# Toward a Framework for VM organisation based on Multi-Objectives

Guilherme Arthur Geronimo
Federal University of Santa Catarina
Florianópolis, Brazil
guilherme.geronimo@ufsc.br

Rafael Brundo Uriarte
IMT Institute for Advanced Studies
Lucca, Italy
rafael.uriarte@imtlucca.it

Carlos Becker Westphall
Federal University of Santa Catarina
Florianópolis, Brazil
carlos.westphall@ufsc.br

***Abstract*—This paper proposes a flexible framework to improve the quality of Virtual Machine's placements, in Clouds. It organises them by relocating the VMs based on the Multiple-Objectives of the environment. These Objectives are represented by Rules, Qualifiers, and Costs, which can be extended and prioritised. Based on Evolutionary Searches, the framework theoretically guarantees the adoption of a better set of Placements. More specifically, it seeks the non-dominated solutions (Pareto's Dominance concept) and compares then considering the implementation cost of the scenario and its benefits. In contrast to existing solutions that address specific objectives, our framework was devised to support many types of objectives and to be easily extensible, which enables the implementation of new and generic prioritised elements. Moreover, we conducted experiments using data from a real Cloud environment and show the flexibility of our approach and its scalability.**

***Keywords*—*Virtual Machine Placement; Cloud Computing; Multi-Objective Optimisation.***

## I. Introduction

Although Cloud Computing (CC) can bring many benefits to consumers and providers, Cloud's mismanagement usually accentuates problems related to waste of resources. For instance, performance degradation due to noisy neighbours rise of thermal hotspots on data centre and shortage of resources due constant migrations [1][2].

Many approaches were proposed to mitigate this problem, such as Simulation-Based, Policy-Based, Bin Packing and Evolutionary Algorithms [3]-[5]. However, these proposals usually focus on specific objectives, e.g., on decreasing the energy consumption. This limitation hinders the adoption of these approaches, since Cloud's objectives, policies and priorities vary, and these proposals cannot adapt to these needs.

To address this limitation some works, e.g., [6]-[8], consider Virtual Machine Placement (VMP) problem as a multiple objective (MO) optimisation problem, that, simultaneously, tries to minimise the total resource wastage, power consumption and thermal dissipation costs. This wider view enables managers to consider other facets of VMP, such as internal policies and Service Level Objectives (SLO).

Due to those many facets, to solve MOs conflicts and guarantee a fast convergence, proposals usually adopt Evolutionary strategies that build non-dominated scenarios sets (using the Pareto Dominance concept). In this case, a non-dominated scenario is better in at least one objective and, at the same time, not worse in any other objective, compared to a base scenario. Despite the efficiency, this strategy stagnates in environments with many objectives, i.e., possibly reaching a state where none result is good enough for all objectives simultaneously. Moreover, the selection of the best scenario is also a challenge due the limited evaluation's strategies, their execution time and cost. Besides, none of the existing solutions consider, at the same time, the important Cloud's characteristics: SLO, SLA, policies and costs.

Consequently, these aspects need to be represent and implement in a standard manner, a model to solve VMP problems. Despite the fact of many works address this issue, none of them aimed at comprehend agnostic-objective methods, focusing in specific and limited objectives. Thus, to the best of our knowledge, no suitable model meets our needs to represent different types of evaluations, quantifiable and qualifiable.

Nonetheless, the VMP problem can be divided in sub-issues, such as Provisioning, Organisation and Dynamic Allocation of VMs. Even though, the environment objectives are always the same, regardless which sub-issue are being solved. However, the majority of the proposals treat them separately and differently – in terms of objectives and strategies.

Considering these limitations, we designed an easily extensible framework to address the VMP organisation problem. This framework adopts a flexible approach that enables the assessment and comparison of a single placements to the whole clusters, enabling evaluations with grater precision. Moreover, it uses MO qualification functions to provide a placement to new Virtual Machines (VMs) or select and relocate VMs to non-dominated scenarios. After filtering the possible scenarios, it chooses the best result regarding its benefits and implementation costs.

This framework takes advantage of constraints and SLAs to reduce the computation cost, enabling a fast local-optimal search. The main contributions of this framework are: (i) the support to generic multiple objectives and (ii) objective prioritisation. The rest of this paper is organised as follows: Section II provides the background concepts and the related works; Section III define the Cloud model; Section IV presents the methods and their designs; Section V describes the framework's tests, implementation and results; Section VI concludes the article addressing the future works and open issues.

## II. Background and Related Works

The first part of this section presents important concepts used to discuss the related works in the second section.

### A. Background

*Extract and Relocate* is an approach to select and migrate VMs (one or more) and is employed to improve the overall state of a cloud. The selection of VMs is commonly performed using heuristics or triggers, which normally attempt to choose the special VMs, according to some criteria, and migrate these.

The Pareto's Dominance concept can be used to filter the many possibilities of Relocation. The Pareto Set is a set of non-dominated solutions, i.e., scenarios that are better in at least one objective and, at the same time, not worse in any other objective compared to a base scenario [9]. Let $S_1 = (p_1, ..., p_y)$ and $S_2 = (q_1, ..., q_y)$ be two scenarios with $y$ placements evaluations, $S_1$ is said to *dominate* $S_2$ if $p_i \geq q_i$ for all $i = 1, ..., y$ and $S_1 \neq S_2$. Figure 1 graphically represents this evaluation considering two objectives. The scenarios marked with a $(\star)$ are the non-dominated solutions and the $(\otimes)$ are dominated by their successors.

In order to build the set of possible scenarios, the *MO Variable Neighbourhood Search* [10] uses the concepts of *Solution Dominance* to explore the neighbour possibilities. Thus, instead of searching all possible scenarios, the approach explores only the non-dominated ones, building a Pareto Set with the dominant solutions found. However, exploring only the non-dominated solutions restricts searching space. Therefore, they vary the neighbourhoods restarting the search from different random scenarios. The union of the Pareto's Sets forms the Pareto Front set, i.e., all the $(\star)$ in Figure 1.

Furthermore, the selection of a scenario from the Pareto Front is not a deterministic choice, depending on the problem's nature and the applied model. There are a wide variety of approaches that could be adopted for VMP, for instance, Zitzler et al. [11] compared twenty-two generic assessments that could be used in MOPs, such as number of fulfilled goals, distance from base scenario and error ratio.

Despite of the variety, those indicators do not consider the cost to implement a scenario, which is an important decision aspect in VMP. Usually, the *Costs* are treated as a consequence of energy consumption or as another assessment to be minimised [5][12][13]. However, Cost is any commodity that Cloud Providers are willing to spend in order to achieve an amount of benefit. Its nature is a variable aspect in kind – such as time, computational resources or money – and it should be part of the decision process.

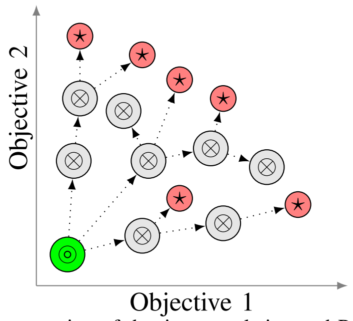

Figure 1. Representation of dominance relation and Pareto Frontier

### B. Related Works

Most of the works regarding VMP focus on specific issues, such as a set of services [13], for network optimisation [14][15] and energy consumption [4][16][17]. Such methods have limited applicability and, therefore, are not considered in this work.

Nevertheless, some proposals adopt approaches which could be adapted and improved to manage generic MOs. For instance, the following proposals adopt the *Extract and Relocate* approach to solve their specific VMP problems. Beloglazov et al. [4] proposes four heuristics to select a VM to be extracted based on the VM's load, the number of migrations, growth potential and random choice. Then, relocate them using a variation of Best Fit Decreasing algorithm (MBFD) to map VMs into Physical Machines (PMs), seeking to reduce power consumption. The Best Fit Decreasing (BFD) algorithm inserts *items*, in *bins*, sorted in decreasing order of size. After, chooses the bin that will provide the minimum empty space after the item is inserted.

On the other hand, Xu et al. in [18], involves three strategies to *Extract* VMs: (i) find the VM with the greatest CPU Load and less memory, to reduce migration overhead (ii) identify the VMs that are competing for scarce resources and (iii) check the possibility to remove every VM and shut down the PM. Then, the extracted VM is migrated to the PM which provides the greatest utility increase. Despite the fact that they consider different strategies during the extraction phase, their method is still bounded and, consequently, limited to those specific objectives.

Contrariwise, to avoid theses dependences and limitations, our proposal aims to be flexible enough to enable the usage of any kind of objective, strategy or affinity, by representing them as Qualifiers. Then, our Framework uses those Qualifiers to Provide placements to new VMs and Organize the Cloud.

Likewise, other works also propose frameworks to generalise and/or facilitate the studies in this area. However, many were still bounded to specific goals, e.g., [12]. Next, we discuss some proposals, which are the exceptions.

The *Snooze* framework [19] proposes a dynamic hierarchical self-organising structure. It deploys agents in PMs, which have a partial view of the system, in order to make decisions about their own hosted VMs, such as migrate, resize, start, suspend, resume, shut down and destroy them. In this framework, objectives, triggers and scheduling are represented as centralised open defined policies, submitted to the PM agents, where they react according to the triggered events. The fact of divide and distribute the decisions across the PMs, induce the improvement of local scenarios, hoping to improve the whole Cloud. Therefore, it lacks the improvement of the Cloud as a single unit, i.e., lacks a holistic view.

The Cielo Framework [20], aims at aiding Cloud operators to adapt the resource allocation to applications to the operational conditions in a Cloud. It adopts a limited set of strategies to manage CPU allocation, Response Time, Power consumption, workload distribution and Bandwidth allocation. Then, it uses an *Evolutionary Game Strategy* to improve the Cloud. Basically, it treats applications as Players, which can play pre-defined strategies in their turn, to improve specific aspects of themselves. However, it adopts placement changes regardless of its effects across the other elements.

## III. Cloud Model

In this section, we describe a Cloud model for VMP problems and its elements. These elements are the key components of the proposed framework.

In Clouds, a *Placement* is defined as a possible relation between a VM and a PM, guest and host respectively. A *Scenario*, in its turn, is a set of placements, which defines where each VM is placed on the Cloud, representing the real state of the Cloud or a possible one.

As presented in Section I, real Cloud environments have Multiple Objectives. In our model, MOs can be represented by the following elements: Rules, Qualifiers, Priorities and Cost.

*Rules* define placement constraints, i.e., rules can forbid VMs to be placed in PMs. They are divided into two types: Rules Free of Context (RFC) and Rules Sensitive to the Context (RSC). The context refers to a given scenario, which is the context where the VMs are placed. RFC define whether a given VM can be hosted by a given PM and RSC define whether a given scenario is valid or not, i.e., RFC validates placements and RSC validates scenarios. In order to illustrate these rules, consider the following examples: (i – RSC) a VM $A$ should not be hosted in the same PM as a VM $B$, and (ii – RFC) a VM $A$ must be placed in a PM with a specific architecture of processor.

*Qualifiers* are functions used to assess the quality of one or more placements of a Scenario. These assessments will enable the comparison and, consequently, selection of scenarios.

*Priorities* are applied to sort the qualifiers. They define weights and preferences between them, avoiding conflicts.

*Cost* is a function that quantifies the implementation cost between two scenarios, a root and a target scenario. Its values disregard units and can vary from the number of migrations to the required resource to implement the target scenario. Differently from the Qualifiers, it is not limited to a range of values and it depends of two scenarios to process a result.

This model was inspired by the model used in [21], which is based on the Organisation Theory.

TABLE I. SYMBOLS USED IN THE CLOUD MODEL DESCRIPTION

| *Symbol* | *Meaning* |
|---|---|
| $y$ | Number of VMs |
| $x$ | Number of PMs |
| $v$ | VM id |
| $h$ | PM id |
| $V$ | Set of $y$ VMs |
| $H$ | Set of $x$ PMs |
| $(V_a, H_b)$ | A placement |
| $C$ | Set of $y$ VM Placements |
| $rf$ | Rule Free of Context (RFC) |
| $rs$ | Rule Sensible to Context (RSC) |
| $Rf$ | Set of RFC |
| $Rs$ | Set of RSC |
| $q$ | Qualifier Function |
| $z$ | Number of Qualifiers |
| $Q$ | Set of $z$ Qualifiers |
| $U$ | Qualifier's weights |
| $i$ | Cost Function |
| $m$ | Maximum Cost |

### A. Formal Model

To avoid ambiguity and to precisely specify the elements involved in the MO driven Clouds, in this section we formally define the model previously described. Table I presents the symbols used in this section and their meanings.

Let $y$ be the number of VMs and $x$ be the number of PMs in the Cloud. $V$ is a set with $y$ VMs $v$ and $H$ is a set with $x$ PMs $h$, as shown in (1). A Placement is a relation between any two elements from $V$ and $H$, such as $(v_a, h_b)$, and a configuration $C$ is a set with $y$ placements, $\{(v_a, h_b)_1, ..., (v_b, h_c)_y\}$, representing a scenario.

In the formal model, RFC and RSC are represented by two distinct sets of functions, $Rf$ and $Rs$, which contains an amount of $f$ and $s$ boolean functions each, respectively. Given a Placement $(v_a, h_b)$, a function $rf$ returns 1 for permitted or 0 otherwise , as shown in (2). Given a Configuration $C$, a function $rs$ returns 1 for permitted or 0 otherwise, as in (3).

Qualifiers, instead, are presented by $Q$, which is a set with $z$ functions $q$, $Q = \{q_1, ..., q_z\}$. Each function $q$, given any Configuration $C$, returns a vector with $y$ qualifications $i$, one for each placement in $C$, as given by (4). These qualifications are mapped between zero and two, excluding zero, i.e., $]0, 2]$. Zero is not included because a qualifier with this value is a RSC, which forbids a scenario.

To sort the qualifiers, there is a vector $U$ referring to a set of weights, which are rational numbers equal or grater than zero, that prioritise the Qualifiers, as in (5).

Finally, $i$ refers to Cloud's Cost Function, which given any scenario ($C$), returns a rational value grater than zero, related to the implementation Cost of the scenario $C$, as given by (6). Still, exists a number $m$ that defines the maximum cost that the Managers are willing to spend.

$$\begin{aligned} V &= \{v_1, ..., v_y\} \\ H &= \{h_1, ..., h_x\} \end{aligned} \tag{1}$$

$$\begin{aligned} Rf &= \{rf_1, ..., rf_f\} \\ rf((v_a&, h_b), \{0 \vee 1\}) \end{aligned} \tag{2}$$

$$\begin{aligned} Rs &= \{rs_1, ..., rs_s\} \\ rs(C&, \{0 \vee 1\}) \end{aligned} \tag{3}$$

$$\begin{aligned} q(C, \{p_1, ..., p_y\}) \wedge \\ \forall p \in ]0, 2] \end{aligned} \tag{4}$$

$$\begin{aligned} U = \{u_1, ..., u_z\} \wedge \\ \forall u \in (\mathbb{Q} \geq 0) \end{aligned} \tag{5}$$

$$\begin{aligned} i(C, \{p_1, ..., p_y\}) \wedge \\ \forall i \in (0 > \mathbb{Q} > m) \end{aligned} \tag{6}$$

## IV. Framework Design

This section describes the solutions proposed for VMP problems. Initially, we describe the steps of the organisation method, presenting also a use case. Then, we explain the provisioning method, which also uses the previous method. Finally, we discuss some aspects and peculiarities of our approach.

Considering the presented Cloud model, VMP methods must: (i) *maximises* the qualifier's evaluations and (ii) *minimise* the implementation cost. Our solution uses a recursive process, which receives the current state of the Cloud (current scenario) and seeks better scenarios by making migrations that increase its evaluations.

### A. The Organising Method

This section explains in details each step of the developed methodology, while Figure 2 illustrates it.

*Initiating*: The method receives: (i) the current scenario, i.e., a vector with the real placements of the Cloud, (ii) an empty VM set to control recursions (*ignoreVms*) and (iii) a boolean indicator to sign recursions (*isMainRecursion*), starting as *true*. At the beginning, if all VMs were explored, it ends the recursion selecting the current scenario.

*Placements Evaluation*: The first step evaluates the received scenario and generate a VM's rank according to its assessments. Each Qualifier evaluates all placements, building an evaluation matrix $ES_{z,y}$. Then, each evaluation is raised to the power of its respective qualifier's weight. Afterwards, the set is reduced to a vector $E_y$ by multiplying all evaluations of each VM. The reduction is represented by (7).

$$E_y \leftarrow \prod_{n=1}^{z} ES_{n,y}^{U_n} \tag{7}$$

*VM Rank Selection*: This step is based on the *Extraction* strategy and its heuristic selects the VM which: (i) has the lowest evaluation and (ii) is not present in the $ignoreVms$ set.

*Generation of Non-Dominated Scenarios*: The objective of this step is to identify where the selected VM could be migrated, and generate a scenario for each possibility, i.e., at most $x - 1$ scenarios. During this step, the following filters are applied: (i) the Rules (RFCs and RSCs, respectively), (ii) Max Cost filter, checking if the Scenario's cost is below the maximum, and (iii) a Pareto filter, which excludes dominated scenarios.

*Search for new Scenarios*: This step, based on the *Neighbourhood Search*, starts a new recursion to explore each Non-Dominated scenario found. However, to avoid loops the following information is sent with it: (i) a boolean signing *false*, meaning that is not the main recursion and (ii) a copy of the *ignoreVMs* set (*newIgnoreVMs*) and added the lowest VM, which was selected on the first step. The $newIgnoreVMs$ is used to avoid deadlocks during the Search.

*Select Best Scenario*: The goals of this step is to select the scenario which offers the best Cost-Benefit, also considering the current scenario among the recursion's results.

To achieve the equilibrium between the implementation cost and the qualifier's evaluation, we propose the Cost-Benefit method represented by (8). The benefit is defined as the difference between the evaluations of the current scenario of the Cloud (Real Scenario – $RS$) and a target scenario of the Cloud (Possible Scenarios – $PS$). More specifically, is the sum of the variations of each VM evaluation of $RS$ and $PS$. Since $PS$ is a non-dominant solution, it is not possible to have a negative variation. Then, the cost-benefit ($cb$) is calculated by the division of the benefit with the result of the cost function.

$$cb = \frac{\sum_{n=1}^{y} PS_n - RS_n}{i(PS)} \tag{8}$$

*Return Decision*: Finally, this step decides either to continue the search or return the selected scenario. If it is not the main recursion this step will return the best scenario, which was selected. Otherwise, it will *vary* the *Neighbourhood* by starting a new recursion with a non-empty set of VMs to be ignored. In this case, it will add the lowest placement of the selected scenario in the $ignoreVms$ set and starts a new recursion using this scenario and this VM set. The result of this recursion is the method's output. Lines 14-19 of Alg. **??** show this step.

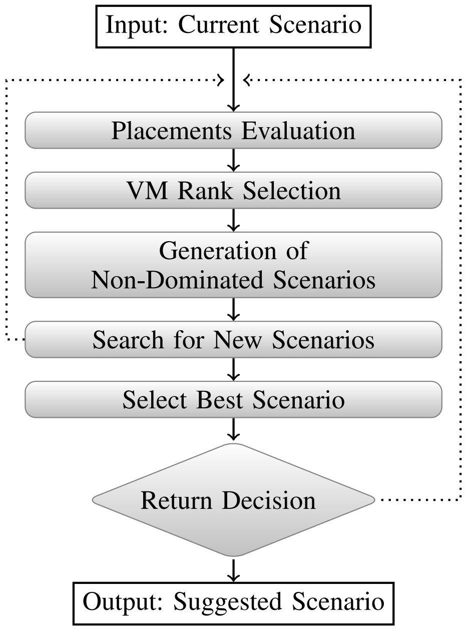

Figure 2. Flow Chart of the Organising Method

### B. Provisioning for New VMs

The objective of this feature is to choose the best placement for a new VM. In this case, the best placement would be the one which provide the greatest benefit. This feature takes advantage of the Organising method to select the placement.

The first step is, based on the current scenario, to adopt a theoretical placement in the first allowed PM. Then, from a set with all VMs of the Cloud, the newcomer is extracted from it, in order to generate the set of VMs to be ignored. Before start the recursion, the Cost function is deactivated, meaning that there is no cost involved in the instantiation. After, an Organising recursion is started sending the theoretical scenario and the built set, as initial variables. The result of the recursion contains the placement of the new VM.

### C. Discussion

In this section we discuss some aspects and differences regarding other proposals and implementation issues.

*The Cloud initial consistency*, with the environment Rules, is demanded for the proper work of the method, which means that the current state of the Cloud must be in accordance with the RFCs and RSCs. The adoption of an initial illegal scenario interferes with the generation of new scenarios. For this, a pre reorganisation process must be applied, which is considered out of the scope of this work.

*Adaptations* of Variable Neighbourhood Search (VNS) and Pareto's Dominance were adopted in this proposal. The VNS strategy builds a Pareto Front by recursively searching Non-Dominated results from different neighbourhoods. However, instead of using random scenarios to vary the searched Neighbourhood, as proposed in [10], we use a method based on the proposed VM Rank to vary the starting point of the search.

Likewise, the original definition of Pareto's Dominance compares the objectives assessments separately. We, instead, compare the placements' evaluations to avoid the method's stagnation due to quantity of objectives. Our judgement is based on the scenario's evaluations, which is a combination of all its qualifiers assessments and its weights.

*The ignore VMs set* is used mainly to avoid deadlocks. A deadlock occurs when the lowest VM is always the same, i.e., their evaluation growth depends on the relocation of other VMs. Thus, the next recursion will ignore this VM and will try to explore other placements.

Although its use together with the *Dominance* Search ensures the convergence of the method, it does not assure finding the general best scenario. For instance, cases where the temporary adoption of a degraded (dominated) scenario is needed to achieve the optimal one. However, it always leads to a better scenario, even if the method is forced to stop before reaching the end of the search.

Our current efforts are to focus on identify and test new deadlock scenarios, testing new approaches in that specific subject, such as: (i) using two independent $ignoreVMs$ sets; (ii) changing from Depth to Wide Search, pushing the Pareto Front forward; and (iii) using a Relaxation Factor strategy in the Pareto filter, enabling the acceptance of dominated results with great benefits.

*Qualifiers based on Monitoring* data, in some cases, are demanded to evaluate placements. However, Evolutionary algorithms compute many scenarios per second, which makes it infeasible to depend on real-time data. For that reason, it is recommended to pre-load the needed data and/or utilise approximation functions in the Qualifiers, if needed.

*Provisioning Gaps*: Despite the proposed strategy works in practically all cases, it's based on the fact that there are resources available during the first phase. We acknowledge this weakness, however, we consider very unlikely to happen, once a certain resources waste are maintained in the PMs.

*The Max Cost* constraint is one of the main convergence catalyser of the method, i.e. it bounds many unaffordable branches during the exploration, considerably decreasing the number of scenarios and accelerating the search. However, the use of this constraint should be based on the premiss that the cost does not decrease from a base scenario to its successors. Otherwise, valid branches could be discarded from the search.

## V. Experiments

In this section, we present the experiments to assess some aspects of our proposal. To this aim, we use real placement data from the main data centre of the Federal University of Santa Catarina. This environment is composed of: 607 VMs, 18 PMs, 99 storage pools and 102 networks.

The experiments were divided into three categories: (i) *Performance*, (ii) *Selection* and (iii) *Filters*.

In *Performance*, we measure the necessary time to evaluate the real scenario and propose better placements. We vary the number of VMs (between 350 and 600) and the maximum number of migrations allowed (5, 10 and 20).

In *Selection*, to understand the advantages of the Rank Selection approach, which improves the worst placements of the scenario, we compare it to: (i) a method based on [20], called Turn Selection, which shuffles the VMs and tries to improve the VMs in turns, and (ii) a method called Highest Selection, which selects VMs on top of the Rank to relocation.

In *Filters*, to show the importance of the Dominance filter, i.e., a method that accepts only scenarios in which not a single placement's evaluation worsen and at least one improves in comparison to the previous scenario, we confront it to a filter that accepts scenarios which evaluations' sum are greater than the previous scenario, named Greater Benefit. The tests were executed in a scenario with 600 VMs and the migration's threshold were varied from 6 to 14.

For the experiments we implemented and employed the following rules: (i) pre-requirements for Live Migration, such as network and storage accessibility; (ii) resource availability for the migration; (iii) the cluster coherence for the migration, i.e., if the VMs are in the same PMs' cluster where they belong; and (iv) the implementation cost cannot exceed the limit. For the qualifiers, the following strategies were implemented: (i) Consolidate VMs in few PMs; (ii) Distribute nodes from the same services in different PMs; (iii) Distribute VMs of the same storage pool. For the Cost evaluation, we employed a function that regards the number of migrations necessary to achieve a given scenario.

The prototype was implemented in PHP 5.5 and the tests use the Test's Framework PHPUnit 4.2. The tests were executed in an environment with Ubuntu 14.04, an *Intel T9400* processor of 2.53GHz with 4GB of RAM. Which, in average, processed 200 scenario per second.

### A. Implementation and Results Discussion

*Performance*: The worst case scenario, considering 600 VMs and 20 migrations, took 62 seconds to finish. The variation of the migration's threshold presented a greater impact when compared with the scenario's size. It considerably increases the Cost-Benefit Rate (+104%), number of analysed scenarios and, consequently, on the execution time (+7355%). Still, a linear pattern on execution time was noticed, due to this threshold, which forced the premature stop of the method, as shown in Figure 3(a).

*Selection*: On average, the Rank method, in comparison to the other methods, shown a greater Cost-Benefit raise on the lowest VMs, as shown in Figure 3(b). It was, on average, 72% faster than the TopRank method and increased 36% of the Cost-Benefit in the worst case scenario. Although this behaviour was expected, since the Random and TopRank methods do not focus on the lowest VMs, the Rank method still managed to raise 2% of the average Cost Benefit of the worst case scenario, which was not expected.

*Filters*: As expected, when we increased the maximum amount of migrations the Dominance Filter prevented the exponential growth of explorations, still maintaining a similar Average Cost-Benefit (+0,002%) in comparison to the other approach. On the other hand, the Greater Benefit Filter raised exponentially the number of searches turning the execution time impractical, as shown in Figure 3(c). Exponential Regression leads us to an approximation of $y = 0.1832e^{0.47x}$, where $x$ is the number of migrations and y the seconds to execute. Otherwise, Pareto's method give us a growth of $y = 1.76e^{0.196x}$, which is an acceptable for real environments.

In summary, due to the Dominance Filter the scenario's growth was drastically reduced and the Rank approach ensured that the gained benefits were focused on the main issues, the lowest placements. Finally, the performance results showed that overall our method is applicable in real Cloud environments, providing a better scenario in an acceptable time.

## VI. Final considerations and Future Works

In this paper, we propose a novel and flexible framework focus on organisation for VMP. To the best of our knowledge, this is the first work to combine multiple objectives evaluations with implementation costs to organise the VM in the Cloud.

The proposed framework supports different types of objectives, SLAs, strategies, best practices and costs in the form of qualifier, rule and cost functions and priorities. In order to demonstrate the advantages of our solution, we conducted several experiments, comparing it also with other approaches, and showed that our solution proposes a better scenario in a reasonable time.

In the future, we plan to: (i) implement a module to adapt Cloud's scenarios out of accordance with the environment rules; (ii) extend the *Cost-Benefit* method to consider multiple Costs during the selection phase; and (iii) implement rules and qualifiers that retrieve their logics from formal languages, such as SLAC [22].

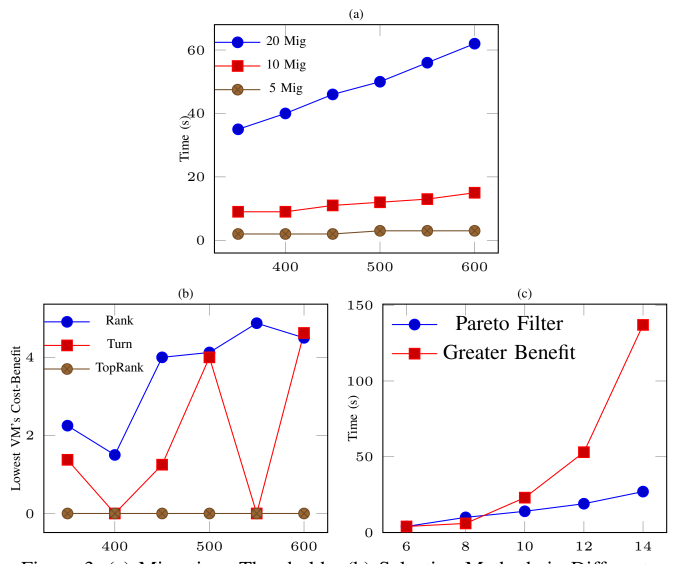

Figure 3. (a) Migrations Thresholds, (b) Selection Methods in Different Scenarios and (c) Filter Methods with Different Migrations Thresholds.

## References

[1] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, "Virtual machine power metering and provisioning," in Symposium on Cloud Computing – SoCC. ACM, 2010.

[2] S. Nathan, P. Kulkarni, and U. Bellur, "Resource availability based performance benchmarking of virtual machine migrations," in I.C. on Performance Engineering – SPEC. ACM, 2013.

[3] S. Abar, P. Lemarinier, G. K. Theodoropoulos, and G. M. OHare, "Automated dynamic resource provisioning and monitoring in virtualized large-scale datacenter," in I.C. on Advanced Information Networking and Applications – AINA. IEEE, 2014.

[4] A. Beloglazov and R. Buyya, "Energy efficient allocation of virtual machines in cloud data centers," in I.C. on Cluster, Cloud and Grid Computing – CCGrid. IEEE/ACM, 2010.

[5] C. C. T. Mark, D. Niyato, and T. Chen-Khong, "Evolutionary optimal virtual machine placement and demand forecaster for cloud computing," in I.C. on Adv. Info. Net. and Applications – AINA. IEEE, 2011.

[6] J. Xu and J. A. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in Green Computing and Communications – GreenCom / CPSCom. IEEE, 2010.

[7] L. Wang, J. Xu, M. Zhao, and J. Fortes, "Adaptive virtual resource management with fuzzy model predictive control," in I.C. on Autonomic Computing – ICAC. ACM, 2011.

[8] O. Abdul-Rahman, M. Munetomo, and K. Akama, "Toward a genetic algorithm based flexible approach for the management of virtualized application environments in cloud platforms," in I.C. on Computer Communications and Networks – ICCCN. IEEE, 2012.

[9] C. von Lucken, B. Barán, and C. Brizuela, "A survey on multi-objective evolutionary algorithms for many-objective problems," in Computational Optimization and Applications. Springer, 2014.

[10] Y.-C. Liang and M.-H. Lo, "Multi-objective redundancy allocation optimization using a variable neighborhood search algorithm," Journal of Heuristics, 2010.

[11] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," Transactions on Evolutionary Computation, 2003.

[12] W. Fang, X. Liang, S. Li, L. Chiaraviglio, and N. Xiong, "Vmplanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers," Computer Networks, 2013.

[13] P. Calyam, R. Patali, A. Berryman, A. M. Lai, and R. Ramnath, "Utility-directed resource allocation in virtual desktop clouds," Computer networks, 2011.

[14] J. Chen, K. Chiew, D. Ye, L. Zhu, and W. Chen, "Aaga: Affinity-aware grouping for allocation of virtual machines," in I.C. on Advanced Information Networking and Applications – AINA. IEEE, 2013.

[15] O. Biran et al., "A stable network-aware vm placement for cloud systems," in I.S. on Cluster, Cloud and Grid Computing – CCGrid. IEEE/ACM, 2012.

[16] D. Dong and J. Herbert, "Energy efficient vm placement supported by data analytic service," in I.S. on Cluster, Cloud and Grid Computing – CCGrid. IEEE/ACM, 2013.

[17] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, "Managing energy and server resources in hosting centers," Special Interest Group on Operating Systems – SIGOPS, 2001.

[18] J. Xu and J. Fortes, "A multi-objective approach to virtual machine management in datacenters," in I.C. on Autonomic Computing – ICAC. ACM, 2011.

[19] E. Feller, L. Rilling, and C. Morin, "Snooze: A scalable and autonomic virtual machine management framework for private clouds," in I.S. on Cluster, Cloud and Grid Computing – CCGrid. IEEE/ACM, 2012.

[20] Y. Ren, J. Suzuki, A. Vasilakos, S. Omura, and K. Oba, "Cielo: An evolutionary game theoretic framework for virtual machine placement in clouds," in I.C. on Future Internet of Things and Cloud – FiCloud. IEEE, 2014.

[21] G. A. Geronimo, J. Werner, C. B. Westphall, C. M. Westphall, and L. Defenti, "Provisioning and resource allocation for green clouds," in I.C. on Networks – ICN. IARIA, 2013.

[22] R. B. Uriarte, F. Tiezzi, and R. D. Nicola, "Slac: A formal service-level-agreement language for cloud computing," in I.C. on Utility and Cloud Computing – UCC. IEEE/ACM, 2014.