A Many-objective Optimization Framework for Virtualized Datacenters

Fabio López Pires[1,2] and Benjamín Barán[2,3]
[1]*Itaipu Technological Park (PTI), Hernandarias, Paraguay*
[2]*National University of Asunción (UNA), San Lorenzo, Paraguay*
[3]*National University of the East (UNE), Ciudad del Este, Paraguay*

Keywords: Virtual Machine Placement, Many-objective Optimization, Datacenter, Virtualization, Cloud Computing.

Abstract: The process of selecting which virtual machines should be located (i.e. executed) at each physical machine of a datacenter is commonly known as Virtual Machine Placement (VMP). This work presents a general many-objective optimization framework that is able to consider as many objective functions as needed when solving the VMP problem in a pure multi-objective context. As an example of utilization of the proposed framework, for the first time a formulation of the many-objective VMP problem (MaVMP) is proposed, considering the simultaneous optimization of the following five objective functions: (1) power consumption, (2) network traffic, (3) economical revenue, (4) quality of service and (5) network load balancing. To solve the formulated many-objective VMP problem, an interactive memetic algorithm is proposed. Simulations prove the correctness of the proposed algorithm and its effectiveness converging to a treatable number of solutions in different experimental scenarios.

1 INTRODUCTION

One of the key challenges in modern datacenters is to efficiently manage power consumption, considering electricity costs and the carbon dioxide footprints (Beloglazov et al., 2011). 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 (Barroso and Hölzle, 2007). Consequently, applying techniques for higher resource utilization can result in more energy-efficient server operation. Virtualization of computational resources is a technology that dynamically improves the utilization of available resources in a datacenter according to the existing demand, improving efficiency. Correctly locating virtual machines (VMs) into physical machines (PMs) reduces the amount of hardware in use, letting unused PMs to be in standby mode or even to shut down. This way, average resource utilization as well as energy efficiency may be improved, resulting in better economical revenue and greener datacenters.

Virtualization in modern datacenters introduces management decisions related to the placement of VMs. In this context, Virtual Machine Placement (VMP) is the process of selecting which VMs should be executed in a given set of PMs of a datacenter.

1.1 Background and Motivation

In datacenters with a considerable amount of PMs and VMs, there is a large number of possible criteria that can be considered when selecting a placement, depending on the priorities and optimization objectives. These criteria can even change from one period of time to another, which implies a variety of possible formulations of the VMP problem and objective functions to be optimized for virtualized datacenters.

According to (López Pires and Barán, 2015), the optimization of the energy consumption is the most studied objective function in VMP literature (Sun et al., 2013; Beloglazov et al., 2012). On the other hand, network traffic (Anand et al., 2013), economical revenue (Shi et al., 2013; Sato et al., 2013), performance (Bin et al., 2011) and resource utilization (Mishra and Sahoo, 2011) optimization are also very studied. For each objective function, several possible formulations can be proposed.

In the VMP context, objective functions can be studied according to the following identified optimization approaches: (1) mono-objective (MOP), (2) multi-objective solved as mono-objective (MAM) and (3) multi-objective (PMO) (López Pires and Barán, 2015). The mono-objective approach considers the optimization of only one objective or the individual

López Pires F. and Barán B..
A Many-objective Optimization Framework for Virtualized Datacenters.
DOI: 10.5220/0005434604390450
In *Proceedings of the 5th International Conference on Cloud Computing and Services Science* (CLOSER-2015), pages 439-450
ISBN: 978-989-758-104-5

optimization of more than one objective, one at a time. On the other hand, multi-objective solved as mono-objective approach considers the optimization of multiple objectives combined into one objective (usually as a weighted sum of different normalized objectives), while a pure multi-objective approach considers the simultaneous optimization of different (possible contradictory) objectives. To the best of the authors' knowledge, there is no many-objective optimization formulation proposed for the VMP problem in the specialized literature (López Pires and Barán, 2015), i.e a multi-objective optimization problem with at least four conflicting objective functions (von Lücken et al., 2014).

Considering the large number of existing objective functions for the VMP problem, this work presents a many-objective optimization framework to be able to consider as many objective functions as needed when solving the VMP problem. As an example of utilization of the proposed framework, for the first time a formulation of the many-objective VMP problem (MaVMP) is proposed, considering the following five objective functions: (1) power consumption minimization, (2) network traffic minimization, (3) economical revenue maximization, (4) QoS maximization and (5) network load balancing optimization. In the presented formulation, a multi-level priority is associated to each VM, representing a Service Level Agreement (SLA) considered in the placement process. To solve the formulated MaVMP problem, an interactive memetic algorithm is proposed considering the issues that give place the formulated problem of many-objective optimization for Pareto-based algorithms.

This paper is structured in the following way: Section 2 presents a multi-objective optimization problem formulation, considering the issues that give place the problem of many-objective optimization. Section 3 details the proposed general many-objective optimization framework, while Section 4 summarizes a many-objective formulation of the VMP problem considering the simultaneous optimization of five objective functions and a multi-level priority of SLA. Section 5 presents a novel interactive memetic algorithm proposed for solving the formulated many-objective problem, while Section 6 presents first experimental results. Finally, conclusions and future work are left to Section 7.

2 MULTI-OBJECTIVE OPTIMIZATION

A general pure multi-objective optimization problem (PMO) includes a set of p decision variables, q objective functions, and r constraints. Objective functions and constraints are functions of decision variables. In a PMO formulation, x represents the decision vector, while y represents the objective vector. The decision space is denoted by X and the objective space as Y. These can be expressed as (Coello et al., 2007):

Optimize:

$$y = f(x) = [f_1(x), f_2(x), ..., f_q(x)] \quad (1)$$

subject to:

$$e(x) = [e_1(x), e_2(x), ..., e_r(x)] \geq 0 \quad (2)$$

where:

$$x = [x_1, x_2, ..., x_p] \in X \quad (3)$$

$$y = [y_1, y_2, ..., y_q] \in Y \quad (4)$$

It is important to remark that optimizing, in a particular problem context, can mean maximizing or minimizing. The set of constrains $e(x) \geq 0$ defines the set of feasible solutions $X_f \subset X$ and its corresponding set of feasible objective vectors $Y_f \subset Y$. The feasible decision space X_f is the set of all decision vectors x in the decision space X that satisfies the constraints $e(x)$, and it is defined as:

$$X_f = \{x \mid x \in X \wedge e(x) \geq 0\} \quad (5)$$

The feasible objective space Y_f is the set of the objective vectors y that represents the image of X_f onto Y and it is denoted by:

$$Y_f = \{y \mid y = f(x) \quad \forall x \in X_f\} \quad (6)$$

To compare two solutions in a multi-objective context, the concept of Pareto dominance is used. Given two feasible solutions $u, v \in X$, u dominates v, denoted as $u \succ v$, if $f(u)$ is better or equal to $f(v)$ in every objective function and strictly better in at least one objective function. If neither u dominates v, nor v dominates u, u and v are said to be non-comparable (denoted as $u \sim v$).

A decision vector x is non-dominated with respect to a set U, if there is no member of U that dominates x. The set of non-dominated solutions of the whole set of feasible solutions X_f, is known as optimal Pareto set P^*. The corresponding set of objective vectors constitutes the optimal Pareto front PF^*.

PMOs with more than three objective functions are known as Many-Objective Optimization Problems (MaOPs), as defined in (Cheng et al., 2014). MaOPs differ significantly from PMOs because several issues should be considered when solving problems with more than three objective functions (Farina and Amato, 2002). In case of Pareto-based algorithms, these

issues are intrinsically related to the fact that as the number of objective increases, the proportion of non-dominated elements in the population grows, being increasingly difficult to discriminate among solutions using only the Pareto dominance relation (Deb et al., 2006). Additionally, determining which solution to keep and which to discard in order to converge toward the Pareto set is still a relevant issue to be addressed (Farina and Amato, 2002). Pareto-based algorithms are still not able to provide the required selection pressure towards better solutions in order to conduct an efficient evolutionary search and, even elitism may be difficult. As the number of objectives grows, the proportion of non-comparable solutions to the total number of solutions tends to one (von Lücken et al., 2014), making more difficult to solve a MaOP. Clearly, difficulties in solving MaOPs explain why it has not yet been studied in the VMP literature.

3 MANY-OBJECTIVE OPTIMIZATION FRAMEWORK

The general many-objective optimization framework for the VMP problem proposed in this work considers that as the number of conflicting objectives of a MaVMP problem formulation increases, the total number of non-dominated solutions increases (even exponentially in some cases), being increasingly difficult to discriminate among solutions using only the dominance relation (Farina and Amato, 2002). For this reason, this work proposes the utilization of lower and upper bounds associated to each objective function $z \in \{1,\ldots,q\}$ ($L_z \leq f_z(x) \leq U_z$) to be able to reduce iteratively the number of possible compromise solutions of the Pareto set approximation, when needed by the decision maker.

A VMP formulation, based on many objective functions and constraints to be detailed in Section 4, may be written as:

Optimize:

$$y = f(x) = [f_1(x), f_2(x), ..., f_q(x)] \tag{7}$$

where:

$$\begin{aligned} f_1(x) &= \text{power consumption minimization} \\ f_2(x) &= \text{network traffic minimization} \\ f_3(x) &= \text{economical revenue maximization} \\ f_4(x) &= \text{QoS maximization} \\ f_5(x) &= \text{network load balancing optimization} \\ &\vdots \\ f_q(x) &= \text{any other considered function} \end{aligned} \tag{8}$$

subject to:

$$\begin{aligned} e_1(x) &: \text{unique placement of VMs;} \\ e_2(x) &: \text{assure provisioning of highest SLA;} \\ e_3(x) &: \text{processing resource capacity of PMs;} \\ e_4(x) &: \text{memory resource capacity of PMs;} \\ e_5(x) &: \text{storage resource capacity of PMs;} \\ e_6(x) &: f_1(x) \in [L_1, U_1]; \\ e_7(x) &: f_2(x) \in [L_2, U_2]; \\ e_8(x) &: f_3(x) \in [L_3, U_3]; \\ e_9(x) &: f_4(x) \in [L_4, U_4]; \\ e_{10}(x) &: f_5(x) \in [L_5, U_5]; \\ &\vdots \\ e_r(x) &: \text{any other considered constraint.} \end{aligned} \tag{9}$$

4 MANY-OBJECTIVE VIRTUAL MACHINE PLACEMENT

A few articles proposed formulations of a pure multi-objective VMP problem (MVMP) (Gao et al., 2013; López Pires and Barán, 2013), considering the simultaneous optimization of at most three objective functions. To the best of the authors' knowledge, this work proposes for the first time the 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 this many-objective formulation, a multi-level priority is associated to each VM, representing a SLA. Formally, the proposed offline many-objective optimization VMP problem can be enunciated as:

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 in 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.

4.1 Input Data

The proposed formulation of the VMP problem models a virtualized datacenter infrastructure, composed by PMs and a network topology. The set of PMs

is represented as a matrix H of dimension (n x 4). Each H_i is represented by processing resources CPU (as ECU)[1], RAM memory [GB], storage [GB] and a maximum power consumption [W] as:

$$H_i = [Hcpu_i, Hram_i, Hhdd_i, pmax_i] \quad \forall i \in \{1,...,n\} \tag{10}$$

where:

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

As shown in Figure 1, the network topology of the virtualized datacenter is represented as:

- G: Network topology;
- L: Set of links l_a in G. For simplicity, we assume that all links are semi-duplex;
- M: Set of paths for all-to-all PM interconnections;
- K: Capacity set of the communication channel, typically in [Mbps].

The set of VMs requested by customers is represented as a matrix V of dimension (m x 5). Each V_j requires processing resources CPU (as ECU)[1], RAM memory [GB] and storage [GB], providing for them an economical revenue R_j [$] for the provider. A SLA is also assigned to each VM to indicate its level of priority. Consequently, a V_j is represented as:

$$V_j = [Vcpu_j, Vram_j, Vhdd_j, R_j, SLA_j] \quad \forall j \in \{1,...,m\} \tag{11}$$

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 $SLA_j \in \{1,\ldots,s\}$;
- m: Number of VMs.

The traffic between VMs is represented as a matrix T of dimension (m x m). Each V_j requires network communication resources [Mbps] to communicate with other VMs. These communication resources are represented as:

$$T_j = [T_{j1}, T_{j2}, ..., T_{jm}] \quad \forall j \in \{1,...,m\} \tag{12}$$

where:

- T_{jk}: Average network traffic between V_j and V_k [Mbps]. Note that we can consider $T_{jj} = 0$.

Figure 1 presents a basic example of a virtualized datacenter infrastructure, composed by 4 PMs $H = \{H_1, H_2, H_3, H_4\}$ and a network topology considering 6 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] respectively. 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 Figure 1.

4.2 Output Data

A calculated solution should indicate the exact placement of each VM V_j on the necessary PMs H_i, considering the many-objective optimization criteria applied. A placement (or possible solution to the formulated problem) is represented in what follows 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., $P_{ji} : V_j \rightarrow H_i$).

4.3 Constraints

4.3.1 Constraint 1: Unique Placement of VMs

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

$$\sum_{i=1}^{n} P_{ji} \leq 1 \quad \forall j \in \{1,...,m\} \tag{13}$$

where:

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

4.3.2 Constraint 2: Assure SLA Provisioning

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

$$\sum_{i=1}^{n} P_{ji} = 1 \quad \forall j \text{ such that } SLA_j = s \tag{14}$$

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

4.3.3 Constraints 3-5: Physical Resource Capacity 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. In this work, it is not considered the overbooking of resources (Tomás and Tordsson, 2013). Consequently, this set of constraints can be mathematically formulated as:

$$\sum_{j=1}^{m} Vcpu_j \times P_{ji} \leq Hcpu_i \quad (15)$$

$$\sum_{j=1}^{m} Vram_j \times P_{ji} \leq Hram_i \quad (16)$$

$$\sum_{j=1}^{m} Vhdd_j \times P_{ji} \leq Hhdd_i \quad (17)$$

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

4.3.4 Adjustable Constraints

This work proposes 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 the decision maker. Consequently, this set of adjustable bounds can be mathematically formulated as the following constraints:

$$f_z(x) \in [L_z, U_z] \quad \forall z \in \{1,\ldots,q\} \quad (18)$$

4.4 Objective Functions

A VMP problem can be defined as a many-objective optimization problem, considering the simultaneous optimization of more than three objective functions.

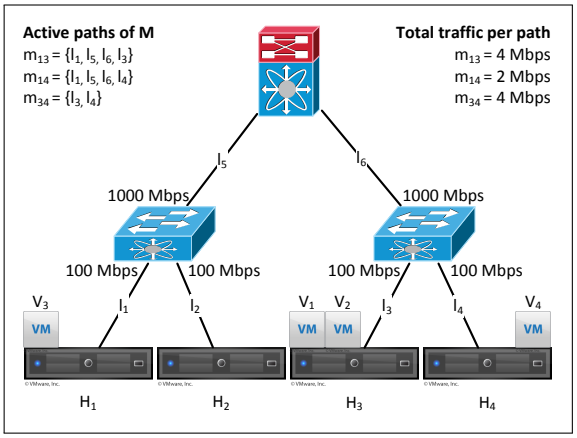

Figure 1: Example of placement in a virtualized datacenter infrastructure, composed by PMs and a network topology.

As a concrete example, this work proposes for the first time the optimization of the following five objective functions:

4.4.1 Power Consumption Minimization

Based on (Beloglazov et al., 2012) formulation, this work also proposes the minimization of power consumption, represented by the sum of the power consumption of each PM H_i:

$$f_1(x) = \sum_{i=1}^{n} ((pmax_i - pmin_i) \times Ucpu_i + pmin_i) \times Y_i \quad (19)$$

where:

- $f_1(x)$: Total power consumption of the PMs;
- $pmin_i$: Minimum power consumption of H_i. In what follows, $pmin_i = pmax_i * 0.6$ (Beloglazov et al., 2012);
- $Ucpu_i$: Utilization ratio of processing resources used by H_i;
- Y_i: Binary variable equals 1 if H_i is turned on; otherwise, it is 0.

4.4.2 Network Traffic Minimization

(Shrivastava et al., 2011) proposed the minimization of network traffic among VMs by maximizing locality. Based on this approach, this work proposes equation (20) 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.

$$f_2(x) = \sum_{j=1}^{m} \sum_{k=1}^{m} (T_{jk} \times D_{jk}) \quad (20)$$

where:

- $f_2(x)$: Total network traffic among VMs;
- 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 do not contribute to increase the total network traffic given by equation (20); therefore, $D_{jk} = 0$ if $P_{ji} = P_{ki} = 1$.

4.4.3 Economical Revenue Minimization

Based on (López Pires and Barán, 2013), this work presents equation (21) 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 placement that is effectively located for execution on any PM.

$$f_3(x) = \sum_{j=1}^{m} (R_j \times X_j) \tag{21}$$

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.

4.4.4 QoS Maximization

In this work, 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 proposed in equation (22).

$$f_4(x) = \sum_{j=1}^{m} (\hat{C}^{SLA_j} \times SLA_j \times X_j) \tag{22}$$

where:

$f_4(x)$: Total QoS figure for a given placement;
$\hat{C}$: Constant, large enough to prioritize services with larger *SLA* over the ones with lower *SLA* (see example in Section 4.4.6).

4.4.5 Network Load Balancing Optimization

This work calculates the total amount of traffic going through a semi-duplex link l_a as:

$$T_{l_a} = \sum_{i=1}^{n} \sum_{i'=1}^{n} F_{aii'} \times \left(\sum_{j=1}^{m} \sum_{j'=1}^{m} P_{ji} \times P_{j'i'} \times D_{jj'} \times T_{jj'} \right) \tag{23}$$

where:

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

Inspired in (Donoso et al., 2005) formulation, this work calculates the Maximum Link Utilization (MLU) as:

$$MLU = \max_{\forall l_a \in L} \left(\frac{Tl_a}{Cl_a} \right) \tag{24}$$

where:

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

In this paper, the load balancing optimization of the network is formulated as the minimization of the MLU, i.e.,

$$f_5(x) = MLU \tag{25}$$

4.4.6 Example

The following example details how the values of $f_4(x)$ and $f_5(x)$ are calculated, considering the datacenter infrastructure presented in Figure 1. For the other three objective functions, see (López Pires and Barán, 2013).

Consider the following placement matrix P where, VMs V_1 and V_2 are executed in H_3, while V_3 is executed in H_1 and V_4 in H_4:

$$P = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{26}$$

Additionally, consider $\hat{C} = 100$ and the values of SLA_j and X_j presented in Table 1, $f_4(x)$ is calculated as:

$$\begin{aligned} f_4(x) &= \hat{C}^{SLA_1} \times SLA_1 \times X_1 + \cdots + \hat{C}^{SLA_4} \times SLA_4 \times X_4 \\ &= 100^2 \times 2 \times 1 + \cdots + 100^3 \times 3 \times 1 \\ &= 3.06\text{x}10^6 \end{aligned} \tag{27}$$

It is important to remark that if $\hat{C}$ is not sufficiently large, $f_4(x)$ could prefer a large number of VMs with lower priority. As an example, if $\hat{C} = 1$, 2 VMs with $SLA_j = 2$ result in a better figure of $f_4(x)$ than 1 VM with $SLA_j = 3$, what is not correct according to what was presented in Section 4.4.4.

On the other hand, considering that each VM V_j communicates at 1 [Mbps] with any other $V_k (k \neq j)$, the placement P results in the values of Tl_a and Cl_a presented in Table 2. So $f_5(x)$ is calculated as:

$$\begin{aligned} f_5(x) &= \max \left(\frac{Tl_1}{Cl_1}, \frac{Tl_2}{Cl_2}, \frac{Tl_3}{Cl_3}, \frac{Tl_4}{Cl_4}, \frac{Tl_5}{Cl_5}, \frac{Tl_6}{Cl_6} \right) \\ &= \max \left(\frac{6}{100}, \frac{0}{100}, \frac{8}{100}, \frac{6}{100}, \frac{6}{1000}, \frac{6}{1000} \right) \\ &= \frac{8}{100} \end{aligned} \tag{28}$$

5 INTERACTIVE MEMETIC ALGORITHM

A Memetic Algorithm (MA) could be understood as an Evolutionary Algorithm (EA) that in addition to

Table 1: Data for requested VMs used in basic example (see Section 4.4.6).

V_j	SLA_j	X_j
V_1	2	1
V_2	2	1
V_3	2	1
V_4	3	1

Table 2: Data of network links used in basic example (see Section 4.4.6).

l_a	Tl_a [Mbps]	Cl_a [Mbps]	Tl_a/Cl_a
l_1	6	100	6/100
l_2	0	100	0/100
l_3	8	100	8/100
l_4	6	100	6/100
l_5	6	1000	6/1000
l_6	6	1000	6/1000
MLU			8/100

the standard selection, crossover and mutation operators of most Genetic Algorithms (GA) includes a local optimization operator to obtain good solutions even at early generations of an EA (Báez et al., 2007).

This work proposes an interactive memetic algorithm for solving the VMP problem in a many-objective context, considering the proposed formulation presented in Section 4 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 (von Lücken et al., 2014) that many-objective optimization using Multi-Objective Evolutionary Algorithms (MOEAs) is an active research area, having multiple challenges that need to be addressed regarding scalability analysis, visualization of the results, algorithm design and experimental algorithm evaluation. The interactive memetic algorithm presented in this section as a viable way to solve a many-objective VMP problem, proposes the inclusion of desirable ranges of values for the objective functions costs in order to interactively control the possible huge number of feasible non-dominated solution, as described in Section 2. The proposed interactive memetic algorithm is based on the one proposed in (López Pires and Barán, 2013) and works as follows:

In step 1, it is verified if the problem has at least one solution (considering only VMs with $SLA_j = s$) 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 2, which generates a set of random candidates P_0, whose solutions are repaired at step 3 to ensure that P_0 contains only feasible solutions. Then, the algorithm tries to improve candidates at step 4 using local search. With the obtained non-dominated solutions, the first set P_{known} (Pareto set approximation) is generated at step 5. After initialization in step 6, evolution begins (iterations between steps 7 and 18).

The evolutionary process basically follows the same behavior: solutions are selected from the union of P_{known} with the evoluationary set of solutions (or population) also known as P_t (step 8), crossover and mutation operators are applied as usual (step 9), and eventually solutions are repaired, as there may be infeasible solutions (step 10). Improvements of solutions of the evolutionary population P_t may be generated at step 11 using local search (local optimization operators). At step 12, the Pareto set approximation P_{known} is updated (if applicable); while at step 13 the generation (or iteration) counter is updated. At step 15 the decision maker adjust the lower and upper bounds if it is necessary, while at step 17 a new evolutionary population P_t is selected. The evolutionary process is repeated until the algorithm meets a stopping criterion (such as a maximum number of generations), finally returning the set of non-dominated solutions P_{known} in step 19.

Algorithm 1: Interactive Memetic Algorithm.

Data: datacenter infrastructure (see Section 4.1)
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** *is not stopping criterion* **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 action (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}

5.1 Population Initialization

Initially, a set of solutions (or population P_0) is randonmly generated. Each solution (or individual) is represented as $C = [C_1, C_2, \ldots, C_m]$. The possible val-

ues that can take each C_k for VMs with the highest value of SLA_j ($SLA_j = s$) are in the range [1, n]. For VMs V_j with $SLA_j < s$, the possible values are in the range [0, n]. Within these ranges defined by the SLA_j of each V_j, the algorithm ensures at the initialization stage that all VMs V_j with the highest level of priority will be located for execution on a PM H_i, while for VMs V_j with lower levels of priority SLA_j, there is always a probability larger than 0 that they may not be located for execution in any PM.

5.2 Infeasible Solution Reparation

With a random generation at the initialization phase (step 2 of Algorithm 1) and/or solutions generated by standard genetic operators (step 9 of Algorithm 1), infeasible solutions may appear, i.e. the resources required by the VMs located for execution on certain PMs could exceed the available resources (see Section 4.3.3), or one or more objectives functions may not meet the adjustable constraints (see Section 4.3.4).

Repairing these 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 and infeasible (Algorithm 2). Later, in the process of repairing infeasible solutions (Algorithm 3), the solutions which do not meet the feasibility criteria 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. Note that at step 3 of Algorithm 3, V_j migration to H'_i can be done to other PMs, even if they are shut down.

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 ≤ 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 ≤ m and feasible = false do
 3 |   if it is possible then
 4 |   |   migrate V_j to H'_i (i' ≠ i)
 5 |   else
 6 |   |   if SLA_j ≠ 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
```

5.3 Local Search

Once a population contains only feasible solutions, a local search is performed (steps 4 and 11 of Algorithm 1) for improving the solutions found so far in the evolutionary population P_t. The local search pseudo-code is presented in Algorithm 4.

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' ≠ 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' ≠ i)
 7 |   end
 8 end
 9 return set of feasible optimized solutions P''_t
```

For each individual in the population P_t, the proposed algorithm attempts to optimize a solution with a local search (step 2 of Algorithm 4). For this purpose, with probability $\frac{1}{2}$, the algorithm tries to maximize the number of located VMs with higher level of priority, locating all possible VMs that were not located so far, directly increasing $f_4(x)$ (total QoS) and $f_3(x)$ (total economical revenue) (steps 3 to 5 of Algorithm

4). On the other hand, also with probability $\frac{1}{2}$, the algorithm tries to minimize 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 (QoS, economical revenue and power consumption) is achieved, as experimentally verified with results presented in next section.

5.4 Fitness Function

The fitness function used in the proposed algorithm is the one presented in (Deb et al., 2002). This fitness value defines a non-domination rank in which a value equal to its Pareto dominance level (1 is the highest level of dominance, 2 is the next, and so on) is assigned to each individual of the population. Between two individuals with different non-domination rank, the individual with lower value (higher level of dominance) is considered better. To compare individuals with the same non-domination rank, a crowding distance is used. The basic idea is to find the Euclidean distance (properly normalized when the objectives have different measure units) between each pair of individuals, based on the q objectives, in a hyper-dimensional space (Deb et al., 2002). The individual with larger crowding distance is considered better.

5.5 Variation Operators

The proposed interactive memetic algorithm uses a Binary Tournament approach for selecting individuals for crossover and mutation (Coello et al., 2007). The crossover operator used in the presented work is the single point cross-cut (Coello et al., 2007). The selected individuals in the ascending population are replaced by descendants individuals.

This work 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 memetic algorithm is based on the population evolution proposed in (Deb et al., 2002). 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 operators.

5.6 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 huge number of potencial solutions in the Pareto set approximation P_{known}, while observing the evolution of its corresponding Pareto front PF_{known}.

6 EXPERIMENTAL RESULTS

To validate the proper operation of the proposed interactive memetic algorithm for solving the MaVMP on a purely many-objective context, different experiments were proposed for problem instances considering both homogeneous as well as heterogeneous hardware configurations of PMs, considering VMs instance types offered by Amazon Elastic Compute Cloud (EC2)[2]. Experiment 1 considered 2 problem instances with homogeneous hardware configurations of PMs ($Hcpu = 4$ [ECU], $Hram = 16$ [GB], $Hhdd = 150$ [GB], $pmax = 1740$ [W]), while Experiment 2 considered heterogeneous hardware configurations of PMs according to Table 6.

A detailed description of the hardware of the VMs instance types considered for the experiments is presented in Table 3. A general description of the considered problem instances is presented in Table 4, while the complete set of datacenter infrastructure input files used for the experiments with the corresponding experimental results are available online[3].

6.1 Experiment 1: Quality of Solutions

To compare the results obtained by the proposed interactive memetic algorithm and to validate its proper operation, an exhaustive search algorithm 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 memetic algorithm after evolving populations with 100 individuals for 100 generations. Both algorithms were implemented

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

[3] https://sites.google.com/site/flopezpires/

Table 3: Instance types of VMs considered in experiments. For notation see equation (11).

Instance Type	Vcpu	Vram	Vhdd	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.xlarge	4	15	80	201
m3.2xlarge	8	30	160	403
c3.large	2	4	32	75
c3.xlarge	4	8	80	151
c3.2xlarge	8	15	160	302
c3.4xlarge	16	30	320	604
c3.8xlarge	32	60	640	1209
r3.large	2	15	32	126
r3.xlarge	4	30	80	252
r3.2xlarge	8	61	160	504
r3.4xlarge	16	122	0	320
r3.8xlarge	32	244	0	320

Table 4: Problem instances considered in experiments, all with 50% of VMs with the highest SLA s = 2.

Input File	# PMs	# VMs	$(n+1)^m$
3x5.vmp	3	5	1024
4x8.vmp	4	8	390625
12x50.vmp	12	50	$\sim 5\text{x}10^{55}$

using ANSI C programming language (gcc)[4] and the source code is also available online[3].

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.

For each problem instance considered in this experiment (3x5.vmp and 4x12.vmp), one run of the exhaustive search algorithm was completed, obtaining the optimal Pareto front $PF*$ and its corresponding Pareto set $P*$. Furthermore, ten runs of the proposed algorithm were completed, after evolving populations of 100 individuals for 100 generations at each run. The results obtained by the proposed algorithm for each run were combined to obtain the Pareto front PF_{known} and its corresponding Pareto set P_{known}. For the 3x5.vmp and 4x8.vmp problem instances, the proposed algorithm obtained every solution of the optimal Pareto set and its corresponding Pareto front. A summary of the number of elements in the corresponding Pareto sets obtained are presented in Table 5.

[4]http://gcc.gnu.org

Table 5: Summary of results obtained in Experiment 1 using the proposed memetic algorithm (see Section 5).

Input File Name	# P^*	# P_{known}	% Found
3x5.vmp	51	51	100%
4x8.vmp	30	30	100%

Table 6: Types of PMs considered in Experiment 2. For notation see equation (10).

PM Type	Hcpu	Hram	Hhdd	pmax
h1.medium	180	512	10000	1000
h1.large	350	1024	10000	1300

6.2 Experiment 2: Interactive Bounds

As mentioned in Section 3, this work proposes the utilization of lower and upper bounds for each objective function $f(z)$ ($L_z \leq f_z(x) \leq U_z$) to be able to reduce iteratively the number of possible solutions of the Pareto set approximation P_{known}, when needed.

For the problem instance considered in this experiment (12x50.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 considering the large number of possible solutions for the particular problem considered (see Table 4). 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 10% of the lower bounds associated to maximization objective functions when the Pareto front has more than 200 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 power consumption and adjusted the upper bound U_1 to $U_1' = 9000$ [W], selecting only 35 of the 251 solutions (not considering 216 otherwise feasible solutions) for the $PF_{reduced}$ as shown in Figure 2.

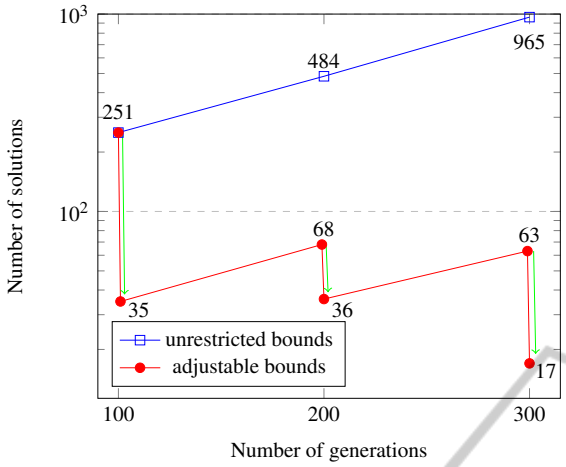

Figure 2: Summary of results obtained in Experiment 2.

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 network traffic and adjusted the upper bound U_2 to $U_2' = 115$ [Mbps], selecting only 36 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 economical revenue and adjusted the lower bound L_3 to $L_3' = 13500$ [$], selecting only 17 of the 63 solutions (not considering 46 feasible solutions) for the final $PF_{reduced}$ as shown in Figure 2. Clearly, at the end of the iterative process, the decision maker found 17 solutions according to his preferences instead of the untreatable number of 965 candidate solutions.

7 CONCLUSIONS AND FUTURE WORK

This work proposed a general many-objective optimization framework that is able to consider as many objective functions as needed when solving the VMP problem. As an example of utilization of the proposed framework, for the first time a formulation of the many-objective VMP problem (MaVMP) is proposed, considering the simultaneous optimization of the following five objective functions: (1) power consumption, (2) network traffic, (3) economical revenue, (4) quality of service and (5) network load balancing. In addition, a generalized multi-level of priorities for the SLA associated to each VM is presented. At the same time, constraints on upper and lower limits of each objective function are also recommended as a way to interactively control a potential explosion in the number of non-dominated solutions, a well known problem of many-objective optimization problems (von Lücken et al., 2014).

A short review of the most studied objective functions was also presented to demonstrate that the number of objective functions may rapidly increase once a complete understanding of the VMP problem is accomplished for practical problems were many different parameters should be ideally considered. Based on the number of possible objective functions for the VMP, this problem can clearly be addressed as a many-objective optimization problem (MaOP).

Multiple challenges need to be considered for solving a many-objective optimization problem; therefore, an interactive memetic algorithm was proposed to solve the proposed many-objective formulation, validating the presented formulation and proving that it is solvable. By no means, the authors claim that the proposed interactive memetic algorithm is the best way to solve this problem. This proposal only illustrates that it is possible to solve the problem although the VMP problem becomes harder when increasing the number of conflicting objective functions.

To validate the proposed algorithm, it was run with different problem instances and experimental results were compared to the exact solution obtained using an exhaustive search algorithm when possible. In fact, the proposed algorithm found the complete Pareto front and Pareto set (100%) for the 2 instances of Experiment 1. To validate the effectiveness of the proposed algorithm reducing the increasing number of non-dominated solutions obtained by the Pareto dominance comparison, several problem instances were executed. In fact, considering the proposed technique of interactively adjusting lower and upper bounds associated to the values of each objective functions, the Pareto front could be reduced from 965 original candidates to a treatable 17 non-dominated solutions that satisfy the decision maker preferences, as illustrated in Experiment 2.

At the time of this writing, the authors are working on VMP formulations considering more complex network topologies with full-duplex links. Considering that this is the first formulation of the VMP problem in a many-objective context, several future works can be proposed considering all the already presented objective functions, studied so far in mono-objective as well as multi-objective contexts.

Considering that this work proposed an offline (static) formulation of the VMP problem in a many-

objective optimization context, several challenges need to be addressed for online (dynamic) formulations of the problem, considering multi-objective and many-objective approaches. At the same time, different meta-heuristics, methods and algorithms should be still tested before a real good tool is ready for massive use in commercial cloud computing datacenters.

REFERENCES

Anand, A., Lakshmi, J., and Nandy, S. (2013). 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.

Báez, M., Zárate, D., and Barán, B. (2007). Algoritmos meméticos adaptativos para optimización multiobjetivo. In *XXXIII Conferencia Latinoamericana de Informática–CLEI*, volume 2007.

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

Beloglazov, A., Abawajy, J., and 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.

Beloglazov, A., Buyya, R., Lee, Y. C., Zomaya, A., et al. (2011). A taxonomy and survey of energy-efficient data centers and cloud computing systems. *Advances in Computers*, 82(2):47–111.

Bin, E., Biran, O., Boni, O., Hadad, E., Kolodner, E. K., Moatti, Y., and Lorenz, D. H. (2011). Guaranteeing high availability goals for virtual machine placement. In *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pages 700–709. IEEE.

Cheng, J., Yen, G. G., and Zhang, G. (2014). A many-objective evolutionary algorithm based on directional diversity and favorable convergence. In *Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on*, pages 2415–2420.

Coello, C. C., Lamont, G. B., and Van Veldhuizen, D. A. (2007). *Evolutionary algorithms for solving multi-objective problems*. Springer.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197.

Deb, K., Sinha, A., and Kukkonen, S. (2006). Multi-objective test problems, linkages, and evolutionary methodologies. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1141–1148. ACM.

Donoso, Y., Fabregat, R., Solano, F., Marzo, J.-L., and Barán, B. (2005). Generalized multiobjective multitree model for dynamic multicast groups. In *Communications, 2005. ICC 2005. 2005 IEEE International Conference on*, volume 1, pages 148–152. IEEE.

Farina, M. and Amato, P. (2002). On the optimal solution definition for many-criteria optimization problems. In *Proceedings of the NAFIPS-FLINT international conference*, pages 233–238.

Gao, Y., Guan, H., Qi, Z., Hou, Y., and Liu, L. (2013). A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. *Journal of Computer and System Sciences*, 79(8):1230–1242.

López Pires, F. and Barán, B. (2013). Multi-objective virtual machine placement with service level agreement. In *Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, pages 203–210. IEEE Computer Society.

López Pires, F. and 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.

Mishra, M. and Sahoo, A. (2011). On theory of vm placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 275–282. IEEE.

Sato, K., Samejima, M., and Komoda, N. (2013). Dynamic optimization of virtual machine placement by resource usage prediction. In *Industrial Informatics (INDIN), 2013 11th IEEE International Conference on*, pages 86–91. IEEE.

Shi, L., Butler, B., Botvich, D., and Jennings, B. (2013). Provisioning of requests for virtual machine sets with placement constraints in iaas clouds. In *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pages 499–505. IEEE.

Shrivastava, V., Zerfos, P., Lee, K.-W., Jamjoom, H., Liu, Y.-H., and Banerjee, S. (2011). Application-aware virtual machine migration in data centers. In *INFOCOM, 2011 Proceedings IEEE*, pages 66–70. IEEE.

Sun, M., Gu, W., Zhang, X., Shi, H., and Zhang, W. (2013). A matrix transformation algorithm for virtual machine placement in cloud. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on*, pages 1778–1783. IEEE.

Tomás, L. and Tordsson, J. (2013). Improving cloud infrastructure utilization through overbooking. In *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*, CAC '13, pages 5:1–5:10, New York, NY, USA. ACM.

von Lücken, C., Barán, B., and Brizuela, C. (2014). A survey on multi-objective evolutionary algorithms for many-objective problems. *Computational Optimization and Applications*, pages 1–50.