Order@Cloud: A VM Organisation Framework Based on Multi-Objectives Placement Ranking

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 presents the implementation and tests of a flexible and extensible framework, named Order@Cloud, that improves the Virtual Machine placements of a Cloud. It receives new VMs on the Cloud and organises them by relocating their placements based on the Multiple-Objectives of the environment. These Objectives are represented by Rules, Qualifiers and Costs, which can be easily added, extended and prioritised. Based on Evolutionary and Greedy Searches, Order@Cloud theoretically guarantees the adoption of a better set of Placements. More specifically, it seeks the non-dominated solutions (Pareto Set) and compares them considering the implementation cost of the scenario and its benefits. In contrast to existing solutions, that address specific objectives, our framework was devised to be objective-agnostic and easily extensible, which enables the implementation of new and generic prioritised elements. To understand the applicability and performance of our solution we conducted experiments using a real Cloud environment and discuss its performance, flexibility and optimality.**

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, mismanagement may cause performance degradation due to noisy neighbours, rise of thermal hotspots on data centres and shortage of resources as consequence of constant migrations [1,2]. Moreover, according to [3], approximately 30% of Physical Machines (PMs) have an average usage ratio of 10%, which corresponds to a considerable waste of energy and resources.

To prevent such problems, Virtual Machine Placement (VMP) methodologies try to mitigate the waste of resources by organising, consolidating and optimising Virtual Machines (VMs) in clouds. Many VMP approaches exists and are usually divided into the ones which aims at specific objectives, e.g., on decreasing the energy consumption or the number of Service Level Agreement (SLA) violations, and the ones that support Multiple-Objectives (MO), which are objetive-agnostic. Simulation-Based, Policy-Based, Bin Packing and Evolutionary Algorithms [4-6] are examples of proposals with specific objectives. However, supporting only specific objectives hinders the applicability of these approaches since they can not adapt to the varying Cloud's objectives, policies and priorities.

978-1-5090-0223-8/16/$31.00

To address this limitation many works, e.g., [7-9], propose VMP as a multiple-objective 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). However, such approaches must address the challenges originated from supporting MOs, such as conflict resolution, search convergence, scenario comparison and selection. To this aim, non-dominated scenarios are usually adopted to solve these conflicts and to guarantee the fast convergence of methods. 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 another scenario. However, this strategy stagnates the convergence in environments with many objectives [10]. Moreover, the high coupling between selection methods and the objectives become a challenge that hinders their applicability to agnostic MOs.

In light of these limitations and based on our previous research [11], we propose an easily extensible framework, named Order@Cloud, to address the VMP organisation problem. This framework adopts a flexible model which enables the assessment and comparison of multiple placements, supporting the cloud decision making. The main features of this Framework are: (i) platform and objective agnostic, (ii) supports objective's prioritisation, (iii) supports generic environment rules and (iv) focus on the VMs with low quality placements. Thus, our framework takes advantage of constraints to reduce the computation cost, enabling a fast local-optimal search.

The main contributions of our paper are:

- Description and formalisation of a Cloud model, devised for VMP problems, which represents MOs as several elements (Section II).
- Definition of a method, and its pseudo-algorithm, for the organisation of Cloud Scenarios, which are based on the Placement rank and Objective-Agnostics (Section III).
- A comprehensive discussion of related works, highlighting the main differences and specifying the works which we based parts of our solutions (Section IV).
- Demonstration of the advantages of our framework using real world data. In particular, we analyse its *Optimality* and *Performance* (Section V).

II. Cloud Model

In this section we propose a Cloud model for VMP problems. We describe its elements and provide their formal representation. These elements are the base for the framework.

In Cloud Computing, a *Placement* is a relation between a VM and a PM, guest and host respectively. A *Scenario*, in its turn, is a set of placements that defines where each VM is placed on the Cloud, which may represent the real Cloud placement or a possible one.

As presented in Section I, due to their dynamism, real Cloud environments require the support of multi-objectives solutions. In our model, MOs are represented by fours main elements: Rules, Qualifiers, Priorities and Costs.

Rules define placement's constraints, e.g., a rule can forbid a VM to be placed in a PM. They are divided into two types: Rules Free of Context (RFC) and Rules Sensitive to the Context (RSC). Here, *Context* refers to the placements of a given scenario, in which the VMs are placed. Therefore, RFC are used to define whether a given VM can be hosted by a specific PM, while RSC is needed to validate scenarios. In order to illustrate the rules, consider the following examples: (i – RFC) a VM A must be placed in a PM with ARM processors, and (ii – RSC) VM A should not be hosted in the same PM as VM B.

Since RFC do not require information about the context, it is possible to build a matrix summarising all the valid placements of each VM in the Cloud, as shown in Table I. This matrix can be used to accelerate and forecast the method's execution time.

Qualifiers are functions used to assess the quality of one or more placements in a scenario. They represent the objectives evaluations about the scenarios, which enable the comparison and selection of scenarios. Table II illustrates these qualifiers in a scenario where we compare four placements for the same VM, using three qualifiers: Energy, Network and Load (Q_{Energy}, $Q_{Network}$, Q_{Load}).

Priorities are applied to sort the qualifiers. They define weights and preferences between them, avoiding conflicts and providing flexibility to handle the dynamic changes of the Cloud. For instance, they are employed to solve conflicts between evaluations, such as "VM_1 in PM_2" and "VM_1 in PM_4" on Table II.

Finally, *Costs* are usually treated in most proposals [6,12,13] as a consequence of energy consumption or as just another assessment to be minimised. However, we also consider *cost* as any resource that cloud providers are willing to spend in order to achieve a certain amount of benefit. Its nature is a variable aspect in kind – such as time, resources or money – and it must take part of the decision process.

Thus, in this model, *Costs* are functions which quantify the implementation between two scenarios, an origin and a target. Its assessment is unit independent, such as the number of migrations or amount of bandwidth required to implement the target scenario. Additionally, a maximal threshold is related to each cost, which represent the maximum amount of resource which the Cloud Administrators are willing to invest in the improvement processes.

A. Formal Model

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

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, while H is a set with x PMs h (1). A Placement is a relation between any two elements from V and H, such as (v_c, h_d), as described in (1). A Scenario C is a set with y placements, $\{(v_c, h_d)_1, ..., (v_d, h_c)_y\}$, representing a Cloud scenario, as represented in (2).

$$V = \{v_1, ..., v_y\} \wedge H = \{h_1, ..., h_x\} \tag{1}$$

$$C = \{(v_1, h_1)_1, ..., (v_c, h_d)_y\} \tag{2}$$

In the formal model, RFC and RSC are represented by two distinct sets of functions, Rf and Rs, which contains f and s boolean functions f_{rfc} and f_{rsc} each, respectively. Given a Placement (v_c, h_d), a function f_{rfc} returns 1 if permitted or 0 otherwise, as showed in (3). Given a scenario C, a function f_{rsc} returns 1 if permitted or 0 otherwise, as described in (4).

Qualifiers, instead, are represented by Q, which is a set with z functions f_q, where $Q = \{f_{q_1}, ..., f_{q_z}\}$. Each function f_q, given any scenario C, returns a vector with y qualifications a, one for each placement in the scenario C, as described in (5). 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 scenarios.

To sort the qualifiers, the *Priorities* are represented as a function f_p which, given any qualifier f_q, returns a number u related to the qualifier's weight – as in (6).

Finally, I refers to a set of Cloud's Cost Functions f_i which, given two scenarios (C_{origin} and C_{target}), returns an integer value greater than zero related to the implementation Cost of the scenario C, as in (7).

Additionally, we have a set M referring to the Max Cost's Functions (f_m). These functions, given a Cost Function f_i, return a rational number ($m \in \mathbb{Q}$) referring the maximum cost allowed in the process, as represented in (8).

TABLE I. Example of the matrix of all possible placements.

VMs	P_1	P_2	P_3
V_1		✓	✓
V_2	✓	✓	
V_3		✓	✓
V_4	✓	✓	✓

TABLE II. Example of evaluation using qualifiers for placements.

Placements	Q_{Energy}	$Q_{Network}$	Q_{Load}
VM_1 in PM_1	0.1	2.0	0.1
VM_1 in PM_2	2.0	1.5	1.0
VM_1 in PM_3	0.1	2.0	2.0
VM_1 in PM_4	1.0	1.5	2.0

TABLE III. 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
c and d	Any random index
(V_c, H_d)	A placement
C	Set of y VM Placements
f_{rfc}	Rule Free of Context (RFC)
f_{rsc}	Rule Sensible to Context (RSC)
Rf	Set of RFC
Rs	Set of RSC
f_q	Qualifier Function
z	Number of Qualifiers
Q	Set of z Qualifiers
f_p	Qualifier's weights function
I	Set of Cost Functions
M	Set of Maximum Cost Function
f_i	Cost Function
f_m	Maximum Cost Function

TABLE IV. Table of Definitions

$$Rf = \{f_{rlc_1}, ..., f_{rlc_f}\} \quad f_{rlc}((v_c, h_d), \{0 \vee 1\}) \tag{3}$$

$$Rs = \{f_{rsc_1}, ..., f_{rsc_s}\} \quad rs(C, \{0 \vee 1\}) \tag{4}$$

$$Q = \{f_{q_1}, ..., f_{q_z}\} \quad f_q(C, \{a_1, ..., a_y\}) \wedge \forall a \in]0, 2] \tag{5}$$

$$f_p(f_q, u) \wedge \forall u \in (\mathbb{Q} >= 0) \tag{6}$$

$$f_i(C_{origin}, C_{target}, i) \wedge \forall i \in (0 < \mathbb{Q}) \tag{7}$$

$$f_m(f_i, m) \wedge \forall m \in \mathbb{Q} \tag{8}$$

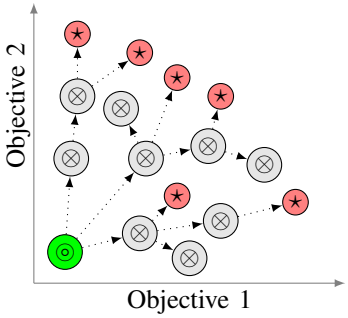

Fig. 1. Representation of dominance relation and Pareto Frontier

III. Order@Cloud Design

Based on the elements presented in the Cloud model and in the discussion presented in the Introduction, VMP methods should aim at: (i) *maximise* the placement's evaluations and (ii) *minimise* the implementation cost. At the same time, we should prioritise the improvement of low quality VMs and respect the Cost's limitations.

Thus, in this section, we present the proposed methodology to address VMP problem. Following, we describe and discuss the proposed method for organising the VMs placements, which focus on improving the placement's quality of each VM in the Cloud.

A. The Organising Method

Order@Cloud is a parallelisable, centralised and recursive process. It receives the current state of the Cloud – i.e., a list of VM placements representing the real scenario – and migrates specially selected VMs, seeking non-dominated scenarios to increase the placement's quality/evaluations. It also uses the Rules and Max Cost constraints to bound invalid branches and accelerate the convergence. At the end, it returns the scenario with the best cost-benefit among the found scenarios.

In this section we explain in details each step of the developed methodology to organise the VMs in the Cloud. Figure 2 and Algorithm 1 illustrate it.

Initiating: The method receives: (i) the current scenario, i.e., a vector with the real placements of the Cloud, (ii) an empty set of references to VM, used to control recursions (*ignoreVms*) and (iii) a boolean indicator to sign recursions (*isMainRecursion*), starting with *true* value. If the *ignoreVms* already contains all VMs of the cloud, it returns the current scenario, as shown in the line 1 of Alg. 1.

Placements Evaluation: To generate the VM's rank, inspired by [14], the framework evaluates all placements considering each qualifier, generating a 2D evaluation matrix with #VM x #Qualifiers of size, i.e., $ES_{z,y}$. Then, the weight defined for each qualifier is applied to their respective evaluation. Afterwards, the set is reduced to a vector E_y by multiplying all evaluations of each VM. Multiplication is used in order to enable evaluation's reduction, otherwise, even disapproval placements would lead to non-dominated scenarios. This reduction is represented in (9) and in line 2 of Alg. 1.

$$E_y \leftarrow \prod_{n=1}^{z} ES_{n,y}^{f_{p_n}} \tag{9}$$

VM Rank Selection: This step is based on the *Extract* step of the *Extract and Relocate* strategy [5], which selects and migrates VMs to improve some specific aspects of the state of a cloud. The VM's selection is commonly performed using heuristics or triggers, which attempt to choose key VMs to mitigate specific issues by migrating them. Since one of the premises of the problem is prioritise the improvement of the lowers placements, our approach selects the VM which: (i) has the lowest evaluation of the rank and (ii) is not present in the *ignoreVms* set. In Alg. 1, this step is shown in line 3.

Generation of Non-Dominated Scenarios: In this step, the method generates all valid scenarios in which the select VM is migrated, i.e., at most the *number of PMs - 1* scenarios. The scenarios are valid only if they comply with the following filters: (i) they respect the Rules (RFCs and RSCs, respectively), (ii) the cost is lower than the max costs specified, and (iii) a Dominance filter, which accepts only non-dominated scenarios.

Related to the Pareto's Dominance concept [10], a scenario is non-dominated if, compared to the current scenario, it improves the quality of at least one placement and, at the same time, does not worsen any other placements. The union of all the non-dominated results forms the Pareto Front set. Figure 1 graphically represents this evaluation considering two objectives. In this figure, the scenarios marked with a $(\star)$ are the non-dominated solutions, the $(\otimes)$ are dominated by their successors and the set of $(\star)$ are the Pareto Front set. This step is represented on Lines 4-7 of Alg. 1.

Search for new Scenarios: Order@Cloud starts a new recursion to explore each of the non-dominated neighbours scenarios. Similarly to the *Neighbourhood Search* approach [15,16], these recursions will be executed until no new non-dominated scenarios can be generated, returning the last scenario found. Additionally, to avoid loops, the following information is sent with it: (i) a boolean signing that it is not the main recursion; and (ii) a copy of the *ignoreVMs* set (*newIgnoreVMs*) with the VM selected in the *Placements Evalutation* included. The $newIgnoreVMs$ is used to avoid duplicated visits to the same scenario. Lines 8-11 of Alg. 1 show this step.

Select Best Scenario: The goal of this step is to select, according to the qualifiers, the best scenario among the recursions results, including also the current scenario (in case the current scenario is better than all the generated scenarios, no migrations are performed).

The selection of a scenario from the Pareto Front is not a deterministic choice and depends on the problem's nature and the applied model. There are a wide range of selection approaches that could be adopted for VMP, such as the number of fulfilled goals, distance from base scenario and error ratio [17]. To achieve the equilibrium between the implementation cost and the qualifier's evaluation, we considered the indicators defined in [17] and propose a method to calculate the Scenarios's Cost-Benefit and, consequently, select the best scenario, as described in (10). Here the benefit is defined as the difference between the evaluations of the current and a target scenario (C_C and C_T). More specifically, it is the sum of the variations of each VM evaluation of C_C and C_T. Then, the cost-benefit (cb) is calculated by the division of the benefit with the scenario's (C_T) cost – Lines 12-13 of Alg. 1.

Return Decision: Finally, this step represents a technical part of the recursion method, where the algorithm decides whether to continue the search or to return the selected scenario. If it is not the main recursion ($isMainRecursion$), it will return the selected scenario. Otherwise, it will *vary* the *Neighbourhood* [16] by stating a new recursion with 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. Then, it starts a new recursion using this scenario and $ignoreVms$ set. The result of this recursion is the method's output. Lines 14-19 of Alg. 1 represent these steps.

$$cb = \frac{\sum_{n=1}^{y}(C_{T_n} - C_{C_n})}{i(C_T)} \tag{10}$$

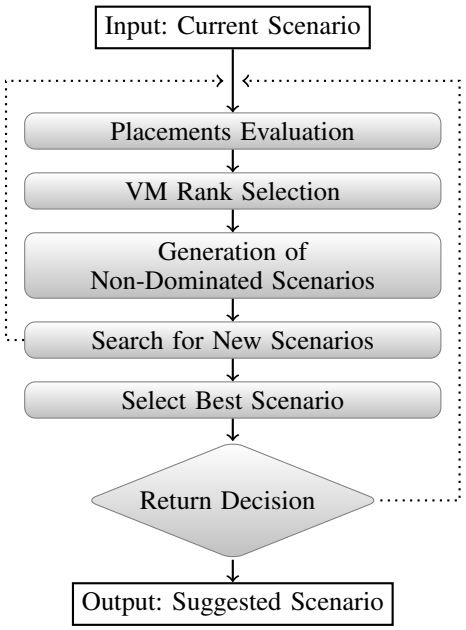

Fig. 2. Flow Chart of Order@Cloud

Algorithm 1: OrderCloud - Organise

```
   Input: baseScenario // current Cloud Scenario
   Input: ignoreVms // Set of VMs, starts empty
   Input: isMainRecursion // boolean, starts true
   Data: paretoSet // Auxiliary set of VMs
   Output: new // Suggest Scenario
1  if wereAllVmsExplored(ignoreVms) then return baseScenario;
2  baseEval = applyQualifiers( baseScenario );
3  lowVM = selectLowerVM( baseEval , ignoreVms );
4  scenarios = generateScenarios( baseScenario , lowVM );
5  foreach scenario in scenarios do
6      scenarioEval = applyQualifiers( scenario );
7      if isNonDominanted( scenarioEval , baseEval ) then
8          newIgnoreVms = ignoreVms;
9          newIgnoreVms.add( lowVM );
10         newScenario =
           OrderCloud(scenario,newIgnoreVms,false);
11         paretoSet.add( newScenario );
12 paretoSet.add( baseScenario );
13 selectedScenario =
   getScenarioWithMaxCostBenefit(paretoSet);
14 if isMainRecursion then
15     selectedScenarioEval = applyQualifiers(selectedScenario);
16     lowVM = selectLowerVM( selectedScenarioEval );
17     ignoreVms.add( lowVM );
18     selectedScenario =
       OrderCloud(selectedScenario,ignoreVms,true);
19 return selectedScenario
```

B. Discussion

The Cloud's initial consistency with the environment Rules is required for using the proposed method, therefore, the current state of the Cloud must be in accordance with the RFCs and RSCs. The adoption of an initial illegal scenario forbids the generation of new Scenarios. In this situation, neighbour scenarios are invalidated due to other illegal placements not related to the VM which is been migrated. Therefore, if the current scenario is invalid, the execution of an *Adaptation* method is necessary to achieve a temporary (fail-safe) scenario before starting the *Organization* process.

The ignore VMs set is used mainly to avoid deadlocks and duplicated visits to the same scenario. 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 explore other placements.

Although the *Ignore VMs set* use, together with the *Dominance* filter, ensures the convergence of the method, it does not assure finding the general best scenario. Cases where the temporary adoption of a degraded (dominated) scenario is needed to achieve the optimal one is the main reason. However, our solution always leads to a better scenario and does not degrade the Cloud in the process. In our experiments we approach this issue trying to assess this optimality.

Qualifiers based on Monitoring data, in some cases, are necessary to evaluate placements. However, an Evolutionary algorithm computes many scenarios per second, which makes it infeasible to depend on real-time data during the evaluations. For that reason it is recommended to pre-load the needed data or utilise approximation functions in the Qualifiers.

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.

TABLE V. Comparison between VMP Frameworks.

Proposal	*Objectives*	*Priorities*	*Organise*	*Provide*	*Cost*
Snooze [18]	✓				
Cielo [19]			✓		
VMPlanner [12]			✓		
MO-BBO [20]			✓		
MO-VMM [21]		✓			✓
MO-VMP [7]				✓	
Order@Cloud	✓	✓	✓	✓	✓

TABLE VI. Growth of the Worst Scenario in comparison with smaller ones.

Assessment / Scenario	(a) 350-VMs & 20-Mig.	(b) 600-VMs & 5-Mig.
Cost-Benefit	+44%	+104%
Execution Time	+77%	+1966%
Analysed Scenarios	+2%	+7355%

IV. Related Works

In this section we present related works, and the concepts and algorithms we based our approach, highlighting the main differences between our work and the related literature.

Order@Cloud uses the *Extract and Relocate* approach to select and migrate VMs to new placements. It was inspired in the approach proposed by Beloglazov et al. in [5], and it is also used by Xu et al. in [21]. However, differently from Beloglazov's work, which uses three pre-defined policies to choose the VM to be *Extracted*, we proposed a Rank's approach based on Biran's et al. [14] solution to mitigate network's traffic. Instead of focusing on Rank VMs by their network demands, we select and *Extract* the lowest VM in the evaluations rank. After, instead of using a Modified Best Fit Decreasing algorithm, we generate and consider all possible scenarios in which the VM could be relocated.

Likewise, the scenario filtering was based on the Solution's Dominance concept, which is also used in [16,17]. However, the original definition compares the objectives assessments separately, which is not suitable to environments with many objectives [10]. We, instead, compare the summary of the placements' evaluations, to avoid the method's stagnation due to quantity of objectives. The judgement is based on the Scenarios' evaluations, which is a combination of all qualifiers assessments and their weights.

During and after the search we use the concept of *Variable Neighbourhood Search* (VNS) [15,16] to improve the result's quality and to accelerate method's convergence. Succinctly, VNS is a strategy which builds a Pareto Front by recursively searching Non-Dominated results from different neighbourhoods of the result space. However, instead of using random scenarios to vary the searched Neighbourhoods, we employ the *IgnoreVms* set with the VM Placement Rank to define the new starting point scenario.

VMP has been extensively studied and many works were published over the last many years. For an extensive review on the existing solutions, we refer to [22-25]. Most of works focus on specific issues, such as network optimisation and energy consumption, hindering their applicability; thus they are not considered in our comparison.

We compare the most important proposals which claim to consider MO and/or propose frameworks to generalise and facilitate the studies in this area.

Table V summarises this comparison, which considers the following questions:

(i) Does it enable implement new Objectives (i.e., is objective-agnostic)?

(ii) Does it enable priority changes?

(iii) Does it offer a method to Organise all the VMs according the Objectives?

(iv) Does it provides a method to place new VMs?

(v) Does it considers the scenario's implementation costs?

V. EXPERIMENTS

In this section, we present the experiments conducted in a real scenario to assess: (i) the cost-benefit achieved, compared to a related work; and (iii) its scalability.

A. Environment Experiment Settings

In the experiments, we use real placement data from the data centre of the Federal University of Santa Catarina. The environment where the data was collected is orchestrated by VMWare 5.5 and is composed of: 607 VMs, 18 PMs, 99 storage pools and 102 networks.

The framework prototype and its tests were implemented in PHP 5.5. The features and the above tests were implemented using the Test's Framework PHPUnit 4.2. Our implementation, the tests described above and more informations are open source and are available on the project's website[1].

The tests were executed in an environment with Ubuntu 14.04, an *Intel T9400* processor of 2.53GHz with 4GB of RAM, which processed on average 250 scenarios per second. They were executed in a scenario with a threshold of 20 migrations and the number of VMs ranged between 350 and 600. Regarding the Order@Cloud settings used in 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 hosting the VM; (iii) the cluster coherence for the migration, i.e., whether the VMs are in the same PMs' cluster where they belong to; and (iv) the scenario's costs cannot exceed their limits.

Despite of the fact that the tests do not focus on *Qualifier's* quality, to test the methods we implemented following strategies: (i) Consolidate VMs in few PMs by rewarding migrations that decrease resource wastage, aiming at freeing idle PMs and turn them off. (ii) Distribute VMs from the same services in different PMs, since services of the same type tend to compete for the same resources [26]. We penalise migrations that overload PMs with sibling VMs. (iii) Distribute VMs of the same storage pool, aiming at balance the Storage Area Network (SAN) use, by dispersing its users (the VMs) among diverse PMs.

Finally, the evaluation of cost computes the number of migrations necessary to reach the target scenario.

B. Cost-Benefit Comparison

Trying to quantify the *optimality* loss due to the *Dominance* constraint in our solution, we compare our Organization method with an adapted approach used by proposal [21]. This approach employs the Greatest Benefit strategy as a heuristic to a Greedy Search, i.e., it seeks the scenarios which offers the greatest benefit. Since their proposal handle VMs on demand, we decided to use the Rank selection method, instead of the Random selection. Furthermore, to avoid discard valid search branches, we disabled the Max Costs constraints. Thus, due to the exponential growth of the compared method, we had to reduce our scenarios to decrease the amount of time to execute all tests, as shows in the results. Using this methodology we performed two different searches, selecting scenarios with the greatest: (i) Benefit and (ii) Cost-Benefit found.

[1] http://ordercloud.lrg.ufsc.br

The results regarding the execution time, represented in Fig. 3 (a), make evident the performance of our method, since, in the longest test, it took less than 1 second to achieve the result. While the compared method took 30 seconds in its fastest test, and approximately 19 minutes in the longest test.

Analysing the biggest scenario (20 Mig.), our method achieved the same result as the compared one using the Search (ii), as shown in Fig. 3(c). The highest benefit found by the Search (i) was 26% higher than ours – Fig. 3(b) – , however its implementation cost was so high that our cost-benefit end up been 208% higher than this result – Fig. 3(c).

In summary, even staying 26% away from the optimal scenario, our method is the best choice to find the scenarios with the best Cost-Benefit. Thus, these results led us to another question, would our method scale to a real Cloud scenarios?

C. The Scalability of Order@Cloud

In experiments related to *scalability*, we measure the necessary time to evaluate a real scenario and to propose better one, varying the number of VMs (350–600) and the maximum number of migrations allowed (5–20).

The worst case scenario – 600 VMs and 20 migrations – analysed 13,000 scenarios and it took barely 62 seconds to reach the best scenario — using a standard hardware as described in the previous section. Fig. 3 (d) graphically presents its assessments growth, while Table VI presents its rates in comparison with two other smaller scenarios: (a) one with 350 VMs and (b) one with 5 migrations.

Interestingly, the *Max Migration* constraint had a greater impact in the execution time than the *scenario's size*. It considerably increased the Cost-Benefit Rate, the number of analysed scenarios and, consequently, the execution time. Still, a linear pattern on the number of scenarios was noticed due to this threshold, which forced the premature stop of the method.

These results suggest that our approach, even without optimisation and parallelisation, cope well with small and medium Clouds scenarios. However, we also notice that its performance is related to the *Average Placement Rate* (APR) of the scenario, which is the average amount of places that each VM can be hosted. The APR can be calculated using the *Summary Matrix*, based on the RFC (Section II). This relates to the number of generated scenarios and, consequently, to the execution time. We consider that the APR is a key factor to forecast the method's performance and to optimise our approach to bigger scenarios, with more than 10K VMs. Thus, we will consider it in future researches.

VI. FINAL CONSIDERATIONS AND FUTURE WORKS

In this paper, we presented a novel and flexible framework for VMP, which focus on the provisioning, organisation and adaptation of Cloud Scenarios. Our framework was implemented as an open-source project available in our website[1]. This framework is the first that combines multiple objectives evaluations with implementation costs to organise the VM in the Cloud. Moreover, it supports different types of objectives, SLAs, best practices and costs in the form of qualifiers, rules, cost functions and priorities.

In order to demonstrate the advantages of our solution, we conducted several experiments comparing our Framework with similar approaches and measuring its scalability.

In the future we plan to: (i) analyse the impact of the *Average Placement Rate* on the methods, as discussed in Section V-C; (ii) use smart strategies to reorder rules' application; and (iii) implement rules which retrieve their logics from formal languages, such as SLAC [27] and SCEL [28].

Acknowledgment

This work has been partially supported by CNPq through the Science Without Borders programme. The authors also would like to thank Rocco De Nicola and the reviewers for their encouragements and fruitful comments.

References

[1] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, "Virtual machine power metering and provisioning," in *1st 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 *4th I.C. on Performance Engineering – SPEC*. ACM, 2013.

[3] M. Uddin, A. Shah, R. Alsaqour, and J. Memon, "Measuring efficiency of tier level data centers to implement green energy efficient data centers," *Middle-East Journal of Scientific Research – MEJSR*, 2013.

[4] S. Abar, P. Lemarinier, G. K. Theodoropoulos, and G. M. OHare, "Automated dynamic resource provisioning and monitoring in virtualized large-scale datacenter," in *Advanced Information Networking and Applications (AINA), 2014 IEEE 28th I.C. on*. IEEE, 2014.

[5] A. Beloglazov and R. Buyya, "Energy efficient allocation of virtual machines in cloud data centers," in *10th I.C. on Cluster, Cloud and Grid Computing – CCGrid*. IEEE/ACM, 2010.

[6] 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. Information Net. and Applications – AINA*. IEEE, 2011.

[7] 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.

[8] G. Lee, N. Tolia, P. Ranganathan, and R. H. Katz, "Topology-aware resource allocation for data-intensive workloads," in *1st Asia-Pacific Workshop on Systems – APSys*. ACM, 2010.

[9] 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 *21st I.C. on Computer Communications and Networks – ICCCN*. IEEE, 2012.

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

[11] G. A. Geronimo, R. B. Uriarte, and C. B. Westphall, "Towards a framework for vm organisation based on multi-objectives," in *15th I.C. on Networks – ICN*. IARIA, 2016.

[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 vd clouds," *Computer networks*, 2011.

[14] O. Biran, A. Corradi, M. Fanelli, L. Foschini, A. Nus, and D. Raz, "A stable network-aware vmp for cloud systems," in *12th I.S. on Cluster, Cloud and Grid Computing – CCGrid*. IEEE/ACM, 2012.

[15] P. Hansen and N. Mladenovic, "Variable neighbourhood search," *Handbook of Metaheuristics, Dordrecht, Kluwer Academic Publishers*, 2003.

[16] Y.-C. Liang and M.-H. Lo, "Multi-objective redundancy allocation optimization using a vns algorithm," *Journal of Heuristics*, 2010.

[17] 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.

[18] E. Feller, L. Rilling, and C. Morin, "Snooze: A scalable and autonomic virtual machine management framework for private clouds," in *12th I.S. on Cluster, Cloud and Grid Computing – CCGrid*. IEEE/ACM, 2012.

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

[20] Q. Zheng, R. Li, X. Li, and J. Wu, "A multi-objective biogeography-based optimization for virtual machine placement," in *15th I.S. on Cluster, Cloud and Grid Computing – CCGrid*. IEEE/ACM, 2015.

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

[22] K. Bilal, S. U. R. Malik, O. Khalid, A. Hameed, E. Alvarez, V. Wijaysekara, R. Irfan, S. Shrestha, D. Dwivedy, M. Ali *et al.*, "A taxonomy and survey on green data center networks," *Future Generation Computer Systems*, vol. 36, 2014.

[23] E. C. Inacio and M. A. R. Dantas, "A survey into performance and energy efficiency in hpc, cloud and big data environments," *Int. J. Netw. Virtual Organ.*, vol. 14, no. 4, Mar. 2014.

[24] C. J. Rathod, "A survey on different virtual machine placement algorithms," *Inter. Journal*, vol. 2, no. 2, 2014.

[25] F. Lopez Pires and B. Baran, "A virtual machine placement taxonomy," in *Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on*. IEEE, 2015, pp. 159–168.

[26] R. B. Uriarte, S. Tsaftaris, and F. Tiezzi, "Service clustering for autonomic clouds using random forest," in *I.S. on Cluster, Cloud and Grid Computing – CCGrid*. IEEE, 2015.

[27] R. B. Uriarte, F. Tiezzi, and R. D. Nicola, "Slac: A formal service-level-agreement language for cloud computing," in *7th I.C. on Utility and Cloud Computing*. IEEE/ACM, 2014.

[28] R. De Nicola, M. Loreti, R. Pugliese, and F. Tiezzi, "Scel: a language for autonomic computing," *Univ. Firenze, Tech. Rep*, 2013.

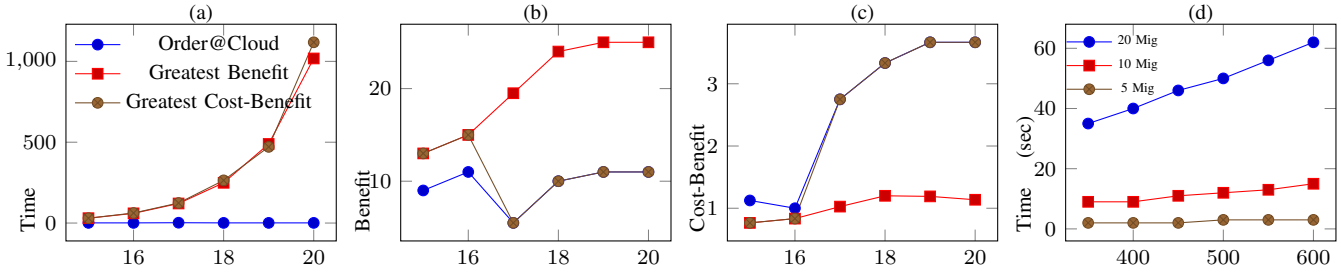

Fig. 3. Comparison between methods, number of migrations by: (a) Execution Time, (b) Benefit, and (c) Cost-Benefits. – (d) Time vs. Number of VMs.