A Pareto-based Genetic Algorithm for Optimized Assignment of VM Requests on a Cloud Brokering Environment

Yacine Kessaci, Nouredine Melab, El-Ghazali Talbi

A Pareto-based Genetic Algorithm for Optimized Assignment of VM Requests on a Cloud Brokering Environment

Yacine Kessaci, Nouredine Melab
and El-Ghazali Talbi
INRIA Lille, CNRS/LIFL, Université Lille 1.
Parc Scientifique de la Haute Borne,
6 rue Héloise Bât.B, Park Plaza,
59650 Villeneuve d'Ascq, France.
Email: {yacine.kessaci, nouredine.melab, talbi}@lifl.fr

***Abstract*—In this paper, we deal with cloud brokering for the assignment optimization of VM requests in three-tier cloud infrastructures. We investigate the Pareto-based meta-heuristic approach to take into account multiple client and broker-centric optimization criteria. We propose a new multi-objective Genetic Algorithm (MOGA-CB) that can be integrated in a cloud broker. Two objectives are considered in the optimization process: minimizing both the response time and the cost of the selected VM instances to satisfy the clients and to maximize the profit of the broker. The approach has been experimented using realistic data of different types of Amazon EC2 instances and their pricing history. The reported results show that MOGA-CB provides efficiently effective Pareto sets of solutions.**

***Keywords*-cloud computing; cloud brokering; scheduling; VM instances; VM requests; multi-objective optimization; genetic algorithm; client satisfaction**

I. Introduction

Cloud computing is an emerging computer science paradigm of distributed computing in which applications, data and infrastructures are proposed as a service that can be consumed in a ubiquitous, flexible and transparent way. This facilitation helped to develop the notion of the resources on demand with pay-as-you-go as an economic model. The computing paradigm can be modeled in two ways. In the first way there is a direct link between the client and the infrastructure service provider. This type of cloud model is a two-tier model. The other type called three-tier model, includes another tier that plays a mediatory role between the client and the provider. The name of this new actor is the broker. The growth of cloud computing paradigm is mainly caused by the trend of reducing the total cost of ownership (TCO). Indeed, the client and/or the broker tiers aim to be released from the infrastructures constraints and therefore form their expenditures.

A lot of works [1], [2] have been proposed to enhance the performances of the two-tier model in different fields. The major issues that arise from this type of cloud models without a broker tier are the high variability in the quality of service proposed by the infrastructure service providers. Indeed, in a two-tier model the provider has the full control over the services and the prices that he/she proposes to the clients. Therefore, since each tier tries to maximize his/her own profits, the client, powerless, follows the market law. Besides, another reason for this variability is due to the existence of a multitude of infrastructure cloud providers (Amazon, Rackspace, ...) which adds concurrency between them taking the clients as hostage. In addition, because of their number and the different instances features that they propose, the client is unable to know or to choose the good instances at the right moment. Hence, one can see the interest of having a new tier (broker), to find a tradeoff between the two parts. The model used in economics to represent the needs of both the client and the provider is called the utility model. This model depicts the relationship between the provided resources and their needs. Some works such as [3], [4] used this model in computer science and gave birth to a new concept called the *utility computing*.

In this paper, we use this notion of utility computing and present a new work that tackles two main parameters that affect cloud brokering, the prices of VM instances and their response time. The objective is to minimize those objectives in order to give the best quality of service (QoS) to the clients by reaching their satisfaction rate while providing an interesting profit for the cloud broker. Indeed, we propose a multi-objective genetic algorithm for cloud brokering (MOGA-CB). It provides a set of Pareto optimal assignments by dispatching the client's virtual machines (VM) requests over the best combination of instances with the minimum cost and response time. This approach uses information provided by the infrastructure service provider (e.g. Amazon) to retrieve the prices of those instances and their different performances to reach its objectives. MOGA-CB makes this possible by its capacity as a metaheuristic, to explore a wide range of potential solutions to the problem.

The main contributions of this paper are the following:

- Using a Pareto multi-objective genetic algorithm for cloud brokering assignment to deal at the same time with the two main criteria that compose the broker's profit and the clients' satisfaction.

- Analyzing the different behaviors of MOGA-CB through different configurations and highlighting the relationship between the satisfaction/profit results and the addressed objectives.
- Simplifying through the Pareto multi-objective genetic approach the utility theory in economics [5] that proposes a client's satisfaction modeling.

The remainder of the paper is organized as follows. In Section II we present the related work. Section III presents the system, satisfaction and profit models used in our problem modeling. Our approach is presented in Section IV. The results of our experimental study are discussed in Section V. The conclusion is drawn in Section VI.

II. Related Work

Several works have been proposed in the field of the cloud computing to deal with different issues. Those issues let the works be classified in two categories. The first category is derived from the grids and for which the major goal is performance improvement. The cloud model is thus a two-tier model, where the provider proposes directly the services to the clients. The economic model is static, like in [6], where two algorithms based on a pricing model are proposed. They both use processor sharing in order to balance between conflicting objectives (profit and resource utilization). In [7], *Burge et al*. describe an allocation method for heterogeneous machines that maximizes the profit by affecting the requests to the machines according to their energy cost. Other approaches based on genetic algorithms and dealing with profit are presented in [8] and [9]. In [9] a linear programming driven genetic algorithm is proposed. In fact, this latter aims to provide the best meta-scheduling in a utility grid based on the idea of minimizing the combined costs of all users in a coordinated way. Yu and Buyya in [8] present a genetic algorithm approach to address scheduling optimization for workflow applications with two QoS constraints (deadline and budget).

However, the marketing of cloud computing paradigm raises a number of issues. Indeed, performance implies expensiveness, mainly when dealing with cloud models where the provider has a full economic control and where the pricing policy is pay-as-you-go. To deal with those issues the second category together with a new model has been introduced. In the latter, a third tier appears and plays the role of an intermediate to find the tradeoff between customers' needs and the providers' profits. A set of works have been conducted over this new model. The work in [10], proposes an optimal virtual machine placement algorithm. The objective is to minimize the costs while assigning VMs in a multiple cloud provider environment. For that, they use a strategy that avoids the two extreme (over and under) assignments. The approach fits the correct resources needs by adjusting the pricing according to the load arrival of VMs.

In [11], the authors outline usage scenarios and a set of requirements for a federation of cloud infrastructures based on RESERVOIR. They also propose an accounting and billing architecture between resource consumers and infrastructure providers to be used within RESERVOIR. The objective is to cope with the migration of virtual machines by managing postpaid and prepaid payment schemes according to the users needs.

In the work presented in [12], the proposed algorithm aims to reallocate virtual machines in a large cloud. The approach consists in identifying only the critical instances using the load trend behavior instead of the thresholds to be used to take decisions.

In [13], the authors present a study in which they compare the VM placement mechanisms over a multi-provider and multi-site cloud. They prove how cloud brokering mechanisms over a multi-provider cloud affects beneficially the VM deployment by improving the performance while lowering the costs. In [14], the authors focus on different scheduling strategies for an optimal deployment across multiple clouds. They present a modular broker architecture based on different scheduling strategies using different optimization criteria, constraints and environmental conditions.

All of the above surveyed works, present different brokering approaches to deal with a set of objectives. However, none of those works consider the relationship between the charged price and the provided performance. They also do not pay attention on how each of those criteria can affect the others. The work presented by *Chen et al.* in [15] deals with those points, by proposing an utility theory based on economics to develop a new model that integrates the client satisfaction over a cloud. Two scheduling algorithms are proposed to find a good tradeoff between the client satisfaction and the provider profit. However, this approach is an aggregation of objectives (i.e. it can only optimize one objective at a time).

Therefore, to deal with all the misses aforementioned, we propose a scheduler using a multi-objective genetic algorithm to optimize the two major objectives that affect both the client's satisfaction and the provider's profit. Those two objectives are, the request response time and the service cost. In other words, our new approach provides a set of Pareto solutions rather than a single solution (i.e. non-dominated solutions), with the best tradeoff between the quality of service (QoS) and the profit.

III. Distributed Cloud Scheduling Model

A. System Model

In this work, an Infrastructure As A Service (IAAS) cloud model is considered. Indeed, we use a three-tier architecture where the tiers represent the clients, the distributed cloud provider and the cloud broker (see Figure 1). The role of the second tier (the broker) is to find the best configuration among the resources proposed by the provider to fit the client

VM requests. Therefore, the broker charges the client for this service. Indeed, in our model we define two types of services. The first one, called the infrastructure service, is related to the provider. The second provided by the broker is called the business service. The prices of instances differ according to offers and demands. The VM instances are proposed for rental by the infrastructure service provider. Depending on the offers that the broker makes over those VM instances the prices may change. The broker reserves itself according to the demands of the clients. The variation of the prices of the VM instances is bounded by a time duration called *scheduling round* after which the instance's price does not change. As for the major economic cloud models the reservation time of the resources is divided into slots. Hence, the user has to pay for the whole time of the reserved slot even if his/her effective time usage of the resources is less than the time slot. The provider model is inspired from the Amazon EC2 Spot [16].

Moreover, the clients in our model send VM resource requests to the broker for their own application usage. Indeed, this work aims to provide the best VM instances combination to host those client's application. This combination (assignment) targets to minimize both the price to pay for the client and the response time for their application. Each request is a quadruplet (*size*, S_{rate}, α, β), where *size* represents the size of the application in terms of CPU computation time, S_{rate} is the satisfaction flexibility of the client. In other words it represents the roughness of the client (i.e. a high value means a demanding client while a small value is for a more relax one). This value is a rate over the best obtained solution by the broker. α and β designate the preference that the client has for respectively the service price or response time.

The originality of this approach is to propose a scheduling (assignnment) algorithm that uses a multi-objective genetic algorithm in order to find the best choices of VM instances among the ones proposed by the infrastructure service provider. The objectives considered in our algorithm are the cost and response time of the instances. Those two objectives are the main parameters for expressing the client's satisfaction and to deduce the broker's profit.

Indeed, our algorithm aims to provide the best Quality of Service (QoS) by satisfying as much as possible the client, while helping the provider to maximize his/her profit. The optimization of the objectives is due to the heterogeneity of the requests features, the fluctuations of the prices and the difference in performance of the instances proposed by the provider.

B. Satisfaction and Profit Model

Several state-of-the-art works deal with QoS focusing on only one criterion which is either the response time or the amount of satisfied requests. The work in [15] based on the utility theory in economics [5] proposes a client's satisfaction modeling. Equation 1 defines the client satisfaction, it is based on the service price *p* and the response time *t*.

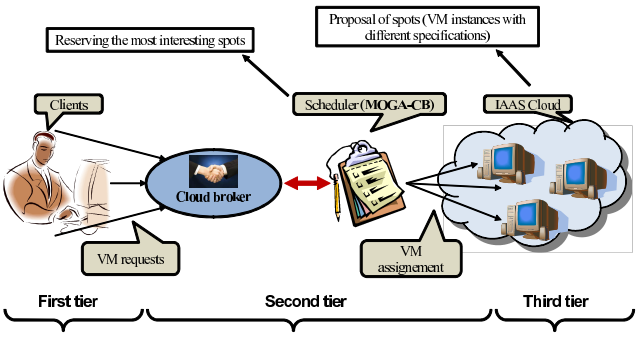

Figure 1. A representation of our three-tier cloud model.

$$Satisfaction(p,t) = S_{max} - \alpha p - \beta t \quad (1)$$

where S_{max} is the maximum satisfaction value that the broker can deliver for the client. The α and β values are used to give more importance to one of the criteria (the price or the response time) while not changing the client satisfaction. The value α/β or β/α is known as marginal rate of substitution in economics. Since our work is based on a Pareto approach, the (α, β) parameters help only to make a choice of the optimal solution that fits better the client's needs. We will also prove latter that thanks to the Pareto approach we can simplify this formula by removing the α and β values from the client's request parameters without changing the client's satisfaction.

In addition, to compute the broker's profit, we deducted Equation 2 from Equation 1. Indeed, with the parameter S_{rate} presented previously we can deduce the satisfaction needed by the client. Therefore, knowing the response time given by the provider, the (α, β) parameters and the maximum satisfaction proposed by the broker S_{max} we can deduce the profit generated by the broker while providing his/her business service to the clients.

$$profit = (S_{max} - Satisfaction \times S_{rate} - \beta t)/\alpha \quad (2)$$

C. Problem Description

In our cloud model, we deal with a three-tier architecture. The first tier is the clients with J VMs requests for running their applications. The second tier is the broker which provides business services. The third tier is infrastructure service provider which proposes N VM instances for rent. The broker has to find the best assignment of VM requests by choosing the right combination of the proposed instances based on their current price, their performance index PI and the load in terms of client's VM requests. The performance index PI is a normalization value that helps to compare the performances of the different instances

when computing the same task. The different values of PI of the used instances in this work are drawn in Section V. The problem consists then of assigning (scheduling) J client requests on N combinations of VM instance types. Moreover, we know that the task scheduling problem is generally NP-hard [17]. Therefore, the brokering problem is NP-hard as well. Thus, a metaheuristic algorithm (genetic algorithm) appears to be an appropriate approach to deal with the problem.

In our model, the client submits requests with QoS requirements. Those requests are defined by two types of parameters. The first ones are fixed by the client at the submission of his/her requests, while the others are deduced and used by the algorithm. The requirements of the user as said previously, are the request size, the satisfaction rate and the α, β values designated for a VM request j by the tuple $(size_j, S_{rate,j}, \alpha, \beta)$. The other types of parameters are defined during the scheduling process. They serve to inform about the state of the request during the process. The first variable $cost_{ji}$ represents the cost of the VM request j when assigned to the VM instance i with the current price of the instance for the scheduling round duration (no price fluctuation during each scheduling round). The other variable rpt_j represents the remaining processing time of the request j on a standard instance. The initial value of this variable is the $size_j$ parameter. The rpt_j value over an instance i noted rpt_{ji} is given by Equation 3, where PI_i is the performance index of the instance i. This value is updated only in case where the request is assigned to another instance before the end of its execution.

$$rpt_{ji} = rpt_j / PI_i \tag{3}$$

The objective functions of our approach aim to **minimize** the total request cost designated by $MinCost$ and the response time of the requests designated by $MinRT$ in order to provide the best client satisfaction and broker profit. It is formulated as follows:

$$MinCost = \sum_{j}^{J} \sum_{i}^{N} cost_j + rpt_{ji} \times price_i = \sum_{j}^{J} \sum_{i}^{N} cost_{ji} \tag{4}$$

where for Equation 4, $cost_j$ is the previous accumulated cost of the request j, rpt_{ji} is the remaining processing time of the request j if processed on the instance i, and $price_i$ is the instance price during the current scheduling round of the instance i.

$$MinRT = \sum_{j}^{J} \sum_{i}^{N} currentTime - arrivalTime_j + rpt_{ji} \tag{5}$$

where for Equation 5, $currentTime$ is the current time, $arrivalTime_j$ is the arrival time of the request j and rpt_{ji}, as in Equation 4, is the remaining processing time of the request j on the instance i, if not concerned by a reassignment.

IV. The Multi-objective Genetic Algorithm for Cloud Brokering

A. Problem Encoding

We propose an encoding for the MOGA-CB solutions as illustrated in Figure 2.

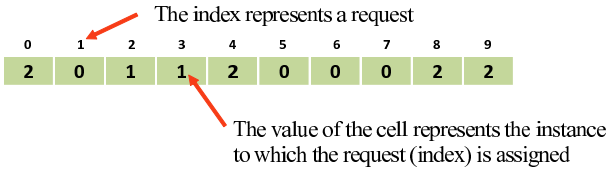

Figure 2. Problem encoding.

Figure 2 represents one possible scheduling among plenty proposed by the multi-objective genetic algorithm. In the proposed example we identify three major specifications. The indexes of the table depict the requests that are scheduled, the number in each cell identifies the instance to which the request is allocated. In other words, the first cell represents the first request of the pool that is currently handled by MOGA-CB. In this case, this request is allocated to the instance 2. The second request is allocated to the instance 0 and so on. This encoding informs about the number of requests currently addressed and that have rpt value different from 0 (i.e. not finished requests), which is 10 in our example. This encoding helps us also to deal with the characteristics of our problem. Indeed, it allows one to schedule all the requests of the pool by assigning each one to only one instance at a time. But an instance can be chosen for more than one request. Note that not all the instances are necessarily used in each solution. We also assume that the infrastructure's provider has always available instances. This is realistic since our approach can be applied for a model with several providers, we can afford then to do not deal with the availability of the provider's resources.

B. Population Initialization

The generation of the initial solution in a genetic algorithm is an important phase. In fact, this step affects the future results quality. In our approach, the initialization of the population is done in two steps. The first step checks the number of not yet finished requests from the previous scheduling round and adds them to the initial solution by assigning them to their previous instances. The second step starts only when the first step is finished or when the first step is not necessary. This can happen for example during the first iteration of the algorithm where no request from

the previous scheduling round is being processed. The role of the second step consists of initializing the rest of the solutions of the population. Hence, it affects the new arrived requests by assigning them to the instances randomly.

C. Scheduling Steps

Before each scheduling, MOGA-CB waits for a fixed period of time called scheduling round as previously said. This period helps to gather a pool of VM requests in order to have a larger choice in the assignment and thus to optimize the future scheduling. In addition, this scheduling round is used by the algorithm to retrieve the current instances' prices.

Once these phases done the pool of requests is managed by MOGA-CB to find the best possible assignments over the different instances. The result of the initialization and the execution is stored as a Pareto archive. Once the set of Pareto solutions (assignments) is proposed, the algorithm chooses one scheduling among this set according to the user's parameters to fit at best his/her satisfaction. The chosen solution from the Pareto set is also used as a state to update the algorithm parameters and to inject the not yet finished requests again to follow their execution during the next scheduling round. This solution will be a basis for the next iteration of the following pool of requests. The algorithm will make another execution on this pool in addition to the new pool of arrived requests. The algorithm keeps iterating until no more requests arrive and no more requests still processing. All the scheduling steps are drawn in Figure 3. The evolutionary core of the MOGA-CB algorithm is based on the NSGAII multi-objective genetic algorithm. Indeed, the method used in the MOGA-CB to rank the individuals of the population, because of the multi-objective context, is the dominance depth fitness assignment. This archive contains the different non-dominated solutions generated through the generations. Besides, the selection process of our genetic algorithm is based on two major mechanisms: elitism and crowding. They allow respectively the convergence of the evolution process to the best Pareto front and maintaining some diversity of the potential solutions.

D. Solution Selection Mechanism

The results obtained using MOGA-CB are stored in a Pareto set. Hence, skipping from processing a pool of requests to a new pool, becomes difficult because of the number of solutions. Therefore, in our scheduling algorithm there is a selection step which comes right after the end of the genetic algorithm execution. This step aims to pick up a solution among the Pareto set in order to update the variables of the remaining processing requests from the last scheduling round. This state will be the starting point from which the next execution of MOGA-CB will schedule a new pool of requests. The idea behind choosing a Pareto approach in our work is to propose to the broker as more non-dominated

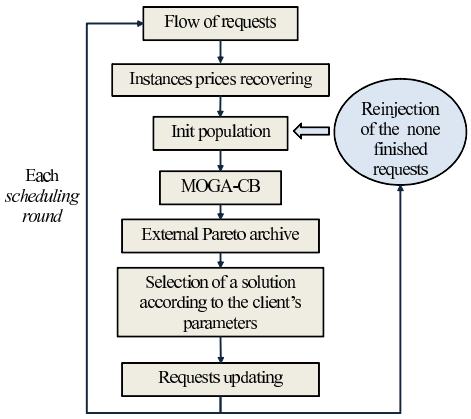

Figure 3. The Flowchart of the MOGA-CB algorithm.

solutions as possible. Each one of these solutions is better than the other regarding a certain objective. The mechanism of the solution selection is based on Equation 1. Indeed a solution in our algorithm is defined with its price p and response time t. The solution that is chosen from the Pareto set is the one that gives the best satisfaction based on α and β parameters in Equation 1. Furthermore, the MOGA-CB algorithm proposes a parameter to promote either the broker's profit or the client satisfaction. Indeed, based on the selected Pareto solution the algorithm can even satisfy the broker requirements by providing him/her for example a certain benefit rate based on the Pareto optimum solution cost, or increases the client satisfaction by providing him/her the requested satisfaction rate. The default setting of the parameter always satisfies the clients. This is due to the fact that this default setting does not lead to a lack in the optimization quality of the broker's profit. Furthermore, the interest of disabling the default setting parameter can be found for example in the case where the broker is in deficit. He/She can then temporary choose the profit option despite the unsatisfied client to recover a correct budget level.

V. Experiments and Results

This section presents the results obtained from our experimental study. The experiments aim to demonstrate and evaluate the contribution of the multi-objective evolutionary approach over the α β tradeoff and the different behaviors that can have the MOGA-CB algorithm according to the variations of the input parameters, the performance indexes and the prices of the instances.

A. Experimental settings

The experimental settings concern both the client and the infrastructure provider sides of our three-tier model. The client side with the VMs requests and the provider side with the rented instances.

- **VM requests' settings:** concerning the inputs of the broker's scheduler we generated VM requests arrivals

according to a Poisson process. Each request arrives in a certain slot of the scheduling round to which it belongs. The scheduling round slot equals 1/10 of the scheduling round. In other words, a request's submission time equals the time of its time arrival (i.e. the time value of the scheduling round to which it belongs) plus the waited slots in this scheduling round. Moreover, the VM request features in our experiments vary according to four parameters. Indeed, as said in Section III-C with the tuple $(size_j, S_{rate,j}, \alpha, \beta)$, the requests differ by their size (execution time), the client satisfaction rate and finally the (α, β) parameters. Therefore, we generated the elements of this quadruplet, where the execution time $size_j$ is a value from [2,50] (this value represents the duration of the request in terms of number of scheduling rounds), the client satisfaction rate $S_{rate,j}$ varies in the interval [0.1,1] where on one hand, the value 0.1 represents the less demanding client (10% of the best obtained satisfaction is enough to gratify him/her) and on the other hand the value 1 is the most demanding client (he/she needs 100% of the best obtained solution). The α parameter varies in the set $\{9, 3, 2, 1\}$ and β in $\{8, 4, 2, 1\}$.

- **Instances settings:** for the infrastructure service provided instances, we used the instances proposed by Amazon EC2 [16]. We used three types of instances the *small* one, the *large* one and *extra large* one. We deduced the performance PI indexes of these instances from the work proposed in [15]. All the parameters' values are drawn in Table I. Regarding the fluctuation of the prices of instances, we downloaded them from the Amazon pricing history [18]. The price fluctuations of all the previously cited types of instance extends over a period of one month on the US California site.

B. Algorithm parameters

In our experimentations, we used some parameters such as the satisfaction/profit selection parameter, the arrival rate of the requests, the request execution time, etc. The satisfaction/profit selection parameter is used in order to promote at a certain time the profit of the broker in case of budget difficulties. Once this option activated, another parameter is added to the algorithm to inform it about the profit rate needed by the broker. We performed experiments with the two state options (enabled and disabled). Regarding the variation of the request arrival rate we used a Poisson process with a λ rate of 15 per scheduling rounds. The number of requests is 10000. In addition, because of the stochasticity of the MOGA-CB approach we performed 30 runs for each configuration. The parameters of the MOGA-CB algorithm are 2000 for the number of generation, 30 for the population size, 1 crossover rate 0.35 mutation rate and 2 for the tournament group size. Table I summarizes the other parameters used in our experiments.

Table I
EXPERIMENTAL PARAMETERS.

Parameter	Value
Number of runs per configuration	30
Number of requests	10000
Request submission time	0.1 to 1 of the scheduling round
Request arrival rate λ	15 per scheduling round
Request execution time	2 to 50 scheduling rounds
Satisfaction rate	0.1 to 1 of the best solution value
Profit rate	0.1 to 1 of the best solution cost
α/β	9, 3, 2, 1, 1/2, 1/4, 1/8, random
Instance types	small, large, extra large
Instance performance indexes (PI)	1, 3.98, 7.12
Instance prices	Amazon EC2 pricing history

C. Performance evaluation

To the best of our knowledge, there does not exist any research work tackling the cloud brokering problem using a Pareto approach. Thus, we perform a bench of experiments with different parameters and configurations. In addition to optimizing the costs and the response times of the VM requests, the approach has to return according to the Pareto solutions proposed by MOGA-CB the solution that meets at best the client's satisfaction. To evaluate our contribution we conducted different experiments to study the behavior of our approach according to different parameters. We conducted experiments in average over all the scheduling rounds to study the general algorithm behavior. Moreover, we also did a real time analysis of the results through evolution of the criteria. We carried out our experiments on 5 different configurations of VM requests. The obtained results were very nearly equivalent. Thus, we discuss only the following configuration.

The results for each configuration of VM requests with its 10000 requests, for each α, β combination and for each profit/satisfaction orientation of the MOGA-CB have been obtained using 30 independent runs. Therefore, the reported results are averaged over these runs. The detailed analysis of our approach is presented in Figure 5 and Figure 4.

The analysis of our approach will be depicted in three parts:

- ***MOGA-CB average behavior:*** Figure 4.**Left** shows that the obtained results over the parameters (profit and satisfaction) do not vary when using our Pareto approach. In other words, the algorithm gives roughly the same results over all the (α,β) settings thanks to the Pareto front. The only parameter that varies a little bit is the computation time of MOGA-CB because of the different complexities which may result from a different solution selection in the Pareto set. In addition, Figure 4.**Middle** shows that the cost and the time response objective complement each other according the client choice trough the α and β parameters. Indeed, when the priority goes to the cost (α bigger than β) the cost

is more minimized therefore the time response suffers from that, same goes for a high time response priority (α lower than β) with opposite behavior. Moreover, we notice that the average results for a random α β configuration gives the same results as a random configuration this is due to our Pareto approach. We can deduce then from the invariability of the satisfaction and the profit results and the similarity of the cost and response time values for the random α β and $\alpha = \beta = 1$ configurations that our Pareto MOGA-CB helps to dispense the client from providing those parameters. The satisfaction model will be as follow: $MaxSatisfaction(p,t) = Min\sqrt{p^2 + t^2}$. Besides, in the experiments for an enabled profit broker option, the clients' satisfaction is highly affected and the disappointment increases along with the increase of the margin of the broker. This proves that there is no interest to use this option except in emergency cases, especially because of the good obtained profit results when disabling this option.

- ***MOGA-CB real time behavior:*** in Figure 5.**Left** we observe the interest of using a Pareto approach to tackle the brokering problem. Indeed, MOGA-CB compensates the instances (spots) cost increasing by a reduction in the instances's response time. The two graphics are complementary when the first increases the second deceases and *vice versa*. In addition, as expected the broker's profit decreases when the instances cost increases (less profit margin). Figure 5.**Right** (Logarithmic scale) shows that there is a tight relation between the number of unfinished VM requests and the instance response time. Conversely, the number of unfinished VM requests is inversely related to the instances cost. This can be explained by the fact that increasing the price of VM instances means in general better instance performances and thus, a better response time and less waiting requests. In Figure 5.**Middle**, the graphics indicate that the client's disappointment and therefore his/her satisfaction is much more affected by the response time of the instances than by their prices. This is explained by the fact that the variation of the instances response time is more significant than their cost. Indeed, the response time depends on the type of instances chosen by the algorithm and on the requests load as well. It also should be noted the general decrease in all the curves of the presented graphics during the course of the interval times. This is mainly due to the decrease in the number of unfinished requests as the algorithm proceeds.
- ***MOGA-CB computation time:*** in this last part represented by the graphic in Figure 4.**Right**. We prove that our MOGA-CB never exceeds 30 seconds in its solution computation time whatever is the load in terms of requests over all the scheduling rounds. This result is interesting since we know that the schedulers waiting time called *scheduling cycle* or *scheduling round* is about 30 seconds. Hence, the computation time of our algorithm is covered by the latter.

VI. Conclusion

In this paper, we presented a new approach for the cloud brokering using a multi-objective genetic algorithm MOGA-CB to minimize the instances' response time, and their cost in order to satisfy the clients and to provide an interesting profit to the broker. This is made possible by exploiting the instances' cost fluctuation and their performances' heterogeneity.

Our new approach has been evaluated using a Poisson process for VM requests arrival. The number of requests was 10000. The experiments stretch over 5 VM requests configuration with 8 α/β configurations and 3 types of instances (spots). The results show that our Pareto-based approach helps to dispose of the α and β parameters while providing a Pareto optimal solution allowing a tradeoff between the response time of the instances and their cost. Moreover, we proved that using a broker profit orientation in our algorithm gives bad results and decreases significantly the client's satisfaction. We also showed that the satisfaction is more related to the instances' response time than to the instances' cost. Furthermore, MOGA-CB assigns efficiently the received requests. It provides effective VM requests assignments before the scheduling round is elapsed.

Besides, the major perspectives of this work is to minimize with more impact the instances' response time and cost by using a better economic model since we proved that the α/β parameters are useless. In addition, we are planning to deploy our broker approach MOGA-CB over a real infrastructure with real Amazon instances for example to validate our contributions over a bigger number of instances with real client satisfaction constraints.

References

[1] Y. Kessaci, N. Melab, and E.-G. Talbi, "A pareto-based metaheuristic for scheduling hpc applications on a geographically distributed cloud federation," *Cluster Computing*, pp. 1–18, 10.1007/s10586-012-0210-2.

[2] N. B. Rizvandi, J. Taheri, A. Y. Zomaya, and Y. C. Lee, "Linear combinations of dvfs-enabled processor frequencies to modify the energy-aware scheduling algorithms," *Cluster Computing and the Grid, IEEE International Symposium on*, vol. 0, pp. 388–397, 2010.

[3] D. Irwin, L. Grit, and J. Chase, "Balancing risk and reward in a market-based task service," in *High performance Distributed Computing, 2004. Proceedings. 13th IEEE International Symposium on*, june 2004, pp. 160 – 169.

[4] B. Chun and D. Culler, "User-centric performance analysis of market-based cluster batch schedulers," in *Cluster Computing and the Grid, 2002. 2nd IEEE/ACM International Symposium on*, may 2002, p. 30.

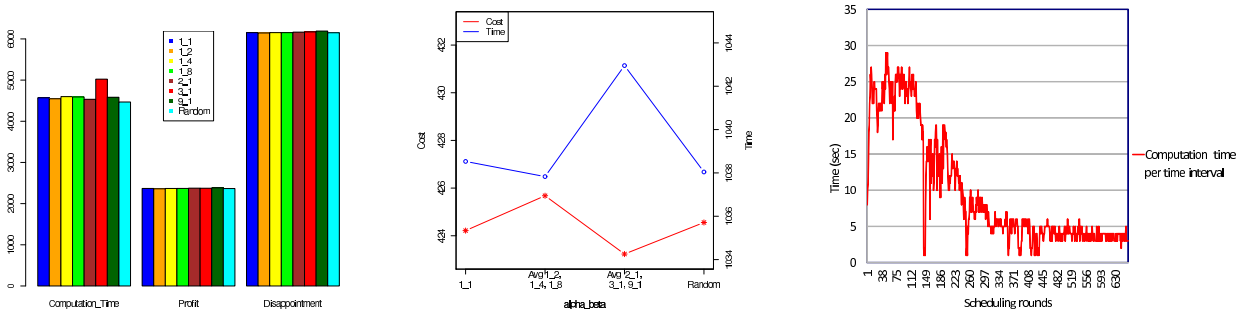

Figure 4. **(Left)**: The none impact of the α,β parameters on the results of the Pareto based MOGA-CB algorithm. **(Middle)**: The tradeoff relationship between the cost and the response time objectives according to the α,β parameters. **(Right)**: The computation time duration of the MOGA-CB algorithm over each scheduling round.

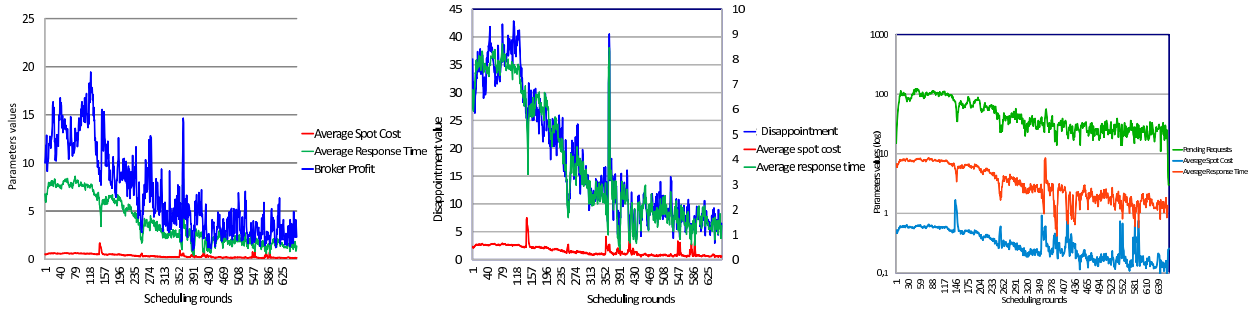

Figure 5. The relationship between the average instances (spots) cost, their average response time: **(Left)**: and the broker profit. **(Middle)**: and the real time client's satisfaction (disappointment) for a disabled broker profit option. **(Right)**: The relationship between the number of unfinished VM requests, the average instances (spots) response time and the average instances (spots) cost.

[5] G. Mankiw, *Principles of economics*. Sourth-Western Pub, 2008.

[6] Y. C. Lee, C. Wang, A. Y. Zomaya, and B. B. Zhou, "Profit-driven service request scheduling in clouds," in *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, May 2010, pp. 15 –24.

[7] J. Burge, P. Ranganathan, and J. Wiener, "Cost-aware scheduling for heterogeneous enterprise machines (cash em)," in *Cluster Computing, 2007 IEEE International Conference on*, 2007, pp. 481 –487.

[8] J. Yu and R. Buyya, "Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms," *Scientific Programming*, vol. 14, no. 3-4, pp. 217–230, 2006.

[9] S. Garg, P. Konugurthi, and R. Buyya, "A linear programming driven genetic algorithm for meta-scheduling on utility grids," in *Advanced Computing and Communications, 2008. ADCOM 2008. 16th International Conference on*, 2008, pp. 19 –26.

[10] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimal virtual machine placement across multiple cloud providers," in *Services Computing Conference, 2009. APSCC 2009. IEEE Asia-Pacific*, dec. 2009, pp. 103 –110.

[11] E. Elmroth, F. Marquez, D. Henriksson, and D. Ferrera, "Accounting and billing for federated cloud infrastructures," in *Grid and Cooperative Computing, 2009. GCC '09. Eighth International Conference on*, aug. 2009, pp. 268 –275.

[12] M. Andreolini, S. Casolari, M. Colajanni, and M. Messori, "Dynamic load management of virtual machines in cloud architectures," in *Cloud Computing*, ser. LNICST, D. Avresky, M. Diaz, A. Bode, B. Ciciani, and E. Dekel, Eds. Springer Berlin Heidelberg, 2010, vol. 34, pp. 201–214.

[13] J. Tordsson, R. S. Montero, R. Moreno-Vozmediano, and I. M. Llorente, "Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers," *Future Generation Computer Systems*, vol. 28, no. 2, pp. 358 – 367, 2012.

[14] J. L. Lucas-Simarro, R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, "Scheduling strategies for optimal service deployment across multiple clouds," *Future Generation Computer Systems*, no. 0, pp. –, 2012.

[15] J. Chen, C. Wang, B. B. Zhou, L. Sun, Y. C. Lee, and A. Y. Zomaya, "Tradeoffs between profit and customer satisfaction for service provisioning in the cloud," ser. HPDC '11. New York, NY, USA: ACM, 2011, pp. 229–238.

[16] (2012) Amazon elastic compute cloud (Amazon EC2). http://aws.amazon.com/fr/ec2/.

[17] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979.

[18] (2012) Amazon ec2 pricing. http://aws.amazon.com/fr/ec2/pricing/.