2017 IEEE International Parallel and Distributed Processing Symposium Workshops

Consumer-and-Provider-oriented efficient IaaS resource allocation

Thibaud Ecarot
UMR CNRS Samovar, Télécom SudParis
and Thales-ThereSIS, Palaiseau, France
thibaud.ecarot@telecom-sudparis.eu

Djamal Zeghlache
UMR CNRS Samovar, Télécom SudParis
and Université Paris Saclay, France
djamal.zeghlache@telecom-sudparis.eu

Cedric Brandily
Thales-ThereSIS, Palaiseau, France
cedric.brandily@thalesgroup.com

***Abstract*—This paper presents a method of cloud resource allocation designed to take into account both consumers and providers' interests. This comes in contrast to today's provider-centered models that subject users to more restrictive terms and conditions. Both parties' interests are computed in the form of integer constraints. Costs and availability are embedded as key objectives and performance criteria in the model. We propose a hybrid resolution method based on an evolutionary algorithm augmented with a tabu search and compare performance with other resource allocation algorithms. The comparison results reveal the efficiency of the proposed hybrid evolutionary algorithms for consumer and provider centric cloud resources allocation.**

***Index Terms*—Cloud Computing; Efficient resource allocation; Cloud Computing Modeling; Optimization; Genetic Algorithm;Tabu Search**

I. Introduction

Advances in cloud computing and virtualization are making cloud infrastructures and services provisioning and management easier and more flexible. With virtualization, providers can optimize the use of their resources and minimize cost by reducing the number of servers to provide on demand and pay per use end user services. The provisioning of containers and physical resources acting as hosts of the virtual resources is achieved through allocation algorithms [1] which optimize resource consumption and reduce costs. Some other algorithms also target consolidation of servers to further reduce costs and increase revenues for the providers [2][3] that are in charge of maintenance, system backups and resilience planning to reach established service agreements with the consumers.

Typically, the state-of-the-art resource allocation algorithms and provisioning systems are provider-centric probably due to the use of highly complex algorithms considering just the providers business objectives. The trend is also to find either exact or very good suboptimal solutions that can scale. Despite these limitations and high complexity, we contend that both the consumers and providers interests can be included into the modeling and the optimization solution using evolutionary algorithms known to be more suitable for multiple (often opposed) objectives, multiple constraints and multiple criteria optimization.

Our main goal here is to implement end-to-end automated resource allocation within a cloud computing platform to meet providers business interests and consumer or end user objectives. Our model consequently includes affinity/anti-affinity constraints in order to reflect the stakeholders' interests. To derive a model for the cloud resource allocation, we first express the problem using a linear programming approach. From this background model, we build evolutionary algorithms based on NSGA-II and NSGA-III. We have enhanced the NSGA-III genetic algorithm with a tabu search algorithm to respect user constraints and objectives. The proposed algorithms are compared with traditional algorithms such as Round Robin, a constraint solver or the unmodified NSGA-II and NSGA-III.

II. Related Work

User service requests (e.g. virtual machines or storage) can be now handled by energy-saving algorithms [4] or cost-effective algorithms [5] for a given platform while taking into account capacity for each hypervisor (or Virtual Machine Monitor, VMM). This state of the art does not provide solutions that meet user needs like data protection and security.

In the existing literature, many authors address energy savings in the Infrastructure as a Service (IaaS) context [6] [7], business profitability [8] [9], provider cost reduction [10] and in some cases even service level agreements (SLAs) [11]. Some authors generalize the solutions through more general modeling using resource description languages and special resolutions [12].

In [13], the authors present a server consolidation (named BtrPlace) using constraint programming. Each virtual machine or server has associated constraints specifying the relationships and dependencies between resources. This model, however, does not include affinity/anti-affinity relationships between cloud resources or services.

Authors in [14] use game theory to solve the resource allocation problem by considering that multiple types of resources should be shared fairly among users. They pack complementary VM types in physical servers to improve their usage. The problem is modeled as a finite extensive game. Each resource-providing physical server is treated as a game player aware of the utility information of other players. This method applies if we want to ensure fairness across users but definitely does not address the complex relationships that can be found between providers and consumers.

In [15], the authors present a tool, ”Wrasse”, that can be used in cloud environments to solve their specific allocation problem. They provide an expressive specification language

978-1-5386-3408-0/17 $31.00 © 2017 IEEE
DOI 10.1109/IPDPSW.2017.97

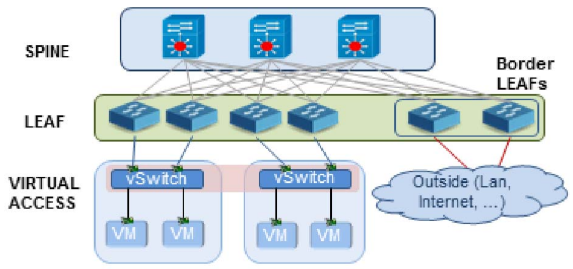

Fig. 1. Spine and Leaf Modern Architecture of datacenter

that remains, however, quite limited due to its initial implementation as a bin-packing problem. In addition, their modeling does not allow defining objectives, but only responds with a solution that meets the constraints.

Similarly, scalable placement problems have been addressed by several authors. In [16], authors present a load-balancing algorithm that unfortunately makes no attempt to co-locate the set of ressources on the same machine. Work of [17][18] introduces efficient placement of virtual machines on cloud infrastructure with consideration of resource availability. These placement strategies target improving the availability of some resources, but neglect the availability of the whole services.

III. Modeling cloud resource allocation

As stated previously, a cloud computing platform offers services to customers. This platform is installed on an infrastructure managed by a provider. This service provider strives to meet the user requirements while ensuring delivery of the provided services. Thanks to virtualization, a single server can simultaneously run multiple virtual components, each embedded in a virtual machine, as long as the total amount of demands in resources does not exceed the server capacity.

Our analysis and model are based on a very popular cloud architecture which has proven to be successful in managing both redundancy [19] and bandwidth [20]: the Core/Leaf-Spine distributed network architecture [21] as depicted in Figure 1.

Figure 2 shows the composition of an IaaS-type cloud computing system with its different layers. The virtual resources are assigned on hypervisors from the hardware layer. Connections between resources like affinity or anti-affinity constraints can be embedded in the allocation model.

In order to model the cloud resource allocation problem taking into account the interests of the providers and the consumers simultaneously, we need to call on appropriate resource descriptors and attributes. This way, we can describe the user request and hosting infrastructure and express the constraints and user or consumer service level requirements. These descriptors will help translate the user requirements and provider business goals into variables and attributes for optimization model variables, objectives and constraints. We define and describe the provider resources and the requested

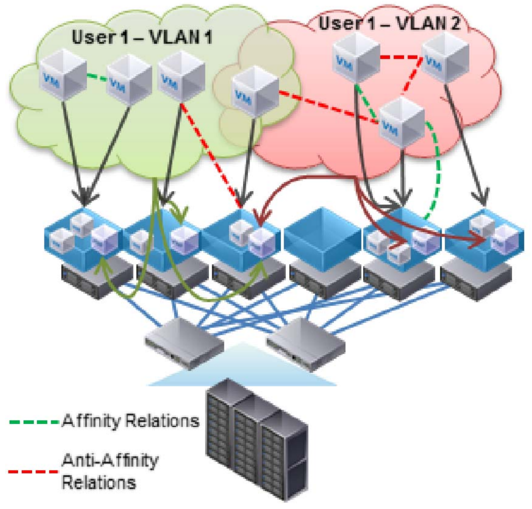

Fig. 2. Virtual resource allocation in IaaS

virtual resources with a focus on common or matching attributes on both layers; the virtual and physical layers. In this paper, we focus on attributes such as CPU, RAM and disk for each virtual and physical resource. In addition, our model can be extended to other specific attributes to provider resources. The values for the operating costs of resources are obtained from the accounting department.

The user requests are translated into a virtual resource topology connecting virtual machines in compliance with their affinity/anti-affinity relationships. Since our goal is also to reduce operating costs of the physical infrastructure (or the IaaS platform), we use matrices to represent and manage the involved multiple constraints and objectives. The use of a matrix representation of operating costs, performance objectives and constraints naturally leads to the use of a constraint solver to achieve resource allocation. In summary, we propose a method to reflect the users' requests and their interests and describe providers' infrastructures in the form of matrices to solve the user- and provider-aware cloud resource allocation problem.

The concrete case that we want to solve is resource allocation problem within IaaS platform where users make requests for virtual infrastructure services. Within the same request, it is possible to have different types of services such as CPU, memory, affinity and anti-affinity constraints. Our scheduler is aware of the cloud platform status in real time. Our idea is to directly include all requests within a cyclic time window during the execution of the allocation optimization process. We want to obtain a scalable model so as to represent requests from both the users and provider. We seek to optimize efficiency in data access and the representation of affinity/antiaffinity constraints. This way, the matrix solution suits our needs in the most adaptive way.

Our algorithm solves the cloud resource allocation problem using the users' and providers' layers with the objective of placing (as close to optimal as possible) the demands to minimize cost, service unavailability and the need to migrate or move resources in the infrastructures. Without loss of generality we assign equal weights to these objectives, that can otherwise be tuned and configured differently by the stakeholders. The objectives are specified as follows:

- **Usage and operating cost** determines the full cost of an infrastructure integrating consumer resources consumption induced cost and provider's exploitation cost;
- **Downtime cost** represents the cost due to the unavailability of consumer resources;
- **Migration cost** represents the cost of migrating consumer resources through the reconfiguration plan.

We identify five major relationships and constraints to describe the user affinity/anti-affinity requirements between used requested resources and providers' resources:

- **Co-localization in same datacenter** imposes the colocation user requested virtual resources in the same data center;
- **Co-localization on same server** imposes placement of the consumer virtual resources on the same server or host;
- **Separation in different datacenters** splits consumer resources onto different datacenters;
- **Separation on different servers** the consumer resources can be hosted in the same data center but need to be placed on different servers.

The description of the user resource requests and the provider physical infrastructure through matrices is meant to be generic enough so it can be reused and generalized to other types of services. The resource allocation model is obtained by defining the user and provider resources as follows: the number of provider data centers is noted g, the number of servers defined as m, the total amount of requested resources is represented by n and the number of attributes by h. In our model, the number of provider's resource attributes equals the number of consumer's resource attributes. The mapping is on identical attributes with respect to the demand. The resource capacities (maximum available resource per node or link) are also represented by two matrices. The first matrix [Eq. 1] represents the capacity of each attribute for the provider resources.

$$\begin{array}{cc} P_{jl} & 1 \leq j \leq m, 1 \leq l \leq h \\ P_{jl} \in \mathbb{R}^{+} & m \times h \end{array} \tag{1}$$

The second matrix [Eq.2] represents the capacity limit of each attribute for the consumer resources.

$$\begin{array}{cc} C_{kl} & 1 \leq k \leq n, 1 \leq l \leq h' \\ C_{kl} \in \mathbb{R}^{+} & n \times h' \end{array} \tag{2}$$

The numbers of attributes for provider resources and consumer resources should be the same for both types of resources ($h = h'$). We also define a factor, reflecting the extra amount of a physical resource required to produce a virtual resource (for example real CPU spent for providing a virtual CPU in a given system). This factor is described by the matrix of size $m \times h$ of [Eq.3].

$$\begin{array}{cc} F_{jl} & 1 \leq j \leq m, 1 \leq l \leq h \\ F_{jl} \in \mathbb{R}^{+} & m \times h \end{array} \tag{3}$$

We note X_{ijk}, a boolean variable, which is true when a consumer resource k is hosted on a provider's server j in datacenter i. We define a **provider resources capacity limit** that cannot be exceeded using [Eq.4]. This constraint includes the virtual-to-physical resource consumption factor for each requested resource type or attribute.

$$\sum_{k=1}^{n} C_{kl} \times X_{ijk} \leq P_{jl} F_{jl}, \forall l = 1...h, \forall i = 1...g \tag{4}$$

The **allocation constraint** is expressed via the sum of resources, allocated to all datacenters and servers, that has to be equal to one. It enforces the allocation of resources of a user request as indicated in [Eq. 5]:

$$\sum_{k=1}^{n} X_{ijk} = 1, \forall i = 1...g, \forall j = 1...m \tag{5}$$

To represent the operation expenditure of each provider resource, we introduce a vector in [Eq.6] whose elements are costs representing the direct costs associated to the use of a server: power, floor space, storage, and IT operations for resource management.

$$E_j \quad 1 \leq j \leq m, \quad E_j \in \mathbb{R}^{+} \tag{6}$$

In addition to operating expenditures, provider resources have a usage cost for each hosted consumer resource. We use a vector [Eq. 7] to express these usage costs.

$$U_j \quad 1 \leq j \leq m, \quad U_j \in \mathbb{R}^{+} \tag{7}$$

Each attribute for each provider resource has a maximum load before quality of service deterioration or degradation occurs in the hosted virtual resources. We note L_{jl}^{M} the maximum loading before degradation and L_{jl} the current load on server j for an attribute l. These servers have a current and a maximum achievable quality of service Q_{jl} and Q_{jl}^{M} respectively. This level of service guarantee is noted C_k^Q and if it is not respected the provider pays a downtime penalty (degradation, QoS or SLA violation) defined as a cost C_k^U. The service provider must guarantee that there will be no interruption in service. We define these load and quality of service matrices in [Eq.8]:

$$\begin{array}{cc} L_{jl}^{M} & 0 \leq L_{jl}^{M} < 1 \\ L_{jl} & 0 \leq L_{jl} < 1 \\ Q_{jl}^{M} & 0 \leq Q_{jl}^{M} < 1 \\ Q_{jl} & 0 \leq Q_{jl} < 1 \end{array} \tag{8}$$

All the variables and constants used in our model are listed for easy reference in Table I.

The model is augmented with all the required relationships and constraints expressed in the user requests and the valid equalities and inequalities as related to the demand and hosting

TABLE I
CLOUD RESOURCES ALLOCATION MODEL VARIABLES

	Provider substrate and requested resources
G	Set of available datacenters $G = \{1, ..., g\}$
M	Set of available servers $M = \{1, ..., m\}$
N	Set of requested resources $N = \{1, ..., n\}$
H	Set of available attributes for each resource $H = \{1, ..., h\}$
	Capacities matrices
P_{jl}	Capacity of provider resource j for attribute l
C_{kl}	Capacity of requested resource k for attribute l
F_{jl}	Capacity factor of attribute l on provider resource j
	Mapping model
X_{ijk}	Boolean variable indicating whether requested resource k is assigned to provider resource j in datacenter i
	Quality of service matrices and vectors
L_{jl}	Load of attribute l on provider resource j
L^M_{jl}	Maximum load of attribute l on provider resource j
Q^M_{jl}	Maximum quality of service for attribute l on provider resource j
Q_{jl}	Quality of service for attribute l on provider resource j
C^Q_k	Quality of service guaranteed by the provider on consumer resource k
	Costs vectors
E_j	Operational expense for a provider resource j
U_j	Usage cost for each consumer resources hosted on a provider resource j
C^U_k	Cost of downtime for a resource k
M_k	Migration cost of a consumer resource k

infrastructures. These constraints and valid conditions are formally defined in the sequel.

The **co-localization in same datacenter** constraint is expressed in [Eq.9]:

$$\sum_{i=1}^{g} \prod_{k=1}^{n} X_{ijk} = 1, \forall j = 1...m \tag{9}$$

The **co-localization on same server** constraint is achieved if the sum of the products on the consumer allocated resources meets [Eq.10]:

$$\sum_{i=1}^{g} \sum_{j=1}^{m} \prod_{k=1}^{n} X_{ijk} = 1 \tag{10}$$

The **separation in different datacenters** constraint is an extension of the previous constraint and involves summing across all data centers as expressed in [Eq. 11]:

$$\sum_{i=1}^{g} \sum_{k=1}^{n} X_{ijk} = 1, \forall j = 1...m \tag{11}$$

The **separation on different servers** constraint involves summation across datacenters, servers and requested resources as expressed in [Eq. 12]:

$$\sum_{i=1}^{g} \sum_{j=1}^{m} \sum_{k=1}^{n} X_{ijk} = 1 \tag{12}$$

The co-localization in same datacenter and co-localization on same server constraints have a non-linear form. So we linearize these two constraints by replacing the product with a variable (Eq. 13).

$$\begin{array}{ll} \sum_{i=1}^{g} \prod_{k=1}^{n} X_{ijk} = 1, & \forall j = 1...m \\ \sum_{i=1}^{g} v_{ijk} = 1 & \forall j = 1...m \\ \sum_{i=1}^{g} \sum_{j=1}^{m} \prod_{k=1}^{n} X_{ijk} = 1 & \\ \sum_{i=1}^{g} \sum_{j=1}^{m} v_{ijk} = 1 & X_{ijk} = 1 \end{array} \tag{13}$$

Then we use inequality constraints to linearize (Eq. 14).

$$\begin{array}{ll} v_{ijk} \equiv \prod_{k=1}^{n} X_{ijk} & X_{ijk} \in \{0,1\} \\ v_{ijk} \leq X_{ijk} & \forall i, j, k \\ v_{ijk} \geq \sum_{k=1}^{n} -(n-1) & \\ v_{ijk} \in \{0,1\} & \end{array} \tag{14}$$

These constraints are not all used at once. It depends on the user requests. All these relationships and constraints are implemented in our linear programming model to find solutions to the cloud resource allocation problem addressed in this work.

We solve the resource allocation problem according to usage and exploitation costs, affinity/anti-affinity relations between resources and required quality of service on the assigned cloud computing resources. The approach takes into account multiple constraints, relationships and objectives but the multiple objectives are aggregated in an overall objective function.

Since our optimization problem is similar to the multidimensional bin packing and the knapsack problems that have been shown to be NP-Hard, our problem is also NP-Hard. The multidimensional bin packing problem is a vector scheduling problem with d dimensions proven to be NP-Hard in [22]. As our objective is to minimize costs, service interruptions and reconfiguration plan sizes, we introduce global objective function [Eq. 15] to compute, namely the costs, the quality of services of resources and the approximate evaluation of the reconfiguration plan sizes. Our resource allocation problem can be summarized by lumping all the objective functions with all the constraints and conditions into the following set of equations:

$$\begin{aligned} Z \quad &= min[\textstyle\sum_{j=1}^{m} E_j \times X_{ijk} + \sum_{k=1}^{n} U_j \times X_{ijk}, {}_{\forall i=1...g} \\ &\quad + \textstyle\sum_{k=1}^{n} C^U_k(\lfloor \frac{Q_{jl}}{C^Q_k} \rfloor) \times X_{ijk}, {}_{\substack{\forall i=1...g \\ \forall j=1...m \\ \forall l=1...h}} \\ &\quad + \textstyle\sum_{k=1}^{n} (X^t_{ijk} - X^{t+1}_{ijk}) M_k, {}_{\substack{\forall i=1...g \\ \forall j=1...m}}] \end{aligned} \tag{15}$$

Subject according to the request:

$$\sum_{k=1}^{n} C_{kl} \times X_{ijk} \leq P_{jl} F_{jl}, \forall l = 1...h, \forall i = 1...g \tag{16}$$

$$\sum_{k=1}^{n} X_{ijk} = 1, \forall i = 1...g, \forall j = 1...m \tag{17}$$

$$\sum_{i=1}^{g} \prod_{k=1}^{n} X_{ijk} = 1, \forall j = 1...m \tag{18}$$

$$\sum_{i=1}^{g} \sum_{j=1}^{m} \prod_{k=1}^{n} X_{ijk} = 1 \tag{19}$$

$$\sum_{i=1}^{g}\sum_{k=1}^{n} X_{ijk} = 1, \forall j = 1...m \tag{20}$$

$$\sum_{i=1}^{g}\sum_{j=1}^{m}\sum_{k=1}^{n} X_{ijk} = 1 \tag{21}$$

Clearly, we can solve the problem through an integer linear programming approach using the model proposed to minimize the costs and the costs of temporary service disruptions in equation 15 subject to the constraints specified in equations 16 to 21.

The objective functions and the metrics are all converted to an equivalent monetary cost so they can be aggregated into a global objective function [Eq.15]. The first term in the objective function is the full cost calculation given by [Eq.22]. This term comprises usage and exploitation costs. For each provider resource, E_j is the exploitation cost of provider resource j. For each consumer resource, variable U_k represents the consumer resource consumption (usage) cost k.

$$\sum_{j=1}^{m} E_j \times X_{ijk} + \sum_{k=1}^{n} U_j \times X_{ijk}, {\scriptstyle \forall i=1...g} \tag{22}$$

The second term [Eq.23] represents the costs of temporary service disruptions or penalties for quality of service degradations or violations by the providers. On the basis of performance assessments, authors [23] [24] prove with empirical investigations that the quality of service of consumer resources decreases exponentially with increasing workload in provider resources. The total dowtime cost is the sum of each downtime cost C_k^U on consumer resources k when the quality of service guarantee C_k^Q is not respected.

$$\sum_{k=1}^{n} C_k^U(\lfloor \frac{Q_{jl}}{C_k^Q} \rfloor) \times X_{ijk}, {\scriptstyle \substack{\forall i=1...g \\ \forall j=1...m \\ \forall l=1...h}} \tag{23}$$

We calculate the quality of service, using [Eq. 24], for each attribute l on provider resource j. We note this variable as Q_{jl}. The quality of service as a function of load behaves as a piecewise function where each provider resource has a maximum load before deterioration L_{jl}^M as reflected in [Eq.24]:

$$Q_{jl} = \begin{cases} Q_{jl}^M & if\ L_{jl} \leq L_{jl}^M \\ Q_{jl}^M e^{\frac{L_{jl}^M - L_{jl}}{1 - L_{jl}}} & if\ L_{jl} > L_{jl}^M \end{cases} \tag{24}$$

The workload on provider resource j for a specific attribute l is calculated [Eq.25] as the sum of the values of the attributes of the resources located on the servers divided by the provider resource capabilities.

$$L_{jl} = \frac{\sum_{k=1}^{n} C_{kl} \times X_{ijk}}{P_{jl}} {\scriptstyle \substack{\forall i=1...g \\ \forall j=1...m \\ \forall l=1...h}} \tag{25}$$

The last objective is the sum of the costs of implementing the reconfiguration plan [Eq.26] as expressed by the third term of the aggregate objective function [Eq.15]. The size of the reconfiguration plan is an estimate based on the current allocation X_{ijk}^t and the next allocation X_{ijk}^{t+1} provided by the optimization process.

$$\sum_{k=1}^{n} (X_{ijk}^t - X_{ijk}^{t+1}) M_k, {\scriptstyle \substack{\forall i=1...g \\ \forall j=1...m}} \tag{26}$$

First, we address this optimization with the help of a Java-based constraint solver, Choco Solver [25], but would like to stretch the optimization beyond such a constraint resolving algorithm. Consequently we propose alternate solutions and compare them with different algorithms including a Round Robin [26] algorithm already tuned for cloud resource allocation where virtual machines can be allocated and sorted by affinity. Since we address an optimization involving multiple criteria, multiple constraints and multiple objectives, we obviously turn to evolutionary algorithms well suited for the purpose. We propose the use of NSGA-II [27] and NSGA-III [28] that we extend and improve with a tabu search [29]. In this case, tabu search is a meta-heuristic that guides a local heuristic search procedure to explore the solution space beyond local optimality by moving virtual machines on different servers. We improve NSGA-III with a constraint checking process as described in Figure 3. Each constraint (capacities constraint, affinity and anti-affinity constraints) is checked during the evaluation process. We also include the constraint solver (NSGA with constraint solver) previously mentioned in the comparison.

We have considered using a classical mono-objective genetic algorithm because it is easier to apply a weighting coefficient on the objectives. In our use case, the Pareto frontier need to be found completely. It is enough to find the one point on the Pareto frontier that "is preferred" by decision makers. Decision maker should not intervene in the decision-making process for full automation of the platform. In a single-objective definition it a weighting coefficient is easier to be applied on the objectives since these preferences may change over time.

In Table II we summarize the advantages and disadvantages of existing algorithms. We focus on NSGA because it is the only one capable of achieving scalability. We will improve NSGA to ensure that this algorithm can take into account the users' requests and their constraints. The O symbol indicates that we worked mainly on these three needs.

TABLE II
COMPARISON OF ALLOCATION ALGORITHMS FOR CLOUD RESOURCE

	Round Robin	Constraint Programming	NSGA	Filtering Algorithm
Compliance with constraints	✗	✓	O	✓
Resource Scalability	✗	✗	✓	✗
Compliance with customer requests	✗	✗	O	✗
Control over infrastructure	✗	✓	O	✗

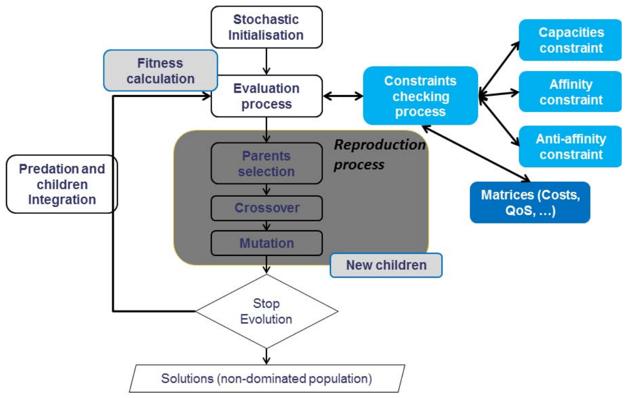

Fig. 3. Various process of our modified NSGA algorithm

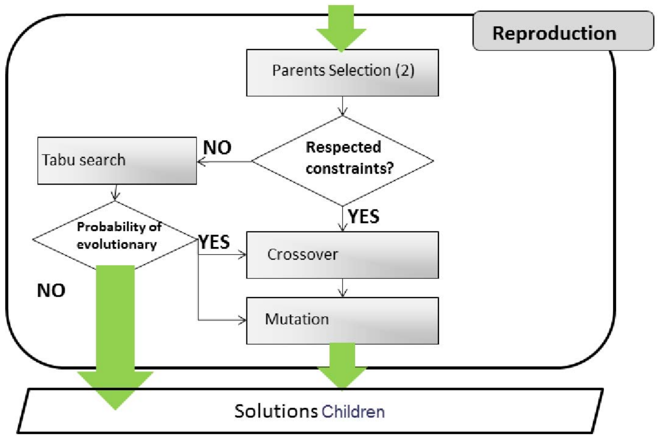

Fig. 4. NSGA-III enhanced with tabu search in reproduction process

Since evolutionary algorithms can hardly handle strict constraints, we resort to methods known to address this weakness such as:

- excluding the individuals that are not in line with the constraints;
- fixing faulty individuals through a repair process;
- preserving the correct individuals through special operators;
- modifying the search space and guiding the algorithm all the way through (via hyperplanes).

In our work, we use the first two methods to meet strict constraints. Method 1, which excludes out-of-limit individuals, reveals to be inefficient because it excludes too many individuals. Method 2, which we adopt in our work, is designed to fix faulty individuals through an associated tabu search algorithm. The repair process through a combined enhanced tabu search and a genetic algorithm (NSGA-III) is described in Figure 4 and implemented through the repair function in Figure 5. We attempted to use a constraint violation penalty for individuals but this happened to lead to serious increases in response times whereas our primary goal was to obtain a response in a very short timeframe (<2mn). In some challenging cases, the algorithm would result in no solution found yet even after having computed for a whole week.

Figure 4 describes the repair process of individuals. If the two selected parents do not respect users constraints, then they are treated by the tabu search algorithm. Once repaired, they can continue to evolve through crossover or mutation.

The repair process through tabu search is made possible through the browsing of a list of potential hosts (servers) for virtual machines (Figure 6) hosted on overloaded servers. The repair process is launched whenever invalid individuals are assessed. The fixing method aims at making the individuals compliant with the constraints. Every faulty gene found within an individual is then processed and modified accordingly.

Each individual possesses chromosomes here standing for virtual machines. Each gene stands for a server ID, indicating where the virtual machine is hosted. To this end, we use SBX

```
1: procedure REPAIR(I)                                   ▷ I is an individual
2:     $serversError \leftarrow exceedingDetection(I)$   ▷ we are scanning for servers where constraints are exceeded
3:     for $i \neq numberOfVM()$ do
4:         $server \leftarrow getServerOfVM(I, i)$
5:         if $server \in serversError$ then
6:             $I(i) \leftarrow findNeighbour(I, i)$
7:         end if
8:     end for
9: end procedure
```

Fig. 5. Tabu search for the repair process of NSGA individuals

```
1: procedure FINDNEIGHBOR(I, i)          ▷ i is an ID of a VM
2:     for $j \neq numberOfServer(I)$ do
3:         if $isValidAllocation(i, j)$ then
4:             $return(j)$                  ▷ We return a valid server
5:         end if
6:     end for
7: end procedure
```

Fig. 6. Finding the nearest valid neighbor

and PM standard. The repairing of individuals was integrated through a tabu search. The tabu search is launched whenever a chromosome goes out of the given constraint framework. While using a Euclidian approach, we choose the solution that is found closer to the ideal point where cost and rejection rate are the next to naught.

IV. EVALUATIONS

We compare the optimization performance of our main algorithm, NSGA-III with a Tabu search, to repair individuals and genes and enhance performance for the joint users' and providers' benefit, with the following algorithms:

- Round Robin
- Constraint Programming
- unmodified NGSA-II
- unmodified NSGA-III

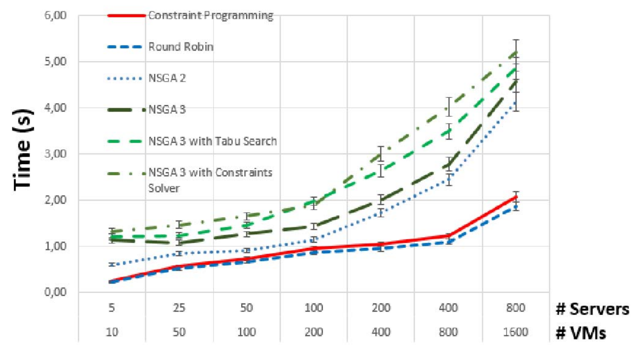

Fig. 7. Average execution time performance comparison with few resources. NSGA-III with Tabu Search is less efficient on small problems

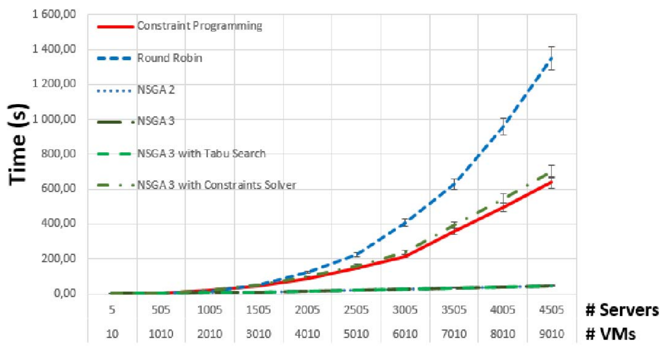

Fig. 8. Average execution time performance comparison with many resources. NSGA-III with Tabu Search is scalable algorithm

- NSGA-III with constraints solver
- NSGA-III with Tabu search (our proposed solution)

We compare the algorithms in terms of : a) execution time, b) user requests rejection rate, c) number of violated constraints and d) incurred cost on the providers. All conditions, scenarios, requests and infrastructures are randomly generated with parameter configurations that reflect typical infrastructures sizes and cloud provider practices.

First, we compare the average computing (execution) time for all algorithms for scenarios involving up to 800 servers and 1600 virtual machines. These values reflect the typical sizes that provider manage simultaneously as clusters or blocks when relying on cloud stacks and managers such as OpenStack. Providers prefer handling several smaller platforms instead of managing oversized infrastructures and installations. The sizes taken into account in our performance and comparison, in the order of a thousand nodes, are sufficient to build an understanding of the relative performance of the algorithms and enough to demonstrate that the NSGA-III with the Tabu search provides the best overall tradeoff in optimization criteria, in compliance with user expressed constraints while taking into account both the users and providers conflicting interests.

Parameter	Value
populationSize	100
Number of evaluations	10000
sbx.rate	0.70
sbx.distributionIndex	15.00
pm.rate	0.20
pm.distributionIndex	15.00

TABLE III
NSGA II AND III SETTINGS

All the simulations are performed on a commercial off-the-shelf, not particularly powerful, PC specifically an Intel NUC with Intel Celeron 847 (1.1 GHz - 2 MB Cache) and 4GB DDR3 RAM. The parameters used by NSGA-II and NSGA-III in the simulations and assessments are listed in Table III. The results correspond to an averaged performance over 100 runs across all randomly generated scenarios.

Figures 7 through Figures 11 report the results of performance comparison of all evaluated algorithms for increasing problem size. In order to assess the relative performance of the algorithms and rank them, it is essential to look at these results jointly and identify the algorithms with the best tradeoff and/or balanced performance. Clearly the Round Robin and constraint based programming algorithms are faster than the evolutionary approaches as depicted in Figure 7. The evolutionary algorithms need to produce good genes by generating new individuals and new populations and this is known to be a time consuming process (making them 2 to 3 times slower; 5 seconds versus 1.5 seconds). They conduct deeper exploration and exploitation to find multiple feasible solutions. In Figure 8 we see that the constraint propagation algorithms, round robin and NSGA-III improved with constraint propagation algorithm doesn't scale with the resolution time criterion. Unmodified NSGA-II and NSGA-III, have a low response time, but do not respect the constraints. We find that the NSGA-3 algorithm to obtain a solution in short time on problems where many resources are requested.

This can be seen by analyzing the results of Figure 9 where the Round Robin and unmodified NSGA algorithms reject many more requests than the evolutionary algorithms. Rejection rate is defined as the number of requests with unviolated constraints divided by the total N number of users requests to be processed in one case. The NSGA-III with the Tabu Search, as proposed, outperforms all other algorithms in terms of acceptance rate and is designed to generate the largest revenues for the providers while meeting the interests of both actors, consumers and providers. The unmodified NGSA algorithms and the NSGA-III with the constraint solver all improve the rejection rate but remain too weak to repair genes and individuals of the NSGA-III that outperforms all algorithms.

Figure 10 shows clearly that the weakness of the unmodified evolutionary algorithms, NSGA-II and III, is the inability to respect or meet constraints. All other algorithms did not violate the constraints for the evaluated scenarios. Both NSGA-II and III did not use all the resources to find solutions while all other algorithms used, all the provider resources (or servers), to host the requests. Figure 10 shows only two types of bars because there are only two algorithms (NSGA-II & NSGA-III)

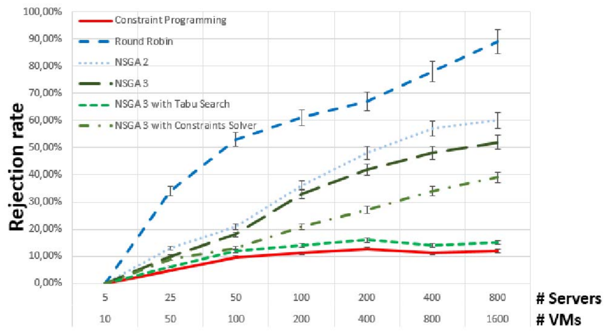

Fig. 9. Rejection rate comparison with increasing problem size. NSGA-III with Tabu Search is too close from the optimal solution

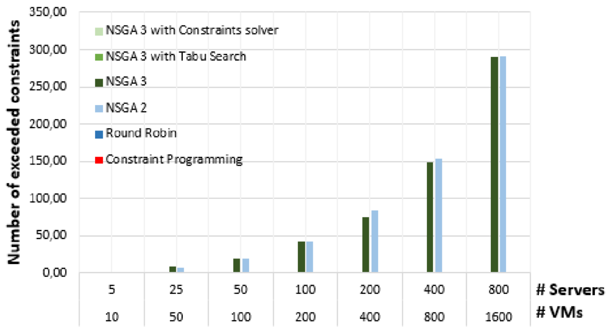

Fig. 10. Violation of constraints performance for increasing problem size. NSGA-II and III generate constraints violations

that generate constraint violations

Finally, the results of Figure 11 conclude this performance evaluation and comparison by reporting the cost induced on the providers to host the consumer requests. We meant to discriminate the operating costs related to the infrastructure from the platform usage costs. Clearly the unmodified evolutionary algorithms incur high cost. The Constraint Programming, the NSGA-III with constraint solver and the Tabu Search induce the lowest cost penalty to the providers. This means that the NSGA-III with tabu search accepts more requests while maintaining provider hosting costs at levels similar to those reached in constraint programming which conversely rejects a greater number of demands (for which no penalty for rejection is added, thus creating a misleading impression that this method performs best).

In summary, the NSGA-III with tabu Search, our choice, outperforms all other algorithms when all the criteria are taken into account or jointly accounted for. Further, this proposed algorithm scales better and handles multiple objectives and constraints.

V. Conclusion

We propose an evolutionary algorithm combined with a tabu search (NSGA-III with tabu Search) to address the cloud resource allocation problem while taking account of the interests of both consumers or users and providers. In

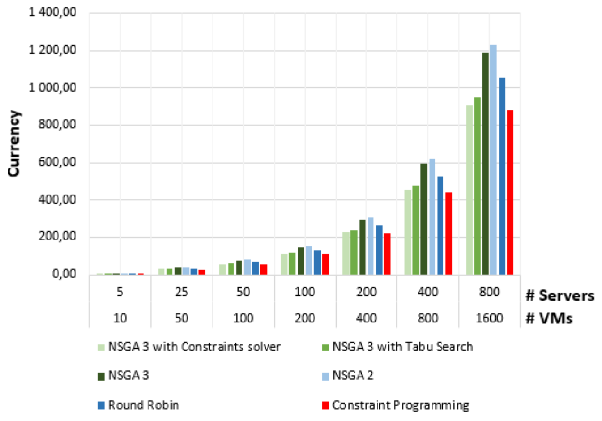

Fig. 11. Average cost induced by each algorithm for the providers. NSGA-III with Tabu Search at higher costs than optimal albeit still reasonable

order to assess the performance improvements and possible performance degradations for some criteria and achievable tradeoffs of the NGSA-III with tabu search, we compare the algorithm with other state-of-the-art algorithms, especially when based on constraint programming.

Simulation results show that the proposed algorithm outperforms the other algorithms with marginal increase in execution time and induced provider costs that are far outweighed with the algorithm gains. The provided benefits in increased revenues for the provider are due to reduced rejection rates, better resource usage and the ability to meet simultaneously or jointly both users and providers' interests. The approach has also the advantage of easy integration of our twofold objective unlike constraint programming, for example, where the inclusion of multiple objectives happens to be complex and hard to express and achieve. The combination of NSGA-III with a tabu search that repairs genes and individuals produces the desired benefits, unlike unmodified evolutionary algorithms such a NSGA-II and III and even NSGA-III with constraint programming.

In future work, we may consider using a genetic algorithm to handle plateform and flow events (user requests, platform failures, etc.). Moreover, we will need to create a normalized and standardized metric on a cost per request basis to propose a better solution in an effort to compare all algorithms in all scenarios.

References

[1] Anuradha, V.P.; Sumathi, D., "A survey on resource allocation strategies in cloud computing," in Information Communication and Embedded Systems (ICICES), 2014 International Conference on , vol., no., pp.1-7, 27-28 Feb. 2014 doi: 10.1109/ICICES.2014.7033931

[2] Beloglazov, A. and Buyya, R. (2012), Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers. Concurrency Computat.: Pract. Exper., 24: 13971420. doi:10.1002/cpe.1867

[3] Ouarnoughi, H.; Boukhobza, J.; Singhoff, F. & Rubini, S. A Cost Model for Virtual Machine Storage in Cloud IaaS Context 2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP), 2016, 664-671

[4] Anton Beloglazov and Rajkumar Buyya. 2010. Energy Efficient Resource Management in Virtualized Cloud Data Centers. In Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGRID '10). IEEE Computer Society, Washington, DC, USA, 826-831. DOI=http://dx.doi.org/10.1109/CCGRID.2010.46

[5] Yu-Ju Hong, Jiachen Xue, and Mithuna Thottethodi. 2011. Dynamic server provisioning to minimize cost in an IaaS cloud. In Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems (SIGMETRICS '11). ACM, New York, NY, USA, 147-148. DOI=http://dx.doi.org/10.1145/1993744.1993799

[6] Feller, E.; Rohr, C.; Margery, D.; Morin, C., "Energy Management in IaaS Clouds: A Holistic Approach," in Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on , vol., no., pp.204-212, 24-29 June 2012 doi: 10.1109/CLOUD.2012.50

[7] Knauth, T.; Fetzer, C., "Energy-aware scheduling for infrastructure clouds," in Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on , vol., no., pp.58-65, 3-6 Dec. 2012 doi: 10.1109/CloudCom.2012.6427569

[8] Lampe, U.; Siebenhaar, M.; Papageorgiou, A.; Schuller, D.; Steinmetz, R., "Maximizing Cloud Provider Profit from Equilibrium Price Auctions," in Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on , vol., no., pp.83-90, 24-29 June 2012 doi: 10.1109/CLOUD.2012.19

[9] Toosi, A.N.; Calheiros, R.N.; Thulasiram, R.K.; Buyya, R., "Resource Provisioning Policies to Increase IaaS Provider's Profit in a Federated Cloud Environment," in High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on , vol., no., pp.279-287, 2-4 Sept. 2011 doi: 10.1109/HPCC.2011.44

[10] Van den Bossche, R.; Vanmechelen, K.; Broeckhove, J., "Cost-Optimal Scheduling in Hybrid IaaS Clouds for Deadline Constrained Workloads," in Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on , vol., no., pp.228-235, 5-10 July 2010

[11] Mattess, M.; Vecchiola, C.; Buyya, R., "Managing Peak Loads by Leasing Cloud Infrastructure Services from a Spot Market," in High Performance Computing and Communications (HPCC), 2010 12th IEEE International Conference on , vol., no., pp.180-188, 1-3 Sept. 2010

[12] Anshul Rai, Ranjita Bhagwan, and Saikat Guha. 2012. Generalized resource allocation for the cloud. In Proceedings of the Third ACM Symposium on Cloud Computing (SoCC '12). ACM, New York, NY, USA, , Article 15 , 12 pages. DOI=http://dx.doi.org/10.1145/2391229.2391244

[13] Fabien Hermenier, Julia Lawall, Gilles Muller. BtrPlace: A Flexible Consolidation Manager for Highly Available Applications. IEEE Transactions on Dependable and Secure Computing, Institute of Electrical and Electronics Engineers (IEEE), 2013, 10 (5), pp.273-286.

[14] Srinivasa, K.G.; Kumar, K.S.; Kaushik, U.S.; Srinidhi, S.; Shenvi, V.; Mishra, K., "Game theoretic resource allocation in cloud computing," Applications of Digital Information and Web Technologies (ICADIWT), 2014 Fifth International Conference on the , vol., no., pp.36,42, 17-19 Feb. 2014 doi: 10.1109/ICADIWT.2014.6814667.

[15] Anshul Rai, Ranjita Bhagwan, and Saikat Guha. 2012. Generalized resource allocation for the cloud. In Proceedings of the Third ACM Symposium on Cloud Computing (SoCC '12). ACM, New York, NY, USA, , Article 15 , 12 pages. DOI=http://dx.doi.org/10.1145/2391229.2391244

[16] Chunqiang Tang, Malgorzata Steinder, Mike Spreitzer, Giovanni Pacifici: A scalable application placement controller for enterprise data centers. WWW 2007: 331-340

[17] I Giurgiu, C Castillo, A Tantawi, M Steinder. Enabling Efficient placement of virtual infrastructures in the cloud, Middleware 2012

[18] Asser Tantawi. A scalable algorithm for placement of virtual clusters in large data centers, MASCOT, 2012

[19] Al-Fares, M.; Loukissas, A. & Vahdat, A. A Scalable, Commodity Data Center Network Architecture SIGCOMM Comput. Commun. Rev., ACM, 2008, 38, 63-74

[20] Alizadeh, M. & Edsall, T. On the Data Path Performance of Leaf-Spine Datacenter Fabrics 2013 IEEE 21st Annual Symposium on High-Performance Interconnects, 2013, 71-74

[21] Greenberg, A.; Lahiri, P.; Maltz, D. A.; Patel, P. & Sengupta, S. Towards a Next Generation Data Center Architecture: Scalability and Commoditization Proceedings of the ACM Workshop on Programmable Routers for Extensible Services of Tomorrow, ACM, 2008, 57-62

[22] Dorit S. Hochbaum and David B. Shmoys. 1987. Using dual approximation algorithms for scheduling problems theoretical and practical results. J. ACM 34, 1 (January 1987), 144-162. DOI=10.1145/7531.7535 http://doi.acm.org/10.1145/7531.7535

[23] Antoniou A, Iosup A (2012) Performance evaluation of cloud infrastructure using complex workloads. TU Delft MSc thesis

[24] Suganya Sridharan, A Performance Comparison of Hypervisors for Cloud Computing, University of North Florida,2012

[25] Choco, "http://choco-solver.org/" April 2016

[26] Komal Mahajan, Ansuyia Makroo and Deepak Dahiya, "Round Robin with Server Affinity: A VM Load Balancing Algorithm for Cloud Based Infrastructure," Journal of Information Processing Systems, vol. 9, no. 3, pp. 379 394, 2013. DOI: 10.3745/JIPS.2013.9.3.379.

[27] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," in IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182-197, Apr 2002. doi: 10.1109/4235.996017

[28] Seada, H., Deb, K.: U-NSGA-III: A unified evolutionary algorithm for single, multiple, and many-objective optimization. COIN Report Number 2014022, Computational Optimization and Innovation Laboratory (COIN), Electrical and Computer Engineering, Michigan State University, East Lansing, USA (2014)

[29] F. Glover, "Tabu Search-Part I", ORSA J. Computing, vol. 1, pp. 190-206, 1986