Multi-Objective Virtual Machine Placement with Service Level Agreement
A Memetic Algorithm Approach

Author: Ing. Fabio López Pires

Thesis developed as part of the requirements for the degree of Master of Information Technology and Communication with Emphasis in Network and Data Communications. Politecnical School, National University of Asuncion (UNA)

Advisor: D.Sc. Benjamín Barán

Abstract — The process of selecting which virtual machines should be located (i.e. executed) at each physical machine of a Datacenter is known as Virtual Machine Placement - VMP. This work proposes for the first time a multi-objective formulation of the VMP considering Service Level Agreement. A novel multi-objective memetic algorithm is also proposed to solve the formulated multi-objective problem. This proposal is validated comparing experimental results of the proposed algorithm with a brute force exhaustive search algorithm. Simulations prove the correctness of the proposed memetic algorithm and its scalability considering different experimental scenarios.

Keywords — Virtualization, Resource Allocation, Cloud Computing, Data Center, Energy Efficiency, Multi-Objective Optimization, Memetic Algorithm.

I. INTRODUCTION

Increasing power demand and energy consumption are limiting factors for the growth of modern system computing capacity due to the high economic costs associated and the high carbon dioxide (CO₂) emission [1]. Consequently, one of the key design factors in modern Data Centers is the minimization of energy consumption, which is directly related to the increase in economical revenue of a Data Center offered services. Peak power and energy consumption in Data Centers are not only determined by the power efficiency of the hardware, but also by the efficiency of the resource management system that administrates the available hardware, the whole infrastructure and the efficiency of applications (software) running on a Data Center infrastructure [1].

One of the main reasons of power and energy consumption inefficiency in a Data Center is the low average utilization of available resources. Data Centers may have average resource utilization lower than 50% presenting a huge inefficiency because workload peaks rarely occur in practice [1].

Resource virtualization is a technology that dynamically improves the utilization of available resources in a Data Center according to the existing demand, improving efficiency. This technology can be classified as a software level power management technique in contexts of specific and multiple devices, being the core technology for modern Cloud Computing. Nowadays, resource virtualization has become a widely used technology in Data Centers due to the possibility of abstraction that allows creating independent execution environments (virtual machines) to meet the requirements of customers with the available physical machines.
Correctly locating virtual machines into the physical machines reduces the amount of hardware in use, letting unused physical machines to be in standby mode or even to shut down. This way, average resource utilization as well as power and energy efficiency may be improved.

Another benefit of resource virtualization is the isolation of each virtual machine resources, even when running with other virtual machines in the same physical computer, sharing resources. Thus, resource virtualization allows a set of physical machines, centrally managed, to host multiple virtual machines, also allowing the movement of any virtual machine to a different physical machine through offline or live migration [24]. In addition to the benefits above mentioned, resource virtualization increases fault tolerance in case of hardware components failures or when physical machines require maintenance, almost without affecting the services running on the virtual machines.

*Virtual Machine Placement (VMP)* is known as the process of selecting which virtual machines should be located (i.e. executed) in a given set of physical machines of a Data Center. In Data Centers with a considerable amount of physical and virtual machines, there are many criteria that can be considered when selecting a possible solution, 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 problem and the objective functions to be optimized [7].

In this context, the authors of this paper (López and Barán) have already presented a survey and taxonomy of most relevant research work related to virtual machine placement, published in recent years in the IEEE Xplore [http://ieeexplore.ieee.org] digital library [7]. According to this taxonomy, the three main objective functions studied in the literature are: (1) energy consumption minimization, (2) network traffic minimization and (3) economical revenue maximization. These three objective functions have been studied separately in the state of the art, essentially in a mono-objective context [7].

López, Melgarejo and Barán have recently proposed for the first time a formulation of the VMP problem, considering simultaneously the three main objective functions above mentioned, in a purely multi-objective context [8]. The presented work is an extension of this first formulation proposed in [8] to a more advanced mathematical formulation that contemplates possible dependencies among the applications running on each virtual machine. More specifically, this new proposal considers Service Level Agreement (SLA) that establishes a level of criticality for each virtual machine (a requirement may be critical or noncritical), that indicates which virtual machine should be located for execution mandatorily (critical) to ensure the correct operation of mission critical applications. An example of this could be the virtual machine running a Domain Name System (DNS), which shall be located for execution mandatorily; otherwise, associated applications will not operate correctly.

Given the virtual machine placement problem formulation as a pure multi-objective optimization problem considering SLA, it is important to select an appropriate technique to solve this new mathematical formulation. After analyzing different techniques studied so far in the state of the art, this paper proposes a novel multi-objective memetic algorithm (MMA), based on an already known multi-objective memetic algorithm [17]. The proposed algorithm aims to solve the virtual machine placement problem in a purely multi-objective context [4].

This paper is structured in the following way: next section introduces a multi-objective optimization problem (MOP), while the third section presents the proposed mathematical formulation of the problem. In section IV, the objective functions of the multi-objective *virtual machine placement* are presented. Next, in section V, the proposed multi-objective memetic algorithm is described. Finally, section VI summarized experimental results and final conclusions are explained in section VII.
II. MULTI-OBJECTIVE OPTIMIZATION

A general Multi-Objective Optimization Problem (MOP) includes a set of \( n \) decision variables, \( k \) objective functions, and \( n \) constraints. Objective functions and constraints are functions of decision variables. This can be expressed as [4]:

\[
\text{Optimize } \quad y = f(x) = [f_1(x) \ f_2(x) \ldots \ f_k(x)] \\
\text{subject to } \quad e(x) = [e_1(x) \ e_2(x) \ldots \ e_n(x)] \geq 0 \\
\text{where } \quad x = [x_1 \ x_2 \ldots \ x_n] \in X \\
\text{and } \quad y = [y_1 \ y_2 \ldots \ y_k] \in Y
\]

To compare two solutions in a multi-objective context, the concept of Pareto Dominance is used. Given two feasible solutions \( u \) and \( v \in X \), \( u \) 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, is known as the Optimal Pareto set \( P_c \). The corresponding set of objective vectors constitutes the Optimal Pareto front \( PF_c \).

III. PROBLEM FORMULATION

Given a set of physical machines \( H = \{H_1, H_2, \ldots, H_n\} \) and a set of virtual machines \( V = \{V_1, V_2, \ldots, V_m\} \), it is sought the correct placement of a set of virtual machines \( V \) in a set of physical machines \( H \) satisfying resources requirements of \( V \), simultaneously optimizing the three main objective functions cited above (minimizing energy consumption, minimizing network traffic and maximizing economical revenue) in a pure multi-objective context, as it will be formalized in the next section.

A. Input Data

Each physical machine \( H_i \) has processing resources [MIPS], RAM memory [MB], storage [GB] and maximum power consumption [W] represented as:

\[
H_i = [\text{Hcpu}_i, \text{Hram}_i, \text{Hhdd}_i, \rho_{\text{max}}_i] ; \ i=1, 2, \ldots, n
\]

Each virtual machine \( V_j \) requires processing resources [MIPS], RAM memory [MB] and storage [GB], providing for them economical revenue [\$. A SLA is also assigned to each virtual machine. Consequently, a virtual machine \( V_j \) will be represented in this paper as:

\[
V_j = [\text{Vcpu}_j, \text{Vram}_j, \text{Vhdd}_j, \text{R}_j, \text{SLA}_j] ; \ j=1, \ldots, m
\]

Each virtual machine \( V_j \) requires network communication resources [Kbps] to communicate with others virtual machines. These communication resources are represented as:

\[
T_j = [T_{j1}, T_{j2}, \ldots, T_{jm}] ; \ j=1, \ldots, m
\]
In (7), $T_{jk}$ represents the average communication rate in [Kbps], between the virtual machine $V_j$ and the virtual machine $V_k$. Note that we can consider $T_{jj} = 0$.

B. Output Data

A calculated solution should indicate the placement of each virtual machine $V_j$ on the necessary physical machines $H_i$, considering the multi-objective optimization criteria that will be described in Section IV.

A placement (or possible solution to the formulated problem) is represented in what follows as a matrix $P = \{P_{ji}\}$ of dimension $(m \times n)$, where $P_{ji} \in \{0, 1\}$ indicates if the virtual machine $V_j$ is located ($P_{ji} = 1$) or not ($P_{ji} = 0$) for execution on a physical machine $H_i$ (i.e., $P_{ji} : V_j \rightarrow H_i$).

C. Constraints

1) Unique placement of virtual machines

A virtual machine $V_j$ can be located to run on a single physical machine $H_i$ or alternatively, it could be not located in any physical machine if it is not critical. Consequently, this restriction is expressed as:

$$\sum_{i=1}^{n} P_{ji} \leq 1 \quad \forall j \in \{1,2,\ldots,m\}$$

where:

- $n$: Number of physical machines $H_i$
- $P_{ji}$: Binary variable equals 1 if the virtual machine $V_j$ is located to run on the physical machine $H_i$; otherwise, it is 0
- $m$: Number of virtual machines $V_j$

2) Service Level Agreement (SLA) provision

A virtual machine $V_j$ with critical SLA (i.e. $SLA_j = 1$) must necessarily be located to run on a physical machine $H_i$. Consequently, this restriction is expressed as:

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

where:

- $SLA_j$: Service Level Agreement $SLA_j = 1$ if $V_j$ is critical, or 0 otherwise.

3) Resource capacity of physical machines

A physical machine $H_i$ must have sufficient available resources to meet the requirements of all virtual machines $V_j$ that are located to run on $H_i$. Consequently, this set of constraints can be mathematically formulated as:
∀ i ∈ \{1,2,...,n\} i.e., for all the physical machines H_i.

where:

- $V_{cpu_j}$: Processing requirements of the virtual machine $V_j$
- $H_{cpu_i}$: Processing resources of the physical machine $H_i$
- $V_{ram_j}$: Memory requirements of the virtual machine $V_j$
- $H_{ram_i}$: Memory resources of the physical machine $H_i$
- $V_{hdd_j}$: Storage requirements of the virtual machine $V_j$
- $H_{hdd_i}$: Storage resources of the physical machine $H_i$

IV. OBJECTIVE FUNCTIONS

A virtual machine placement problem can be defined as a Multi-Objective Optimization Problem (MOP), using the notation defined in Section III, and considering the simultaneous optimization of various objectives function. This work proposes the optimization of the following objective functions [8]:

A. Energy Consumption Minimization

Wang et al. [26] presented equation (13) to calculate the energy consumption per unit of time, represented by the sum of the energy consumption of each physical machine $H_i$.

$$E = \sum_{i=1}^{n} (\rho_{max_i} - \rho_{min_i}) \times U_{cpu_i} + \rho_{min_i}$$  \hspace{1cm} (13)

Inspired in equation (13), López et al. have recently proposed in [8] equation (14) that contemplates shutting down some physical machines $H_i$ that are not needed to run any virtual machine, to minimize energy consumption.

$$E = \sum_{i=1}^{n} ((\rho_{max_i} - \rho_{min_i}) \times U_{cpu_i} + \rho_{min_i}) \times Y_i$$  \hspace{1cm} (14)

where:

- $E$: Total energy consumption per unit of time [W] of the physical machines
- $\rho_{max_i}$: Maximum power consumption of the physical machine $H_i$
\( \rho_{\text{min}}^i \): Minimum power consumption of the physical machine \( H_i \). According to [25], \( \rho_{\text{min}}^i \sim \rho_{\text{max}}^i \cdot 0.01 \)

\( \text{Ucpu}_i \): Utilization ratio of processing resources (CPU) used by the physical machine \( H_i \)

\( Y_i \): Binary variable equals 1 if the physical machine \( H_i \) is turned on; otherwise, it is 0.

B. Network Traffic Minimization

Shrivastava et al. [25] proposed the minimization of network traffic among virtual machines by maximizing locality. Therefore, López et al. have recently proposed equation (15) to estimate network traffic represented by the sum of the traffic generated by each virtual machine \( V_j \), that is located to run on a physical machine \( H_i \), with other virtual machines \( V_k \) that are located to run on a different physical machine \( H_q \) (\( i \neq q \)) [8].

\[
T = \sum_{j=1}^{m} \sum_{k=1}^{m} (T_{jk} \times D_{jk})
\]

(15)

where:

\( T \): Total network traffic [Kbps]

\( T_{jk} \): Average rate of communication in [Kbps], between the virtual machine \( V_j \) and the virtual machine \( V_k \)

\( D_{jk} \): Binary variable that equals 1 if \( V_j \) and \( V_k \) are located in different physical machines; or 0 otherwise.

Two virtual machines \( V_j \) and \( V_k \) which are located on the same physical machine \( H_i \) do not contribute to increase the total network traffic \( T \) given by equation (15); therefore, \( D_{jk} = 0 \) when \( V_j \) and \( V_k \) are run on the same computer [8].

C. Economical Revenue Maximization

López et al. have recently proposed in [8] equation (16) to estimate the total economical revenue that a Data Center receive for meeting the requirements of its customers, represented by the sum of the economical revenue obtainable by each virtual machine \( V_j \) that is effectively located for execution on a physical machine \( H_i \).

\[
R = \sum_{j=1}^{m} (R_j \times X_j)
\]

(16)

where:

\( R \): Total economical revenue [$]

\( R_j \): Economical revenue obtained for locating virtual machine \( V_j \) for execution

\( X_j \): Binary variable that equals 1 if \( V_j \) is located for execution on a physical machine \( H_i \); or 0 otherwise.
V. MULTI-OBJECTIVE MEMETIC ALGORITHM

This work proposes a memetic algorithm [17] for solving the VMP problem in a multi-objective context considering the mathematical formulation presented in Section III to optimize the three objective functions presented in Section IV.

A. Chromosome representation

Jing and Fortes proposed a chromosome representation (denoted as C) for the VMP problem [23]. This representation seems to have several disadvantages; consequently, Lopez et al. proposed a new chromosome representation in [8] using a vector representation of dimension \( m \), i.e., \( C = [C_1, C_2, ..., C_m] \), \( C_k \in \mathbb{N} \), where \( C_k \) represents the physical machine \( H_i \) in which the virtual machine \( V_k \) is located for execution. If \( C_k = 0 \), the virtual machine \( V_k \) has not been located to run on any physical machine; therefore, there will not be any economical revenue for this virtual machine \( V_k \) when calculating the objective function \( R \), given by equation (15). This chromosome representation simplifies the data structure for programming as well as objective function evaluation. The chromosome representation proposed in [8] (used in this work) is presented in Fig 1.

Each chromosome \( C \) can be uniquely mapped to a solution \( P \) (described in Section III), as illustrated in the following example.

![Fig. 1. Chromosome representation proposed by López et al. [8].]

**Example 1**

Chromosome \( C = [1, 1, 1, 2, 2, 2, 2, 3, 3] \) shown in Fig. 1 is represented in (17) using the mathematical notation described in Section III for the solution matrix \( P \).

\[
\text{Solution } P = \begin{bmatrix}
1 & 0 & 0 \\
1 & 0 & 0 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 1 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
0 & 0 & 1 
\end{bmatrix} \tag{17}
\]

It is important to remember that:

- If \( C_j = 0 \), then \( P_{ji} = 0, \ \forall \ i \in \{1, 2, ..., n\} \), i.e., the virtual machine \( V_j \) is not located for execution on any physical machine (no economical revenue \( R_j \)).

- If \( C_j = k \), then \( P_{jk} = 1 \), and \( P_{jk'} = 0, \ \forall \ k' \neq k \), i.e., the virtual machine \( V_j \) is located for execution only on physical machine \( H_k \).
B. Pseudo-code

The multi-objective memetic algorithm proposed in the presented work as a technique for solving the virtual machine placement problem is described in Pseudo-code 1. This pseudo-code is based on a standard memetic algorithm presented by Báez et al. in [17]. Basically, the proposed memetic algorithm works as follows: In step 01 it is verified if the problem has at least one solution (considering only critical virtual machines) 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 02, which generates a set of random candidates \( P_0 \), whose solutions are repaired in step 03 to ensure that \( P_0 \) contains only feasible solutions. Then, the algorithm tries to optimize candidates at step 04 using local search. With the obtained non-dominated solutions, the first set \( P_c \) (Pareto set approximation) is generated at step 05. After initialization in steps 06 and 07, the evolution begins (iterations between steps 08 and 16).

The evolutionary process basically follows the same behavior: solutions are selected from the union of \( P_c \) with \( P_t \) (step 09), crossover and mutation operators are applied as usual (step 10), and eventually the solutions are repaired, as there may be no feasible solutions (step 11). Improvements of solutions of the evolutionary population \( P_t \) may be generated at step 12 using local search (local optimization operators); while at step 13 the generations (or iterations) counter is updated. At step 14 the Pareto set approximation \( P_c \) is updated (if applicable), while at step 15 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 in step 17 the set of non-dominated solutions \( P_c \).

C. Solutions initialization

A solution initialization process consist on random generation of a population \( P_0 \) composed of individuals \( C = [C_1, C_2, ..., C_m] \). The possible values that can take \( C_k \) for virtual machines \( V_j \) with noncritical SLA\(_j\) are in the range \([0, n]\). For virtual machines \( V_j \) with critical SLA\(_j\) the possible values are in the range \([1, n]\). Within these ranges defined by the SLA\(_j\) of each \( V_j \), the algorithm ensures at the initialization phase that all virtual machines \( V_j \) with critical SLA\(_j\) will be located for execution on a physical machine \( H_i \), while for virtual machines \( V_j \) with noncritical SLA\(_j\) there is a possibility that they may not be located in any physical machine.

D. Infeasible solutions reparation

With a random generation at the initialization phase (step 02 of Pseudo-code 1) and / or solutions generated by standard genetic operators (step 10 of Pseudo-code 1), it could be obtained infeasible solutions in which the resources required by the virtual machines located for execution on certain physical machines could exceed the available resources.

Repairing these infeasible solutions (lines 03 and 11 of Pseudo-code 1) may be performed in two stages: firstly, in the feasibility verification process, the population is classified in two classes: feasible and unfeasible (Pseudo-code 2). Later, in the process of repairing infeasible solutions (Pseudo-code 3), the solutions which do not meet the feasibility criteria are repaired. Note that at step 03 of Pseudo-code 3, \( V_j \) migration to \( H_i^j \) can be made to other physical machines \( H_i^j \) even if the physical machine \( H_i^j \) is shut down.

E. Local search

Once a population contains only feasible solutions, a local search is performed (step 04 and 12 of Pseudo-code 1) for improving the solutions found so far in the evolutionary population \( P_t \). The local search process is presented in Pseudo-code 4.

For each individual in the population \( P_t \), the proposed algorithm attempts to optimize a solution with a local search (step 02 of Pseudo-code 4). For this purpose, with probability \( \frac{1}{2} \), the algorithm tries to maximize
the number of located virtual machines, locating all possible virtual machines that were not located so far, directly increasing the total economic revenue (step 03-05 of Pseudo-code 4). On the other hand, also with probability $\frac{1}{2}$, the algorithm tries to minimize the number of physical machines turned on, directly reducing the total energy consumption (steps 06-08 of Pseudo-code 4). With the proposed probabilistic local search method, a balanced exploitation of objective functions (revenue and energy consumption) is found.

F. Fitness function

The fitness function used in the proposed multi-objective memetic algorithm is the one presented in [14]. This fitness value defines a nondomination rank in which to each individual it is assigned a value equal to its Pareto dominance level (1 is the highest level of dominance, 2 is the next, and so on). Between two individuals with different nondomination rank, the individual with lower value (higher level of dominance) is considered better. To compare individuals with the same nondomination rank, a crowding distance is defined. The basic idea of the crowding distance is to find the Euclidean distance (properly normalized when the objectives have different measure units) between each pair of individuals, based on the k objectives, in a hyper-k-dimensional space [14]. The individual with larger crowding distance is considered better.

G. Selection and Crossover

The proposed memetic algorithm uses a Binary Tournament method for selecting individuals for crossover and mutation [4].

The crossover method used in the presented work is the single point cross-cut [4]. The selected individuals in the ascending population are replaced by descendant’s individuals.

H. Mutation

This work uses a mutation method in which each gene is mutated with a probability $p = \frac{1}{m}$ where $m$ represents the number of virtual machines.

This method offers the possibility of full gene mutation, with a very low probability (but larger than zero), which is beneficial to the exploration of the search space, reducing the possibility of stagnation in local optimal.

I. Population evolution

The population evolution in the proposed memetic algorithm is based on the population evolution proposed in [14]. A population $P_{t+1}$ is formed from the union of the best known population $P_t$ and offspring population $Q_t$, applying nondomination rank and crowding distance operators.

VI. EXPERIMENTAL RESULTS

To validate the proposed mathematical formulation and the proper functioning of the proposed memetic algorithm for solving the virtual machine placement problem on a purely multi-objective context, all the experimental tests were run with real data of physical machines, virtual machines and traffic network among virtual machines from the Itaipu Technological Park Data Center [www.pti.org.py] in Paraguay. In addition, some data of the virtual machine’s requirements were obtained from global cloud computing providers as Amazon EC2 [aws.amazon.com] and RackSpace [www.rackspace.com].

Experimental tests in the presented work were executed in several scenarios, considering different number of physical and virtual machines. For this, a Scenario Generator was developed to facilitate the generation of different test scenarios. Based on actual data, the developed Scenario Generator scales these data to propose
scenarios with different number of physical machines, virtual machines and network traffic among virtual machines.

To compare the experimental results obtained by the proposed multi-objective memetic algorithm and to validate its proper operation, an exhaustive search algorithm was also implemented for finding all possible solutions of a given instance of the virtual machine placement problem, when this alternative is computationally possible for the authors.


```
01: Check if the problem has a solution
02: Initialize set of solutions $P_0$
03: $P_1 = \text{repair infeasible solutions of } P_0$
04: $P_1'' = \text{apply local search to solutions of } P_0''$
05: Update set of nondominated solutions $P_c$ from $P_0''$
06: $t = 0$
07: While (stopping criterion is not met), do
08: $P_t = P_0''$
09: $Q_t = \text{selection of solutions from } P_t \cup P_c$
10: $Q_t' = \text{crossover and mutation of solutions of } Q_t$
11: $Q_t''' = \text{repair infeasible solutions of } Q_t''$
12: $Q_t''' = \text{apply local search to solutions of } Q_t''$
13: increment $t$
14: Update set of nondominated solutions $P_c$ from $Q_t'''$
15: $P = \text{fitness selection from } P_t \cup Q_t''$
16: End while
17: Return Pareto set approximation $P_c$
```

Pseudo-code 2. Feasibility verification.

```
01: While (there are solutions not verified), do
02: feasible = true
03: $i = 1$
04: While ($i < n$ and feasible = true), do
05: If ($H_i$ doesn’t have enough resources to attend all the virtual machines assigned)
06: feasible = false
07: Else
08: increment $i$
09: End if
10: End while
11: If (feasible = false)
12: repair solution
13: End if
14: End while
15: Return population of feasible solutions only
```

Both algorithms were implemented using ANSI C programming language (GNU C Compiler) and experimental tests were executed on an Ubuntu 11.10 Linux Operating System, with an Intel Core i7 1.2 GHz processor and 8 GB of RAM memory.

The exhaustive search algorithm generates $(n+1)^m$ possible solutions when there is no critical virtual machine [8]. However, if there are $c$ critical machines, where $c \in \{1, 2, ..., m\}$, the exhaustive search algorithm only generates $N$ possible solutions:

$$N = (n + 1)^m - \sum_{r=m-c}^{m-1} \sum_{s=0}^{c} (n + 1)^r \times n^s$$

(18)
where  
\( c: \) number of critical virtual machines, \( c \in \{1, 2, \ldots, m\}. \)

Pseudo-code 3. Infeasible solutions reparation.

01: feasible = false, \( j = 1 \)
02: While \( (j < m \text{ and } \text{feasible} = \text{false}) \), do  
03: \( \text{If} \) (SLA\(_j\) is critical)
04: \( \text{If} \) (it is possible, migrate \( \text{V}_j \) to \( \text{H}_{i'} \) (\( i' \neq i \)))
05: \( \text{Else if} \) (it is possible turn off \( \text{V}_j \)’ on \( \text{H}_i \) (\( i' \neq j \) and SLA\(_j\)’ is noncritical))
06: \( \text{Else if} \) (it is possible, turn off \( \text{V}_j \) on \( \text{H}_{i'} \) and try migration of \( \text{V}_j \) to \( \text{H}_{i'} \) (\( i' \neq i \) and SLA\(_k\) is noncritical))
07: \( \text{Else if} \) (it is possible, turn on \( \text{H}_{i'} \) and try migration of \( \text{V}_j \) to \( \text{H}_{i'} \) (\( i' \neq i \)))
08: \( \text{Else} \) (replace the solution with other solution from the Pareto set \( P_c \))
09: \( \text{End if} \)
10: \( \text{Else} \)
11: \( \text{If} \) (it is possible, migrate \( \text{V}_j \) to \( \text{H}_{i'} \) (\( i' \neq i \)))
12: \( \text{Else} \) Turn off \( \text{V}_j \)
13: \( \text{End if} \)
14: \( \text{End if} \)
15: increment \( j \)
16: \( \text{End while} \)
17: Return feasible solution

Pseudo-code 4. Local search.

01: probability = random number between 0 and 1
02: For (all physical machines), do  
03: \( \text{If} \) (probability < 0.5)
04: Try to turn off all the possible \( \text{H}_i \) by migrating all the \( \text{V}_j \) assigned to \( \text{H}_i \)' with available resources (\( i' \neq i \)).
05: Try to turn on all the possible \( \text{V}_j \) assigning them to a \( \text{H}_i \) with available resources
06: \( \text{Else} \)
07: Try to turn on all the possible \( \text{V}_j \) assigning them to a \( \text{H}_i \) with available resources
08: Try to turn off all the possible \( \text{H}_i \) by migrating all the \( \text{V}_j \) assigned to \( \text{H}_i \)' with available resources (\( i' \neq i \)).
09: \( \text{End if} \)
10: \( \text{End for} \)
11: Return optimized population

These results were compared to the results obtained by the proposed memetic algorithm after evolving populations with 100 individuals, for 100 generations.

A. Experimental Test 1

The following test scenarios were generated through the developed Scenario Generator for this experimental test:

- 3x5x50: 3 physical machines and 5 virtual machines with 50% of critical SLA.
- 4x10x50: 4 physical machines and 10 virtual machines with 50% of critical SLA.
For each test scenario (3x5x50 and 4x10x50) one run of the exhaustive search algorithm was completed, obtaining the optimal Pareto front and its corresponding Pareto set. Furthermore, ten runs of the proposed memetic algorithm were completed, after evolving populations of 100 individuals for 100 generations at each run. The results obtained by the proposed memetic algorithm for each run were combined to obtain the Pareto front and its corresponding Pareto set.

For the 3x5x50 scenario, the proposed memetic algorithm obtained every solution of the optimal Pareto set and its corresponding Pareto front. For the 4x10x50 scenario, the proposed memetic algorithm obtained 100% of the optimal Pareto front but only 93.75% of the corresponding Pareto set.

B. Experimental Test 2

The following test scenario was generated through the developed Scenario Generator for this experimental test:

- 10x20x50: 10 physical machines and 20 virtual machines with 50% of critical SLA.

For this test scenario (10x20x50) one run of the exhaustive search algorithm was not successfully completed, because it is impracticable to generate all possible solutions with the available resources. Clearly, for instances of the problem of the mentioned dimension or superior, it is necessary to implement alternatives to exhaustive search such as the meta-heuristic proposed in the presented work.

Consequently, ten runs of the proposed memetic algorithm were completed, after evolving populations of 100 individuals for 100 generations at each run. The results obtained by the proposed memetic algorithm at each run were combined to obtain a Pareto front approximation and its corresponding Pareto set. In this case, the Pareto set and the Pareto front contain 48 solutions.

Clearly, the limited scalability of the exhaustive search was verified while the benefits of using bio-inspired meta-heuristics such as the memetic algorithm proposed in the presented work was demonstrated.

C. Experimental Test 3 (effects of critical virtual machines)

The following test scenarios were generated through the developed Scenario Generator to study the effects of critical virtual machines in the resolution of VMP problems:

- 6 scenarios of 3x5x[%]: 3 physical and 5 virtual machines with [0, 20, 40, 60, 80, 100] % of critical SLA.
- 6 scenarios of 4x10x[%]: 4 physical and 10 virtual machines with [0, 20, 40, 60, 80, 100] % of critical SLA.

As an example for a better understanding of the notation used to resume the scenarios for this experimental test, the 3x5x80 scenario represents one of the six scenarios of 3 physical machines and 5 virtual machines with 80% of critical SLA. Analogously, the 4x10x10 scenario represents one of the six scenarios of 4 physical machines and 10 virtual machines with 10% of critical SLA. The available resources of each physical machine and the requirements of each virtual machine are the same for all the six scenarios of 3 physical machines and 5 virtual machines. The only difference among scenarios is the percentage of virtual machines with critical SLA.

For the six test scenarios of 3 physical machines and 5 virtual machines (3x5x00, 3x5x20, 3x5x40, 3x5x60, 3x5x80 and 3x5x100) and for the six test scenarios of 4 physical machines and 10 virtual machines (4x10x00, 4x10x20, 4x10x40, 4x10x60, 4x10x80, 4x10x100) one run of the exhaustive search algorithm was completed, obtaining the number of all feasible solutions for each test scenario. Fig. 2 shows a summary of the number of feasible solutions. The horizontal axis represents the percentage of virtual machines with critical SLA for the scenarios of 3x5x[%] (3 physical machines and 5 virtual machines with the mentioned percentage of critical SLA) and also for the scenarios of 4x10x[%] (4 physical machines and 10 virtual machines with the corresponding percentage of critical SLA). The vertical axis represents the number of all
feasible solutions of the scenario with the respective percentage of virtual machines with critical SLA. In this experiment, the behavior of the number of feasible solutions showed in Fig. 2 for both principal scenarios (3x5 and 4x10) considering each scenario with different percentage of virtual machines with critical SLA is validated using a data correlation analysis. The correlation for the scenarios of 3x5 and 4x10 are -0.92 and -0.99 respectively.

A negative correlation close to 1 clearly indicates a high degree of dependence between the variables, in this case the number of all feasible solutions and the percentage of virtual machines with critical SLA. Considering the correlation results (-0.92 for 3x5 scenario and -0.99 for the 4x10 scenario), incrementing the percentage of virtual machines with critical SLA in the 3x5 scenario will decrement the number of all feasible solutions.

In addition of the execution of the exhaustive search algorithm described above, five runs of the proposed memetic algorithm were completed after evolving populations of 100 individuals. The stopping condition used was the calculation of at least 95% of all possible solutions found by the exhaustive search algorithm for each of the test scenarios. The execution time of each run was averaged to obtain the average execution time of the proposed algorithm for each test scenario with its corresponding percentage of virtual machines with critical SLA. Fig. 3 shows a summary of the average execution time of the proposed algorithm for each test scenario with different percentage of virtual machines with critical SLA. In Fig. 3, the horizontal axis represents the percentage of virtual machines with critical SLA for the different scenarios while the vertical axis represents the average execution time of the proposed algorithm.

In this experiment (see Fig. 3), the execution time decreases with the percentage of virtual machines with critical SLA for both tested scenarios (3x5 and 4x10). The correlation for the scenarios of 3x5 and 4x10 are -0.96 and -0.78 respectively; therefore, incrementing the percentage of virtual machines with critical SLA will decrement the execution time, what seems completely consistent with equation (18).

![Fig. 2. Summary of the number of all feasible solutions for Experimental Test 3, with different percentage of virtual machines with critical SLA.](image)

VII. CONCLUSIONS AND FUTURE WORK

This work proposed a novel formulation of the *virtual machine placement* problem in a purely multi-objective context, considering the criticality of applications given by their SLA, for the three main objective functions studied in Lopez and Barán [7]. A first formulation considering these three objectives, in a purely multi-objective context, was proposed in [8]. Thus, this work extends that previous work to a more advanced formulation that contemplates possible dependencies between applications running on each virtual machine. In this case, a simple Service Level Agreement (SLA) giving a level of criticality indicates which virtual
machine should be necessarily located for execution to ensure correct operation of mission critical applications.

![Figure 3: Summary of the execution time in seconds for Experimental Test 3 with different percentage of virtual machines with critical SLA.](image)

To solve this new formulated VMP, a novel multi-objective memetic algorithm was proposed. To validate the proposed algorithm, it was run with several different scenarios 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 (100%) for the 2 instances of the Experimental Test 1.

The scalability of the proposed algorithm was also verified for scenarios with a considerable amount of physical and virtual machines (Experimental Test 2), where the exhaustive search algorithm could not complete one run because of the huge number N of possible solutions, given by equation (18).

At the same time, this work verified that by increasing the percentage of virtual machines with critical SLA, the number of feasible solutions decrements. Consequently, increasing the percentage of virtual machines with critical SLA also decreases the execution time required by the proposed algorithm to find at least 95% of feasible solutions, because the search space becomes smaller, confirming equation (18).

At the time of this writing, the authors are also considering alternative formulations for the problem, considering more SLA levels and other objective functions studied in the taxonomy proposed in [7], as well as testing other bio-inspired meta-heuristic that have not yet been tested for solving the multi-objective virtual machine placement problem given the novelty of the proposed context.

VIII. ACKNOWLEDGMENT

To Itaipu Binacional and the Itaipu Technological Park for giving me the opportunity of studying this master course.

To the Professor Benjamin Barán, for guiding this thesis with such dedication, enthusiasm and constant contributions. Working with him is certainly an enriching experience.

To my classmates, teachers and colleagues for giving me confidence and support throughout the development of this work.
To my family, for their constant support in every endeavor of my life.

IX. REFERENCES


