i - pmin_i) \times Ucpu_i + pmin_i) \times Y_i \tag{13}$$

where

$f_1(x)$: Total power consumption of the PMs;
$pmin_i$: Minimum power consumption of H_i. It should be noted that $pmin_i \approx pmax_i \times 0.6$ according to [13];
$Ucpu_i$: Utilization ratio of processing resources used by H_i;
Y_i: Binary variable that equals 1 if H_i is turned on; otherwise, it is 0.

2.2.8 Objective Function 2: Inter-VM Network Traffic Minimization

Shrivastava et al. proposed in [14] the minimization of network traffic among VMs by maximizing locality. Based on this approach, the work presented in [8] proposes Eq. (14) to estimate network traffic represented by the sum of average network traffic generated by each VM V_j, that is located to run on any PM, with other VMs V_k that are located to run on different PMs.

$$f_2(x) = \sum_{j=1}^{m} \sum_{k=1}^{m} (T_{jk} \times D_{jk}) \tag{14}$$

where

$f_2(x)$: Total network traffic among VMs;
T_{jk}: Average network traffic between V_j and V_k [Mbps]. Note that it is considered that $T_{jj} = 0$.
D_{jk}: Binary variable that equals 1 if V_j and V_k are located in different PMs; otherwise, it is 0.

The traffic between two VMs V_j and V_k which are located on the same PM H_i does not contribute to increase the total network traffic given by Eq. (14); therefore, $D_{jk} = 0$ if $P_{ji} = P_{ki} = 1$.

2.2.9 Objective Function 3: Economical Revenue Minimization

Based on [11], the work presented in [8] proposes Eq. (15) for the estimation of the total economical revenue that a datacenter receives when supporting the requested resources of its customers, represented by the sum of the obtained revenue of each VM V_j that is located for execution on any PM.

$$f_3(x) = \sum_{j=1}^{m} (R_j \times X_j) \tag{15}$$

where

$f_3(x)$: Total economical revenue for placing VMs;
X_j: Binary variable that equals 1 if V_j is located for execution on any PM; otherwise, it is 0.

2.2.10 Objective Function 4: QoS Maximization

In the work presented in [8], the QoS maximization proposes to locate the maximum number of VMs with the highest level of priority associated to the SLA. This objective function is formulated in Eq. (16).

$$f_4(x) = \sum_{j=1}^{m} (\hat{C}^{SLA_j} \times SLA_j \times X_j) \tag{16}$$

where

$f_4(x)$: Total QoS figure for a given placement;
$\hat{C}$: Constant, large enough to prioritize services with a larger *SLA* over the ones with a lower *SLA*.

2.2.11 Objective Function 5: Network Load Balancing Optimization

The work presented in [8] calculates the total amount of network traffic going through a semi-duplex link l_a as:

$$T_{l_a} = \sum_{i=1}^{n} \sum_{i'=1}^{n} F_{aii'} \times \left(\sum_{j=1}^{m} \sum_{j'=1}^{m} P_{ji} \times P_{j'i'} \times D_{jj'} \times T_{jj'} \right) \tag{17}$$

where:

Tl_a: Total amount of traffic going through link l_a [Mbps];
$m_{ii'}$: Network path between H_i and H_i';
$F_{aii'}$: Binary variable that equals 1 if $l_a \in m_{ii'}$; otherwise, it is 0.

Inspired in the formulation presented in [15], the work presented in [8] calculates the Maximum Link Utilization (MLU) as:

$$MLU = \max_{\forall l_a \in L} \left(\frac{Tl_a}{Cl_a} \right) \tag{18}$$

where:

MLU: Maximum Link Utilization;
Cl_a: Channel capacity of link l_a [Mbps].

In [8], the load balancing optimization of the network is formulated as the minimization of the MLU, denoted as:

$$f_5(x) = MLU \tag{19}$$

2.3 Interactive Memetic Algorithm for MaVMP

A Memetic Algorithm (MA) could be understood as an Evolutionary Algorithm (EA) that in addition to the standard selection, crossover, and mutation operators of most Genetic Algorithms (GAs) includes a local optimization operator to obtain good solutions even at early generations of an EA [16]. In the VMP context, it is valuable to obtain good quality of solutions in short time. Consequently, a MA could be considered as a promising solution technique for VMP problems.

The work presented in [8] proposes an interactive MA for solving the VMP problem in a many-objective context, considering the proposed formulation presented in Sect. 2.2 to simultaneously optimize the five objective functions presented in the previous section. The proposed algorithm is extensible to consider as many objective functions as needed while only minor modifications may be needed if the number of objective functions changes.

It was shown in [5] that many-objective optimization using Multi-Objective Evolutionary Algorithms (MOEAs) is an active research area, having multiple challenges that need to be addressed. The interactive MA presented in this section is a viable way to solve a MaVMP problem, including desirable ranges of values for the objective functions in order to interactively control the possible huge number of feasible non-dominated solution. The interactive MA presented in Algorithm 1 is based on the MA proposed in [11] and works as described next:

At step 1, the algorithm verifies that the problem is solvable (considering only VMs with $SLA_j = s$) to continue its execution. If the problem could not be solved, the algorithm returns an appropriate error message. If the problem is solvable, the algorithm continues with step 2, generating a set of aleatory population P_0, whose candidate solutions are repaired at step 3 to ensure that P_0 contains feasible solutions only. Then, the algorithm tries to improve solutions at step 4 using a local search. With the obtained non-comparable solutions, the first Pareto set approximation P_{known} is

Algorithm 1: Interactive Memetic Algorithm

```
   Data: datacenter infrastructure (see Section 2.2.1)
   Result: Pareto set approximation $P_{known}$
 1 check if the problem has a solution
 2 initialize set of solutions $P_0$
 3 $P'_0$ = repair infeasible solutions of $P_0$
 4 $P''_0$ = apply local search to solutions of $P'_0$
 5 update set of solutions $P_{known}$ from $P''_0$
 6 $t = 0; P_t = P''_0$
 7 while stopping criterion is not met do
 8     $Q_t$ = selection of solutions from $P_t \cup P_{known}$
 9     $Q'_t$ = crossover and mutation of solutions of $Q_t$
10     $Q''_t$ = repair infeasible solutions of $Q'_t$
11     $Q'''_t$ = apply local search to solutions of $Q''_t$
12     update set of solutions $P_{known}$ from $Q'''_t$
13     increment t
14     if interaction is needed then
15         ask for decision-maker modification of ($L_z$ and $U_z$)
16     end
17     $P_t$ = non-dominated sorting from $P_t \cup Q'''_t$
18 end
19 return Pareto set approximation $P_{known}$
```

generated at step 5. After initialization at step 6, evolution begins (iterations between steps 7 and 18). The evolutionary process follows the same behavior: solutions are selected considering the union of P_{known} with the evolutionary set of solutions (or population) also known as P_t (step 8), crossover and mutation operators are applied as usual (step 9), and solutions are eventually repaired, as there may be infeasible solutions (step 10). Improvements of solutions may be generated at step 11 using local search in the evolutionary population P_t (local optimization operators). At step 12, the Pareto set approximation P_{known} is updated (if applicable); while at step 13 the generation counter is updated. At step 15, the decision-maker adjusts the lower and upper bounds if it is necessary, while at step 17 a new evolutionary population P_t is selected. The evolutionary process stops according to defined stopping criterion (as maximum number of generations), returning at the end the set of found non-dominated solutions P_{known} at step 19.

2.3.1 Population Initialization

Initially, a set of solutions P_0 is randomly generated. Each possible solution (or individual) x is represented as a chromosome $C = [C_1, C_2, \dots, C_m]$ (matrix P in Sect. 2.2.2). The possible values that can take each C_k for VMs with the highest value of SLA_j ($SLA_j = s$) are in the range [1, n]. On the other hand, for VMs V_j with $SLA_j < s$, C_k can take values in the range [0, n]. Within these ranges defined by the SLA_j of each V_j, the algorithm ensures that all VMs V_j with the highest level of priority will be located for execution on a PM H_i, while for a VM V_j with lower levels of priority SLA_j, there is always a probability larger than 0 that it may not be located for execution in any PM.

2.3.2 Infeasible Solution Reparation

With a random generation at the initialization phase (step 2 of Algorithm 1) and/or solutions generated by genetic operators (step 9 of Algorithm 1), infeasible solutions may appear, i.e., the resources required by the VMs allocated on particular PMs could exceed available resources, or at least one objective function may not meet adjustable constraints.

Repairing infeasible solutions (steps 3 and 10 of Algorithm 1) may be done in two stages: first, in the feasibility verification process, the population is classified in two classes: feasible or infeasible (Algorithm 2). Next, in the process of repairing infeasible solutions (Algorithm 3), the infeasible solutions are repaired in three ways: (1) migrating some VMs to an available hardware, (2) turning on some PMs and then migrating VMs to them, or (3) turning off some VMs with $SLA_j < s$.

2.3.3 Local Search

With a population composed by feasible solutions only, a local search is performed (steps 4 and 11 of Algorithm 1) improving solutions found until then in the evolutionary population. The local search pseudocode is presented in Algorithm 4.

For each individual in the evolutionary population P_t, the interactive MA proposed in [8] attempts to optimize a solution with a local search (step 2 of Algorithm 4).

Algorithm 2: Feasibility Verification

```
Data: set of solutions $P_t$
Result: set of feasible solutions $P'_t$
1  while there are solutions not verified do
2      feasible = true ; i = 1
3      while $i \leq n$ and feasible = true do
4          if solution does not satisfy constraints (3-5) then
5              feasible = false ; break
6          else
7              increment i
8          end
9      end
10     if feasible = false then
11         call Algorithm 3 (repair solution)
12     end
13 end
14 return set of feasible solutions $P'_t$
```

Algorithm 3: Infeasible Solutions Reparation

```
   Data: infeasible solution
   Result: feasible solution
1  feasible = false ; j = 1
2  while $j \leq m$ and feasible = false do
3      if it is possible then
4          migrate $V_j$ to $H'_i$ ($i' \neq i$)
5      else
6          if $SLA_j \neq s$ then
7              turn off $V_j$ on $H_i$
8          else
9              replace solution with another solution from $P_{known}$
10         end
11     end
12 end
13 return feasible solution
```

Algorithm 4: Local Search

```
   Data: set of feasible solutions $P'_t$
   Result: set of feasible optimized solutions $P''_t$
1  probability = random number between 0 and 1
2  while there are solutions not verified do
3      if probability $< 0.5$ then
4          Try to turn off all the possible $H_i$ by migrating all the $V_j$ assigned to $H'_i$ with available
           resources ($i' \neq i$) and then try to turn on all the possible $V_j$ (using $SLA_j$ priority order)
           assigning them to a $H_i$ with available resources
5      else
6          Try to turn on all the possible $V_j$ (using $SLA_j$ priority order) assigning them to a $H_i$
           with available resources and then try to turn off all the possible $H_i$ by migrating all
           the $V_j$ assigned to $H'_i$ with available resources ($i' \neq i$)
7      end
8  end
9  return set of feasible optimized solutions $P''_t$
```

For this, with probability $\frac{1}{2}$, the algorithm tries maximizing the number of allocated VMs with higher level of priority, locating all possible VMs that were not located so far, increasing $f_3(x)$ (total economical revenue) and $f_4(x)$ (total quality of service) (steps 3 to 5 of Algorithm 4). Additionally, also with probability $\frac{1}{2}$, the algorithm tries minimizing the number of PMs turned on, directly reducing $f_1(x)$ (total power consumption) (steps 6 to 8 of Algorithm 4). With the proposed probabilistic local search method, a balanced exploitation of objective functions (economical revenue, quality of service and power consumption) is achieved, as experimentally verified with results presented in next section.

2.3.4 Fitness Function

The fitness function considered in the proposed algorithm is the one proposed in [17]. This fitness defines a non-domination rank in which a value equal to its Pareto dominance level (1 is the highest level of dominance) is assigned to each individual of the population. Between two solutions with different non-domination rank, the individual with lower value (higher level of dominance) is considered better.

To compare solutions with the same non-domination rank, a crowding distance is used. Basically, a crowding distance finds the Euclidean distance (properly normalized when the objectives have different measure units) between each pair of solutions, based on the q objectives, in a hyper-dimensional space [17]. The solution with higher crowding distance is considered better.

2.3.5 Variation Operators

The proposed interactive MA considers a Binary Tournament approach for selecting individuals for crossover and mutation [18]. The crossover operator used in the presented work is the single point cross-cut [18]. The selected individuals in the ascending population are replaced by descendants individuals.

The work presented in [8] uses a mutation method in which each gene is mutated with a probability $\frac{1}{m}$, where m represents the number of VMs. This method offers the possibility of full uniform gene mutation, with a very low probability (but larger than zero), which is beneficial to the exploration of the search space, reducing the probability of stagnation in a local optimum. The population evolution in the proposed interactive MA is based on the population evolution proposed in [17]. A population P_{t+1} is formed from the union of the best known population P_t and offspring population Q_t, applying non-domination rank and crowding distance.

2.3.6 Many-Objective Considerations

Given that the number of non-dominated solutions may rapidly increase, an interactive approach is recommended. That way, a decision-maker can introduce new constraints or adjust existing ones, while the execution continues learning about the shape of the Pareto front in the process. For simplicity, the present work considers lower and upper bounds associated to each objective function in order to help the decision-maker to reduce interactively the potential huge number of solutions in the Pareto set approximation P_{known}, while observing the evolution of its corresponding Pareto front PF_{known} to the region of his preference.

Table 1 Types of PMs considered in experiments. For notation see Eq. (4)

PM type	Hcpu [ECU]	Hram [GB]	Hhdd [GB]	pmax [W]
h1.small	4	16	150	440
h1.medium	180	512	10000	1000
h1.large	350	1024	10000	1300

2.4 *Experimental Results*

This section summarizes experimental results obtained by the proposed algorithm [8] in carefully designed experiments to validate its effectiveness considering challenges associated to the resolution of a MaVMP problem previously introduced.

First, Experiment 1 performed a quality evaluation of the solutions obtained by the proposed algorithm against optimal solutions obtained with an exhaustive search algorithm in two different scenarios. Next, Experiment 2 performed an evaluation using lower and upper bounds associated to each objective function $f(z)$ ($L_z \leq f_z(x) \leq U_z$) to be able to converge to a manageable number of solutions in the Pareto set approximation. Finally, Experiment 3 evaluates the proposed algorithm solving instances of the problem with large numbers of PMs and VMs. For simplicity, all experiments considered a datacenter infrastructure composed by PMs interconnected in a simple two-tier network topology.

2.4.1 Experimental Environment

Different problem instances were proposed for the above-mentioned experiments considering both homogeneous and heterogeneous hardware configurations of PMs, as well as homogeneous and heterogeneous VMs instance types offered by Amazon Elastic Compute Cloud (EC2).[2] A detailed description of the hardware configuration of the PMs and VMs instance types considered for the experiments is presented in Tables 1 and 2 respectively. Additionally, a general description of the considered problem instances including its decision space size is presented in Table 3.

The complete set of datacenter infrastructure input files used for the experiments with the corresponding experimental results are available online.[3]

Algorithms considered in the experiments were implemented using ANSI C programming language (gcc) and the source code is available online[3]. All the presented experiments were executed on a CentOS 6.5 Linux Operating System, with an Intel(R) Xeon(R) CPU E5530 at 2.40 GHz processor and 8 GB of RAM.

[2]http://aws.amazon.com/ec2/instance-types.

[3]https://github.com/flopezpires/iMaVMP.

Table 2 Instance types of VMs considered in experiments. For notation see Eq. (5)

Instance type	Vcpu [ECU]	Vram [GB]	Vhdd [GB]	R [$]
t2.micro	1	1	0	9
t2.small	1	2	0	18
t2.medium	2	4	0	37
m3.medium	1	4	4	50
m3.large	2	8	32	100
m3. × large	4	15	80	201
m3.2 × large	8	30	160	403
c3.large	2	4	32	75
c3. × large	4	8	80	151
c3.2 × large	8	15	160	302
c3.4 × large	16	30	320	604
c3.8 × large	32	60	640	1209
r3.large	2	15	32	126
r3. × large	4	30	80	252
r3.2 × large	8	61	160	504
r3.4 × large	16	122	0	320
r3.8 × large	32	244	0	320

Table 3 Problem instances considered in experiments, all with 50% of VMs with SLA $s = 2$

Experiment	Input	# PMs	# VMs	PMs and VMs	$(n+1)^m$
1	3 × 5.vmp	3	5	Homogeneous	1024
1	4 × 8.vmp	4	8	Heterogeneous	390625
2	12 × 50.vmp	12	50	Heterogeneous	$\sim 5 \times 10^{55}$
3	100 × 1000.vmp	100	1000	Heterogeneous	$\sim 2 \times 10^{2004}$

2.4.2 Experiment 1: Quality of Solutions

To compare the results obtained by the proposed interactive MA and to validate its proper operation, an Exhaustive Search Algorithm (ESA) was also implemented for finding all $(n + 1)^m$ possible solutions of a given instance of the VMP problem, when this alternative is computationally possible for the authors. These results were compared to the results obtained by the proposed interactive MA.

Considering that this particular experiment aims to validate the good level of exploration in the set of feasible solutions X_f, the local search of the algorithm was disabled, strengthening its capability of exploration rather than the rapid convergence to good solutions even in early generations of the population.

Table 4 Summary of results obtained by the proposed algorithm in Experiment 1

Input	P^* size	P_{known} size	Execution time (ESA) (s)	Execution time (MA) (s)
3×5.vmp	51	51	∼1	∼12
4×8.vmp	30	30	∼720	∼29

For each problem instance considered in this experiment (see Table 3), one run of the exhaustive search algorithm was completed, obtaining the optimal Pareto set P^* and its corresponding Pareto front PF^*.

Furthermore, 10 runs of the proposed algorithm were completed, after evolving populations composed by 100 individuals for 100 generations at each run. The results obtained by the proposed algorithm for each run were combined to obtain the Pareto set approximation P_{known} and its corresponding Pareto front PF_{known}.

For both considered problem instances, the proposed algorithm obtained 100% of the solutions of P^* and its corresponding PF^*. Additionally, the proposed algorithm performed well in execution time against the ESA, even obtaining the same optimal results in less execution time for the 4×8.vmp scenario. A summary of the number of elements in the corresponding Pareto sets obtained and the execution time of both algorithms is presented in Table 4.

2.4.3 Experiment 2: Interactive Bounds

For the problem instance considered in this experiment (12×50.vmp), one run of the proposed algorithm was completed, after evolving populations of 100 individuals for 300 generations. The number of generations was incremented for this experiment from 100 to 300, taking into account the large number of possible solutions for the particular considered problem (see Table 3). An interactive adjustment of the lower or upper bounds associated to each objective function was performed after every 100 generations in order to converge to a treatable number of solutions. It is important to remark that the interactive adjustment used in this experiment is only one of several possible ones. As an example, we may consider: (1) automatically adjusting a % of the lower bounds associated to maximization objective functions when the Pareto front has a defined number of elements or (2) manually adjusting upper bounds associated to minimization objective functions until the Pareto front does not have more than 20 elements, just to cite a pair of alternatives.

The Pareto front approximation PF_{known} represents the complete set of Pareto solutions considering unrestricted bounds ($L_z = -\infty$ and $U_z = \infty$). On the other hand, Pareto front approximation $PF_{reduced}$ represents the reduced set of Pareto solutions obtained by interactively adjusting bounds L_z and U_z. In the first 100 generations, the proposed algorithm obtained 251 solutions with unrestricted bounds. A decision-maker evaluated the bounds associated to $f_1(x)$ (power consumption) and adjusted the upper bound U_1 to $U_1' = 9000$ [W], selecting only 35 out of the 251 solu-

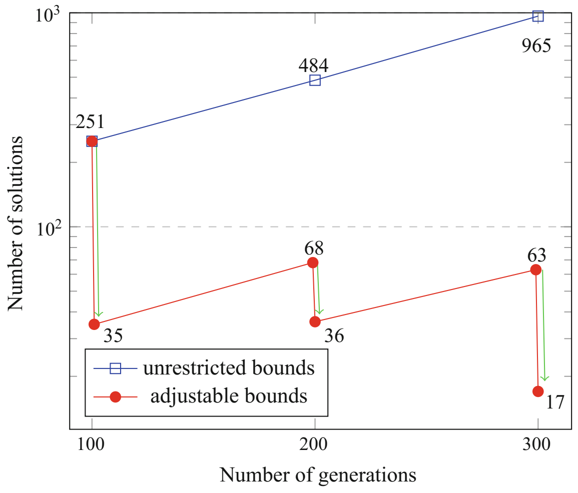

Fig. 2 Summary of results obtained in Experiment 2 using adjusted lower and upper bounds

tions (not considering 216 otherwise feasible solutions) for the $PF_{reduced}$ as shown in Fig. 2. After 200 generations, the algorithm obtained a total of 484 solutions with unrestricted bounds. Considering instead $U_1' = 9000$ [W], the algorithm only found 68 solutions. The decision-maker evaluated the bounds associated to $f_2(x)$ (network traffic) and adjusted the upper bound U_2 to $U_2' = 115$ [Mbps], selecting only 36 out of the 68 solutions (not considering 32 otherwise feasible solutions) for the $PF_{reduced}$. Finally, after 300 generations, the algorithm obtained a total of 965 solutions with unrestricted bounds. Considering $U_1' = 9000$ [W] and $U_2' = 115$ [Mbps], the algorithm found 63 solutions. The decision-maker evaluated the bounds associated to $f_3(x)$ (economical revenue) and adjusted the lower bound L_3 to $L_3' = 13500$ [\$], selecting only 17 out of the 63 solutions (not considering 46 feasible solutions) for the final $PF_{reduced}$ as shown in Fig. 2. Clearly, at the end of the iterative process, the decision-maker found 17 solutions according to his preferences instead of the unmanageable number of 965 candidate solutions.

2.4.4 Experiment 3: Algorithm Scalability

It should be noted that increasing the number of PMs and VMs in a VMP problem could result in extremely large decision spaces, considering all $(n + 1)^m$ possible solutions (see Table 3). Consequently, algorithms designed for the resolution of VMP problems should be able to effectively solve VMP problem instances composed by large numbers of VMs and PMs in a reasonable time.

For the problem instance considered in this experiment (100×1000.vmp), one run of the proposed algorithm was completed, after evolving populations composed by 100, 200, and 300 individuals for 500 generations. For this particular experiment, the Pareto front approximation PF_{known} represents the complete set of Pareto solu-

Table 5 Summary of results obtained by the proposed algorithm in Experiment 3

Input	# of individuals	P_{known} size	# of generations
100 × 1000.vmp	100	397	100
100 × 1000.vmp	200	399	100
100 × 1000.vmp	300	509	100
100 × 1000.vmp	100	769	200
100 × 1000.vmp	200	811	200
100 × 1000.vmp	300	1087	200
100 × 1000.vmp	100	1103	300
100 × 1000.vmp	200	1329	300
100 × 1000.vmp	300	1641	300
100 × 1000.vmp	100	1434	400
100 × 1000.vmp	200	1791	400
100 × 1000.vmp	300	2178	400
100 × 1000.vmp	100	1742	500
100 × 1000.vmp	200	2192	500
100 × 1000.vmp	300	2719	500

tions considering unrestricted bounds for each $f_z(x)$ ($L_z = -\infty$ and $U_z = \infty$) in order to experimentally demonstrate that large instances of the formulated MaVMP problem could result in unmanageable number of solutions. A summary of the results obtained by the proposed algorithm is presented in Table 5. The obtained results prove the capabilities of the proposed algorithm to effectively solve instances of the proposed MaVMP problem with large numbers of PMs and VMs, as considered in real-world scenarios. Additionally, it could be observed that increasing the number of individuals on populations or the number of generations, the algorithm obtained larger numbers of non-dominated solutions with unrestricted bounds. Considering that the proposed algorithm could find more non-dominated solutions than the obtained in this particular experiment if more computational resources for calculation are considered (or increasing the number of individuals or the number of generations), it could be noted the importance of including additional methods to the Pareto dominance relation (e.g., adjustable bounds) for the selection of a manageable subset of P_{known} in MaVMP problems for initial placement of VMs.

3 Many-Objective VMP with Reconfiguration of VMs

Once an initial placement of VMs has been performed (as presented in Sect. 2), a virtualized datacenter could be reconfigured through live migration in order to maintain efficiency in operations, considering that the set of requested VMs changes over time (i.e., the set V presented in Sect. 2.2.1 is a function of time). Studying this particular semi-dynamic formulation of a MaVMP with reconfiguration of VMs represents a first approximation to dynamic formulations in real-world cloud computing environments, where several dynamic parameters should also be considered.

According to [2, 9], the optimization of the power consumption is the most studied objective function in VMP literature [13, 19]. Furthermore, network traffic [20] and economical revenue [21, 22] are also very much studied as objective functions for the VMP problem. For a VMP problem formulation with reconfiguration of VMs, two additional objective functions associated to migration of VMs represent challenges for CSPs: minimizing the total number of VM migrations [23] as well as the total network traffic overhead due to VM migrations [24].

Considering the large number of existing objective functions for the VMP problem identified in [2, 9], López-Pires and Barán have proposed in [8, 25] a many-objective optimization framework in order to consider as many objective functions as needed when solving a MaVMP problem for initial placement of VMs in virtualized datacenters (see Sect. 2). To the best of the authors' knowledge, there was no published work presenting a formulation of a MaVMP problem with reconfiguration of VMs. Consequently, this section extends the formulations presented in Sect. 2 [8, 25] presenting the first MaVMP with reconfiguration of VMs, considering this time the simultaneous optimization of the following five objective functions: (1) power consumption, (2) inter-VM network traffic, (3) economical revenue, (4) number of VM migrations, and (5) network traffic overhead for VM migrations [7].

To solve the formulated MaVMP problem, the interactive MA presented in Sect. 2.3 was extended to consider particular challenges associated to the resolution of a MaVMP problem with reconfigurations of VMs, as next introduced.

Several challenges need to be addressed for MaVMP formulations with reconfiguration of VMs. In Pareto-based algorithms, the Pareto set approximation can include a large number of non-dominated solutions. Selecting one of the non-dominated solutions can be considered a problem for a MaVMP problem. In consequence, the work presented in [7] evaluates the following five selection strategies: (1) random, (2) preferred solution, (3) minimum distance to origin, (4) lexicographic order (provider preference), and (5) lexicographic order (service preference) to identify convenient strategies for automatic selection of a non-dominated solution.

3.1 Problem Formulation

This chapter presents the formulation of a MaVMP with reconfiguration of VMs [7], considering this time the simultaneous optimization of the following five objective functions: (1) power consumption, (2) inter-VM network traffic, (3) economical revenue, (4) number of VM migrations and (5) network traffic overhead for VM migrations. Formally, the presented offline (semi-dynamic) MaVMP problem with reconfiguration of VMs can be enunciated as:

Given the available PMs and their specifications, the requested VMs and their specifications, the network traffic between VMs and the current placement of the VMs, it is sought a new placement of the set of VMs in the set of PMs, satisfying the constraints of the problem while simultaneously optimizing all defined objective functions (as power consumption, inter-VM network traffic, economical revenue, number of VM migrations and network traffic overhead for VM migration), in a pure many-objective context, before selecting a specific solution for a given time instant t.

3.1.1 Input Data

The set of available PMs is represented as a matrix $H \in \mathbb{R}^{n\times 4}$, previously introduced in Sect. 2.2.1 (see Eq. (4)). Accordingly, the set of VMs at instant t is now represented as a matrix $V(t) \in \mathbb{R}^{m\times 5}$:

$$V(t) = \begin{bmatrix} Vcpu_1 & Vram_1 & Vhdd_1 & SLA_1 & R_1 \\ \dots & \dots & \dots & \dots & \dots \\ Vcpu_{m(t)} & Vram_{m(t)} & Vhdd_{m(t)} & SLA_{m(t)} & R_{m(t)} \end{bmatrix} \tag{20}$$

Each V_j represents the required processing resources of CPU [ECU], RAM memory [GB], storage [GB], SLA, and revenue [\$]:

$$V_j = [Vcpu_j, Vram_j, Vhdd_j, SLA_j, R_j], \quad \forall j \in \{1, \dots, m(t)\} \tag{21}$$

where

$Vcpu_j$: Processing requirements of V_j;
$Vram_j$: Memory requirements of V_j;
$Vhdd_j$: Storage requirements of V_j;
R_j: Economical revenue for placing V_j;
SLA_j: Service Level Agreement SLA_j of a V_j, where $SLA_j \in \{0, 1, \dots, s\}$ being s the highest priority level;
$m(t)$: Number of VMs at instant t, then $m(t) \in \{1, \dots, m_{max}\}$;
m_{max}: Maximum number of VMs.

Once a V_j is powered off by the tenant, its resources are released, so the physical resources can be reused. For simplicity, the index j is not considered to be reused; therefore, for the work presented in [7] V_j is not a function of time.

The traffic between VMs at instant t is represented as a matrix $T(t) \in \mathbb{R}^{m(t)\times m(t)}$:

$$T(t) = \begin{bmatrix} T_{1,1}(t) & \dots & T_{1,m(t)}(t) \\ \dots & \dots & \dots \\ T_{m(t),1}(t) & \dots & T_{m(t),m(t)}(t) \end{bmatrix} \tag{22}$$

In Eq. (22), $T_{jk}(t)$ represents the average communication rate in [Mbps], between VM V_j and VM V_k at instant t. Note that we can consider $T_{jj}(t) = 0$.

The placement at instant t is represented as a matrix $P(t) \in \mathbb{R}^{m(t)\times n}$:

$$P(t) = \begin{bmatrix} P_{1,1}(t) & \dots & P_{1,n}(t) \\ \dots & \dots & \dots \\ P_{m(t),1}(t) & \dots & P_{m(t),n}(t) \end{bmatrix} \tag{23}$$

where:

$P_{ji}(t) \in \{0, 1\}$ indicates if V_j is located ($P_{ji} = 1$) or not ($P_{ji} = 0$) for execution on a PM H_i (i.e., $P_{ji}(t) : V_j \rightarrow H_i$) at instant t.

3.1.2 Output Data

A solution of the problem at each instant is a new VM placement $P(t + 1)$. In order to accommodate a new placement, a series of management actions (MAc) (i.e., VM migrations, creation or destruction) must be performed. These are presented by the following output data: (1) the new VM placement and (2) the list of required management actions.

The new placement at instant $(t + 1)$ is represented as a matrix $P(t + 1)$ of dimension $m(t + 1) \times n$:

$$P(t+1) = \begin{bmatrix} P_{1,1} & \dots & P_{1,n} \\ \dots & \dots & \dots \\ P_{m(t+1),1} & \dots & P_{m(t+1),n} \end{bmatrix} \tag{24}$$

where $P_{ji}(t+1) \in \{0, 1\}$ indicates if V_j is located ($P_{ji}(t+1) = 1$) or not ($P_{ji}(t+1) = 0$) for execution on a PM H_i at instant t (i.e., $P_{ji}(t + 1) : V_j \rightarrow H_i$).

The set of necessary management actions in order to evolve from $P(t)$ to $P(t+1)$ is represented by:

$$MAc_{t \rightarrow t+1} = \left[MAc(V_1), \dots, MAc(V_{m(t+1)}) \right] \tag{25}$$

where $MAc(V_j) \in \{0, 1, 2, 3\}$ which represents the management actions that a hypervisor must execute in order to accommodate $P(t + 1)$ corresponding to V_j.
Values returned by the $MAc(V_j)$ function should be interpreted as follows:

$MAc(V_j) = 0$: no management action is necessary, i.e., $P_{ji}(t + 1) = P_{ji}(t), \forall i$;
$MAc(V_j) = 1$: a new VM V_j is placed on a PM H_i, i.e., $P_{ji}(t + 1) = 1$;
$MAc(V_j) = 2$: an existing VM V_j is migrated from $H_{i'}$ to another H_i, i.e., $P_{ji'}(t) = 1$ and $P_{ji}(t + 1) = 1$;
$MAc(V_j) = 3$: a VM V_j is shutdown, i.e., $P_{ji}(t) = 1$ but $P_{ji}(t + 1) = 0$.

3.1.3 Constraint 1: Unique Placement of VMs

A VM V_j should be located to run on a single PM H_i or alternatively, it could be not located in any PM if the associated SLA_j is not the highest level of priority (in [7] $s = 2$). Consequently, this constraint is expressed as:

$$\sum_{i=1}^{n} P_{ji}(t) \leq 1 \quad \forall j \in \{1, ..., m(t)\}, \quad \forall t \tag{26}$$

where

$P_{ji}(t)$: Binary variable equals 1 if V_j is located to run on H_i at instant t; otherwise, it is 0.

3.1.4 Constraint 2: Assure SLA Provisioning

A VM V_j with the highest level of SLA ($s = 2$) must necessarily be located to run on a PM H_i. Consequently, this constraint is expressed as:

$$\begin{gathered}\sum_{i=1}^{n} P_{ji}(t) = 1 \quad \forall j \text{ such that } SLA_j = s \\ \forall t \text{ where } V_j \text{ should be active.}\end{gathered} \tag{27}$$

It should be remarked that different levels of SLA can be considered, as presented in [8].

3.1.5 Constraints 3–5: Physical Resources Capacities of PMs

A PM H_i must have sufficient available resources to meet the requirements of all VMs V_j that are located to run on H_i at instant t. In the work presented in [7], the overbooking of resources [26] is not considered; consequently, the set of constraints can be mathematically formulated as:

$$\sum_{j=1}^{m(t)} Vcpu_j \times P_{ji}(t) \leq Hcpu_i \tag{28}$$

$$\sum_{j=1}^{m(t)} Vram_j \times P_{ji}(t) \leq Hram_i \tag{29}$$

$$\sum_{j=1}^{m(t)} Vhdd_j \times P_{ji}(t) \leq Hhdd_i \tag{30}$$

$\forall i \in \{1, ..., n\}$, i.e., for all physical machines H_i and $\forall t$.

Next section presents five objective functions that are simultaneously optimized in the presented MaVMP formulation with reconfiguration of VMs. These objective functions are mathematically formulated as follows.

3.1.6 Objective Function 1: Power Consumption Minimization

Based on Eq. (13), the power consumption at each discrete time t can be represented by the sum of the power consumption of each PM H_i:

$$f_1(x,t) = \sum_{i=1}^{n}((pmax_i - pmin_i) \times Ucpu_i(t) + pmin_i) \times Y_i(t) \tag{31}$$

where

$f_1(x,t)$: Total power consumption of the PMs at each discrete time t;
$Ucpu_i(t)$: Utilization ratio of processing resources used by H_i at instant t;
$Y_i(t)$: Binary variable equals 1 if H_i is turned on; otherwise, it is 0.

3.1.7 Objective Function 2: Inter-VM Network Traffic Minimization

A very much studied approach for inter-VM network traffic minimization is the placement of VMs with high communication rate in the same PM (or at least in the same rack) to avoid the utilization of network resources (or at least core network equipment).

The minimization of network traffic among VMs, by maximizing locality, was proposed in [14]. Based on Eq. (14), Eq. (32) represents the sum of average network traffic between VM V_j and VM V_k when located on different PMs.

$$f_2(x,t) = \sum_{j=1}^{m(t)}\sum_{k=1}^{m(t)}(T_{jk}(t) \times D_{jk}(t)) \tag{32}$$

where

$f_2(x,t)$: Total inter-VM network traffic at each discrete time t;
$D_{jk}(t)$: Binary variable that equals 1 if V_j and V_k are located in different PMs at instant t; otherwise, it is 0.

The traffic between two VMs V_j and V_k located on the same PM H_i does not contribute to increase the total network traffic given by Eq. (32); therefore, $D_{jk}(t) = 0$ if $P_{ji}(t) = P_{ki}(t) = 1$.

3.1.8 Objective Function 3: Economical Revenue Maximization

Based on Eq. (15), Eq. (33) is presented to estimate the total economical revenue that a datacenter receives for meeting the requirements of its customers, represented by the sum of the economical revenue obtainable by each VM V_j that is effectively located for execution on any PM at instant t.

$$f_3(x,t) = \sum_{j=1}^{m(t)} (R_j \times X_j(t)) \tag{33}$$

where

$f_3(x,t)$: Total economical revenue for placing VMs at each discrete time t;
$X_j(t)$: Binary variable that equals 1 if V_j is located for execution on any PM at instant t; otherwise, it is 0.

3.1.9 Objective Function 4: Number of VM Migrations Minimization

Performance degradation may occur when migrating VMs from one PM to another [24]. Logically, it is desirable that the number of migrated VMs is kept to a minimum for better quality of service (QoS). Therefore, Eq. (34) represents the number of VM migrations at time instant t:

$$f_4(x,t) = \sum_{j=1}^{m(t)} Z_j(t) \tag{34}$$

where

$f_4(x,t)$: Number of VM migrations at instant t;
$Z_j(t)$: Binary variable that equals 1 if $MA(V_j) = 2$, i.e., V_j is migrated, see (25); otherwise, it is 0 (V_j is not migrated).

3.1.10 Objective Function 5: Network Traffic Overhead for VM Migrations Minimization

As explained in [24], the overhead of VM migrations on network resources is proportional to the memory size of the migrated VM. In the work presented in [7], (35) is proposed to minimize the amount of RAM memory that must be copied between PMs at instant t.

$$f_5(x,t) = \sum_{j=1}^{m(t)} Vram_j \times Z_j(t) \tag{35}$$

where

$f_5(x, t)$: Network traffic overhead for VM migrations at instant t;

It should be mentioned that there are other possible modeling approaches to estimate the migration overhead, as presented in [27].

Finally, it should be noted that the main difference between the above-described objective functions for the MaVMP with reconfiguration of VMs (see Eqs. (31)–(35)) with the ones previously presented in Sect. 2 for the MaVMP for initial placement of VMs (see Eqs. (13)–(18)) is that Eqs. (31)–(35) are calculated at each discrete time t.

3.2 Extended Memetic Algorithm for MaVMP

The work presented in [7] extends the interactive MA proposed in [8, 25] for solving the MaVMP problem with reconfiguration of VMs, as the one formulated in Sect. 3.1. The proposed algorithm simultaneously optimizes the five objective functions presented in the previous sections.

Many-objective optimization using Multi-Objective Evolutionary Algorithms (MOEAs) is an active research area, with multiple challenges that need to be addressed regarding scalability analysis, solutions visualization, algorithm design, and experimental algorithm evaluation as shown in [5]. At each time instant, the set of feasible placement solutions can be composed by a large number of non-dominated solutions. Therefore, the algorithm proposed in [7] automatically selects one of the possible placements after each time instant according to one of the considered selection strategies (see Sect. 3.3). The proposed algorithm is based on the one proposed in Sect. 2 [8] and it works as follows (see Algorithm 5):

The algorithm iterates over each set of requested VMs received at each instant t. At step 3, the algorithm verifies if the problem has at least one solution to continue with next steps. If there is no possible solution to the problem, the algorithm returns an appropriate error message. If the problem has at least one solution, the algorithm proceeds to step 4 in order to determine the current placement. After the first iteration, the current placement is the one selected from the previous iteration.

At step 5, a set P_0 of candidates is randomly generated. These candidates are repaired at step 6 to ensure that P_0 contains only feasible solutions. Then, the algorithm tries to improve candidates at step 7 using local search. With the obtained non-dominated solutions, the first set P_{known} (Pareto set approximation) is generated at step 8. After initialization in step 9, evolution begins (between steps 10 and 18).

The evolutionary process basically follows the same behavior: solutions are selected from the union of P_{known} with the evolutionary set of solutions (or population) also known as P_u (step 11), crossover and mutation operators are applied as usual (step 12), and eventually solutions are repaired, as there may be infeasible solutions (step 13). Improvements of solutions of the evolutionary population P_u may be generated at step 14 using local search (local optimization operators).

At step 15, the Pareto set approximation P_{known} is updated (if applicable); while at step 16 the generation (or iteration) counter is updated. At step 17 a new evolutionary population P_u is selected. The evolutionary process is repeated until the algorithm meets a stopping criterion (such as a maximum number of generations), returning one solution $P_{selected}$ from the set of non-dominated solutions P_{known} in step 20, using one of the selection strategies presented in Sect. 3.3.

It should be mentioned that the main phases of Algorithm 5 are based on the ones previously presented in Sect. 2 (see Sects. 2.3.1–2.3.5 for details).

3.3 Solution Selection Strategies

Several challenges need to be addressed for a MaVMP problem with reconfiguration of VMs. In Pareto-based algorithms, the Pareto set approximation can include a large number of non-dominated solutions; therefore, selecting one of the non-dominated solutions (step 19 of Algorithm 5) can be considered as a new difficulty for MaVMP problems with reconfiguration of VMs.

The work presented in [7] performed an experimental evaluation of the following five selection strategies: (1) random, (2) preferred solution, (3) minimum distance

Algorithm 5: Extended Memetic Algorithm

```
   Data: datacenter infrastructure (see Section 3.1.1) and solution selection strategy parameter
   Result: solution $P_{selected}$ for instant $t$
1  $t = 0$
2  while there are VM requests to process do
3      check if the problem has a solution
4      $P_{previous} = P_{selected}$
5      initialize set of solutions $P_0$
6      $P'_0$ = repair infeasible solutions of $P_0$
7      $P''_0$ = apply local search to solutions of $P'_0$
8      update set of solutions $P_{known}$ from $P''_0$
9      $u = 0$; $P_u = P''_0$
10     while is not stopping criterion do
11         $Q_u$ = selection of solutions from $P_u \cup P_{known}$
12         $Q'_u$ = crossover and mutation of solutions of $Q_u$
13         $Q''_u$ = repair infeasible solutions of $Q'_u$
14         $Q'''_u$ = apply local search to solutions of $Q''_u$
15         update set of solutions $P_{known}$ from $Q'''_u$
16         increment number of generations $u$
17         $P_u$ = non-dominated sorting from $P_u \cup Q'''_u$
18     end
19     $P_{selected}$ = selected solution (selection strategy parameter)
20     return $P_{selected}$
21     increment instant $t$; reset $P_{known}$
22 end
```

to origin, (4) lexicographic order (provider preference), and (5) lexicographic order (service preference), as next explained.

3.3.1 Random ($S1$)

Considering that the Pareto set approximation is composed by non-dominated solutions, randomly selecting one of the solutions could be an acceptable strategy.

3.3.2 Preferred Solution ($S2$)

A solution is defined as preferred to another non-comparable solution when it is better in more objective functions [28]. When several solutions can be considered as preferred ones (there is a tie), only one of these solutions is randomly selected.

3.3.3 Minimum Distance to Origin ($S3$)

The solution with the minimum Euclidean distance to the origin is selected, considering all normalized objective functions in a minimization context. For this purpose, $f_3(x, t)$ is redefined as the difference between the maximum possible revenue at instant t and the attainable revenue of each possible solution. When several solutions have equal Euclidean distance, only one of these solutions is randomly selected.

3.3.4 Lexicographic Order

Each objective function is given in an order of evaluation, similar to the ordering of letters in a dictionary. The objective functions can be arranged in several ways in order of priority. The work presented in [7] proposes two different lexicographic orders, representing the possible preferences associated to providers (provider preference) and quality of service (service preference). Logically, different orders of priority criteria may be considered depending on each specific context.

- **Provider preference order** ($S4$)**:** The priority order is: (1) economical revenue, (2) power consumption, (3) inter-VM network traffic, (4) number of VM migrations and (5) network traffic overhead for VM migration.
- **Service preference order** ($S5$)**:** The priority order is: (1) number of VM migrations, (2) network traffic overhead for VM, (3) inter-VM network traffic, (4) power consumption, and (5) economical revenue.

The work presented in [7] evaluates the above-mentioned selection strategies, where several experiments were performed. The following subsections summarize the experimental results.

Table 6 Hardware configuration of PM types considered in Experiment 4

PM type	Hardware configuration				Number of PMs	
	Hcpu [ECU]	Hram [GB]	Hhdd [GB]	pmax [W]	10 × 100.vmp	100 × 1000.vmp
h1.small	180	512	10,000	1,000	3	30
h1.medium	260	512	10,000	1,350	3	30
h1.large	350	1,024	10,000	1,800	3	30
h2.large	400	1,024	10,000	2,000	1	10
Total PMs					10	100

Table 7 Instance types of VMs considered in experiments. For notation see Eq. (20)

Instance type	Vcpu [ECU]	Vram [GB]	Vhdd [GB]	R [$]
t2.micro	1	1	0	9
t2.small	1	2	0	18
t2.medium	2	4	0	37
m3.medium	1	4	4	50
m3.large	2	8	32	100
m3. × large	4	15	80	201
m3.2 × large	8	30	160	403
c3.large	2	4	32	75
c3. × large	4	8	80	151
c3.2 × large	8	15	160	302
c3.4 × large	16	30	320	604
c3.8 × large	32	60	640	1209
r3.large	2	15	32	126
r3. × large	4	30	80	252
r3.2 × large	8	61	160	504
r3.4 × large	16	122	0	320
r3.8 × large	32	244	0	320

3.4 Experimental Environment

The Extended Memetic Algorithm presented in Sect. 3.2 was implemented using the ANSI C programming language (gcc). The source code is available online.[4]

The experimental scenarios included heterogeneous PMs with hardware configurations described in Table 6. Considered VMs were based on real instance types offered by Amazon Elastic Compute Cloud (EC2) [29] as presented in Table 7.

[4]https://github.com/dihara/MaVMP.

The experiments were performed considering two different experimental scenarios (a small and a medium sized datacenter infrastructure) simulating a theoretical day (i.e., 24 h) in a datacenter where VMs requests are received and processed hourly. In these experiments, the following configurations were considered:

- **10 × 100.vmp**: Problem instance with 10 PMs initially running 100 VMs.
- **100 × 1000.vmp**: Problem instance with 100 PMs initially running 1,000 VMs.

For simplicity, in what follows, the traffic between VMs was considered as constant, i.e., $T_{i,j}(t) = T_{i,j}$. The initial load for Experiment 4 represents 28% of CPU resources while in Experiment 5, it is 33% of CPU resources (see Table 9).

Experiments for each selection strategy were repeated 10 times, given the probabilistic nature of the Extended Memetic Algorithm. Results are analyzed in Sect. 3.5. The average number of non-dominated solutions found with each selection strategy is shown in Table 8. It can be seen that in both experiments, a similar average number of solutions and standard deviation were observed for all strategies.

Table 8 Number of non-dominated solutions per selection strategy

Selection strategy	10 × 100.vmp		100 × 1000.vmp	
	Average	Standard Dev.	Average	Standard Dev.
Random	25.2	8.6	36.2	11.3
Preferred solution	24.0	8.6	37.7	9.2
Distance to origin	21.4	8.7	30.8	10.1
Provider preference	20.8	9.9	24.0	8.7
Service preference	34.5	9.5	38.9	9.1

Table 9 Details of Experiment 4

Parameters	10 × 100.vmp	100 × 1000.vmp
# PM	10	100
Available CPU	2,770	27,700
Initial # VM	100	1,000
VMs with SLA 0	26	325
VMs with SLA 1	38	344
VMs with SLA 2	36	331
Initial CPU load	784 (28%)	9,023 (33%)
Initial revenue	33,973 US$	330,645 US$
Discrete time instants	24 h	24 h

Selection strategy	Objective functions averages					Dominance ($row \succ column$)					Preference ($row \succ_p column$)				
	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$S1$	$S2$	$S3$	$S4$	$S5$	$S1$	$S2$	$S3$	$S4$	$S5$
10x100.vmp															
$S1$	9,908	19,981	32,623	44	1,526	■					■				
$S2$	9,827	19,991	32,623	8	180		■				≻	■		≻	
$S3$	9,639	19,228	32,623	6	124	≻	≻	■			≻	≻	■	≻	≻
$S4$	8,543	21,038	32,623	19	520				■		≻			■	
$S5$	10,395	21,957	32,623	5	150					■					■
100x1000.vmp															
$S1$	104,559	371,664	325,217	650	26,886	■					■				
$S2$	104,835	373,467	325,217	37	1,204		■				≻	■		≻	
$S3$	104,378	370,489	325,217	26	804	≻	≻	■			≻	≻	■	≻	
$S4$	103,175	374,210	325,217	92	3,531				■		≻			■	
$S5$	104,860	373,230	325,217	20	618					■		≻		≻	■

Fig. 3 Selection Strategy Comparison. For selection strategy notation see Sect. 3.3

3.5 Experiment 4: Selection Strategy Evaluation

Figure 3 summarizes the results obtained in both experiments. As expected, when the lexicographic order is used, the most important objective function is the one with the best results, i.e., the $S4$ strategy (provider preference) obtains the best results in power consumption $f_1(x, t)$, with 20% less power consumption than the worst strategy in the 10×100.vmp instance and 2% less power consumption than the worst strategy in 100×1000.vmp. Analogously, when service perspective is prioritized (strategy $S5$), the objective functions $f_4(x, t)$ and $f_5(x, t)$ obtain the best results (Table 9).

However, as the focus of the work presented in [7] is the simultaneous optimization of all five objective functions with a multi-objective approach, a comparison is made considering the concept of Pareto dominance. As seen in Fig. 3 (dominance column), the $S3$ strategy dominates $S2$ and $S1$ in both experiments; however, it is non-comparable with respect to $S4$ and $S5$ in both tested problem instances.

Given that $S3$ cannot be declared as the best strategy considering exclusively Pareto dominance, a further comparison of selection strategies using the preference criteria (i.e., larger number of better objective functions) [5] is presented in the corresponding column of Fig. 3.

It may seem intuitive that the $S2$ strategy (that uses the preference criterion) should be the best; however, Table 3 shows that strategy $S3$ is preferred not only to $S2$ but also to $S1$ and to $S4$ in both tested problem instances. Additionally, it can be seen that $S3$ is preferred to $S5$ in problem instance 10×100.vmp while no strategy is preferred to $S3$, indicating that $S3$ (distance to origin) is the best strategy for solving the presented MaVMP problem formulation with reconfiguration of VMs.

As a consequence of the above results, for production cloud datacenters, instead of calculating all the Pareto set or a Pareto set approximation, the $S3$ strategy (distance to origin) could be used to combine all considered objective functions into only one objective function, therefore solving the studied problem considering a Multi-Objective solved as Mono-Objective (MAM) approach. It is important to mention that the obtained results are consistent with the selection strategy evaluation presented in [28] for solution of a traffic engineering problem in computer networks.

4 Open Research Problems: Many-Objective VMP for Cloud Computing Environments

After demonstrating the viability to formulate and solve MaVMP problems for initial placement of VMs and MaVMP problems with reconfiguration of VMs, this section presents relevant open research topics for the formulation and resolution of MaVMP problems for cloud computing environments.

4.1 IaaS Environments for VMP Problems

In real-world environments, IaaS providers dynamically receive requests for the placement of VMs with different characteristics according to different dynamic parameters. In this context, preliminary results of the authors identified that the most relevant dynamic parameters in the VMP literature are [30]: (1) resource capacities of VMs (associated to vertical elasticity) [31], (2) number of VMs of a cloud service (associated to horizontal elasticity) [32] and (3) utilization of resources of VMs (relevant for overbooking) [20]. Considering the mentioned dynamic parameters, environments for IaaS formulations of VMP problems could be classified by one or more of the following classification criteria: (1) service elasticity and (2) overbooking of physical resources [30].

In order to model these advanced IaaS environments, cloud services (i.e., a set of interrelated VMs) are considered instead of just VMs. A cloud service may represent cloud infrastructures for basic services such as Domain Name Service (DNS), web applications or even elastic applications such as MapReduce programs [30].

To the best of the authors' knowledge, there is no published work considering all these fundamental criteria, directly related to the most relevant dynamic parameters in the specialized literature [30]. CSPs efficiently solving formulations of the VMP problem in advanced IaaS environments considering service elasticity, including both vertical and horizontal scaling of cloud services, as well as overbooking of physical resources, including both server (CPU and RAM) and networking resources will represent a considerable advance on this research area and its cloud datacenters will be able to scale according to trending types of requirements with sufficient flexibility. A recommended path for future work is exploring and addressing challenges of particular environments identified in [30] as research opportunities before considering this advanced IaaS environment for solving VMP problems.

4.2 Uncertainty in VMP for Cloud Computing

Extensive research of uncertainty issues could be found in several fields such as: computational biology and decision-making in economics, just to cite a few. Particularly, studies of uncertainty for cloud computing are limited and uncertainty in resource allocation and service provisioning have not been adequately addressed, representing research challenges [33].

According to [33], uncertainties in cloud computing could be grouped into: (1) parametric and (2) system uncertainties. Parametric uncertainties may represent incomplete knowledge and variation of parameters, as presented in the considered VMP problem. The analysis of these uncertainties quantifies the effect of random input parameters on model outputs [33].

Research challenges in the context of VMP problems include designing novel resource management strategies to handle uncertainty in an effective way, as described by Tchernykh et al. in [33]. IaaS providers must satisfy requests for virtual resources in highly dynamic environments. Due to the randomness of customer requests, algorithms for solving VMP problems should be evaluated under uncertainty.

4.3 Two-Phase Optimization Schemes for VMP Problems

The VMP could be formulated as both online and offline optimization problems [2]. A VMP problem formulation is considered to be online when solution techniques (e.g., heuristics) makes decisions on-the-fly, without knowing upcoming VM requests [24]. On the other hand, if solution techniques have a complete knowledge of future VM requests of a problem instance, the VMP problem formulation is considered to be offline [11]. Considering the on-demand model of cloud computing with dynamic resource provisioning and dynamic workloads of cloud applications [34], the resolution of VMP problems should be performed as fast as possible in order to be able to support these dynamic requirements. In this context, the VMP problem for IaaS environments was mostly studied in the VMP literature considering online formulations, taking into consideration that VM requests are unknown a priori [2].

It is important to consider that online decisions made along the operation of a cloud computing infrastructure negatively affects the quality of obtained solutions of VMP problems when comparing to offline decisions [35]. Clearly, offline algorithms present a substantial advantage over online alternatives, when considering the quality of obtained solutions. This advantage is presented for the following two main reasons: (1) an offline algorithm has a complete knowledge of future VM requests of a VMP problem instance (which is impracticable on real-world IaaS environments because VM requests are uncertain) and (2) it considers migration of VMs between PMs, reconfiguring the placement when convenient.

To improve the quality of solutions obtained by online algorithms, the VMP problem could be formulated as a two-phase optimization problem, combining advantages of online and offline formulations for IaaS environments. In this context, VMP problems could be decomposed into two different subproblems: (1) incremental VMP (iVMP) and (2) VMP reconfiguration (VMPr) [36]. This two-phase optimization strategy combines both online (iVMP) and offline (VMPr) algorithms for solving each considered VMP subproblem.

The iVMP subproblem is considered for attending dynamic arriving requests where VMs should be created, modified and removed at runtime. Consequently, this subproblem should be formulated as an online problem and solved as fast as possible, where existing heuristics could be reasonably appropriate. Additionally, the VMPr subproblem is considered for improving the quality of solutions obtained by the iVMP, reconfiguring a current placement $P(t)$ through migration of VMs between PMs to an improved placement $P'(t)$. This VMPr subproblem could be formulated offline, where alternative solution techniques could result more suitable (e.g., metaheuristics).

The considered iVMP + VMPr optimization scheme has been briefly studied in the specialized literature. Consequently, several challenges for IaaS environments remain unaddressed or could be improved, considering that only basic methods have been proposed, specifically for VMPr Triggering and VMPr Recovering methods:

- ***Research Question 1 (RQ1)***: when the VMPr problem should be triggered? (VMPr Triggering method).
- ***Research Question 2 (RQ2)***: what should be done with cloud service requests arriving during the VMPr reconfiguration period? (VMPr Recovering method).

Research should advance by proposing more sophisticated VMPr Triggering methods, probably considering several different objective functions, as presented in this chapter for MaVMP problems. Additionally, most of the existing research works do not consider any VMPr Recovering method, when applicable. Only Calcavecchia et al. studied in [37] a very basic approach, canceling the VMPr whenever a new request is received. Consequently, the VMPr is only performed in periods with no requests that could result unrealistic for IaaS providers. Future works could be focused on proposing novel VMPr Recovering methods.

References

1. López-Pires, F., & Barán, B. (2015). Virtual machine placement literature review. http://arxiv.org/abs/1506.01509.
2. López-Pires, F., & Barán, B. (2015). A virtual machine placement taxonomy. In *Proceedings of the 2015 IEEE/ACM 15th International Symposium on Cluster, Cloud and Grid Computing*. IEEE Computer Society.
3. Cheng, J., Yen, G. G., & Zhang, G. (2014, October). A many-objective evolutionary algorithm based on directional diversity and favorable convergence. In *2014 IEEE International Conference on Systems, Man and Cybernetics (SMC)* (pp. 2415–2420).
4. Farina, M., & Amato, P. (2002). On the optimal solution definition for many-criteria optimization problems. In *Proceedings of the NAFIPS-FLINT International Conference* (pp. 233–238).

5. von Lücken, C., Barán, B., & Brizuela, C. (2014). A survey on multi-objective evolutionary algorithms for many-objective problems. *Computational Optimization and Applications*, 1–50.
6. Guzek, M., Bouvry, P., & Talbi, E.-G. (2015). A survey of evolutionary computation for resource management of processing in cloud computing. *Computational Intelligence Magazine, IEEE, 10*(2), 53–67.
7. Ihara, D., López-Pires, F., & Barán, B. (2015). 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.
8. López-Pires, F., & Barán, B. (2015). A many-objective optimization framework for virtualized datacenters. In *Proceedings of the 2015 5th International Conference on Cloud Computing and Service Science* (pp. 439–450).
9. López-Pires, F., & Barán, B. (2017). Cloud computing resource allocation taxonomies. *International Journal of Cloud Computing* (To appear).
10. Gao, Y., Guan, H., Qi, Z., Hou, Y., & Liu, L. (2013). A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. *Journal of Computer and System Sciences*, *79*, 1230–1242.
11. López-Pires, F., & Barán, B. (2013). 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* (pp. 203–210). IEEE Computer Society.
12. Tomás, L., & Tordsson, J. (2013). Improving cloud infrastructure utilization through overbooking. In *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference, CAC'13* (pp. 5:1–5:10). New York, NY, USA.
13. Beloglazov, A., Abawajy, J., & Buyya, R. (2012). Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, *28*(5), 755–768.
14. Shrivastava, V., Zerfos, P., Lee, K.-W., Jamjoom, H., Liu, Y.-H., & Banerjee, S. (2011). Application-aware virtual machine migration in data centers. In *INFOCOM, 2011 Proceedings IEEE* (pp. 66–70). IEEE.
15. Donoso, Y., Fabregat, R., Solano, F., Marzo, J.-L., & Barán, B. (2005). Generalized multiobjective multitree model for dynamic multicast groups. In *2005 IEEE International Conference on Communications, 2005. ICC 2005* (Vol. 1, pp. 148–152). IEEE.
16. Báez, M., Zárate, D., & Barán, B. (2007). Adaptive memetic algorithms for multi-objective optimization. In *2007 XXXIII Latin American Computing Conference (CLEI)* (Vol. 2007).
17. Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. A. M. T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197.
18. Coello Coello, C., Lamont, G. B., & Van Veldhuizen, D. A. (2007). *Evolutionary algorithms for solving multi-objective problems*. Springer.
19. Sun, M., Gu, W., Zhang, X., Shi, H., & Zhang, W. (2013). A matrix transformation algorithm for virtual machine placement in cloud. In *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)* (pp. 1778–1783). IEEE.
20. Anand, A., Lakshmi, J., & Nandy, S. K. (2013). Virtual machine placement optimization supporting performance SLAs. In *2013 IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom)* (Vol. 1, pp. 298–305. IEEE.
21. Sato, K., Samejima, M., & Komoda, N. (2013). Dynamic optimization of virtual machine placement by resource usage prediction. In *2013 11th IEEE International Conference on Industrial Informatics (INDIN)* (pp. 86–91). IEEE.
22. Shi, L., Butler, B., Botvich, D., & Jennings, B. (2013). Provisioning of requests for virtual machine sets with placement constraints in iaas clouds. In *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)* (pp. 499–505). IEEE.
23. Li, W., Tordsson, J., & Elmroth, E. (2011). Modeling for dynamic cloud scheduling via migration of virtual machines. In *2011 IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom)* (pp. 163–171). IEEE.

24. Beloglazov, A., & Buyya, R. (2012). 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.
25. López-Pires, F., & Barán, B. (2017). Many-objective virtual machine placement. *Journal of Grid Computing* (In Review).
26. Tomás, L., & Tordsson, J. (2013). Improving cloud infrastructure utilization through overbooking. In *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference* (p. 5).
27. Svärd, P., Hudzia, B., Walsh, S., Tordsson, J., & Elmroth, E. (2015). Principles and performance characteristics of algorithms for live vm migration. *ACM SIGOPS Operating Systems Review*, *49*(1), 142–155.
28. Talavera, F., Crichigno, J., & Barán, B. (2005). Policies for dynamical multiobjective environment of multicast traffic engineering. In *IEEE ICT*.
29. Amazon Web Services (2015, June). Amazon ec2 instances. http://aws.amazon.com/ec2/instance-types/.
30. Ortigoza, J., López-Pires, F., & Barán, B. (2016, April). A taxonomy on dynamic environments for provider-oriented virtual machine placement. In *2016 IEEE International Conference on Cloud Engineering (IC2E)* (pp. 214–215).
31. Li, K., Wu, J., & Blaisse, A. (2013). Elasticity-aware virtual machine placement for cloud datacenters. In *2013 IEEE 2nd International Conference on Cloud Networking (CloudNet)* (pp. 99–107). IEEE.
32. Wang, W., Chen, H., & Chen, X. (2012). An availability-aware virtual machine placement approach for dynamic scaling of cloud applications. In *2012 9th International Conference on Ubiquitous Intelligence & Computing and 9th International Conference on Autonomic & Trusted Computing (UIC/ATC)* (pp. 509–516). IEEE.
33. Tchernykh, A., Schwiegelsohn, U., Alexandrov, V., & Talbi, E.-G. (2015). Towards understanding uncertainty in cloud computing resource provisioning. *Procedia Computer Science*, *51*, 1772–1781.
34. Mell, P., & Grance, T. (2009). The nist definition of cloud computing. *National Institute of Standards and Technology*, *53*(6), 50.
35. López-Pires, F., Barán, B., Amarilla, A., Benítez, L., Ferreira, R., & Zalimben, S. (2016). An experimental comparison of algorithms for virtual machine placement considering many objectives. In *9th Latin America Networking Conference (LANC)* (pp. 75–79).
36. Zheng, Q., Li, R., Li, X., Shah, N., Zhang, J., Tian, F., et al. (2015). Virtual machine consolidated placement based on multi-objective biogeography-based optimization. *Future Generation Computer Systems*.
37. Calcavecchia, N. M., Biran, O., Hadad, E., & Moatti, Y. (2012). Vm placement strategies for cloud scenarios. In *2012 IEEE 5th International Conference on Cloud Computing (CLOUD)* (pp. 852–859). IEEE.