Multi-objective Genetic Algorithm for Multi-cloud Brokering

Alba Amato, Beniamino Di Martino, and Salvatore Venticinque

Department of Industrial and Information Engineering, Second University of Naples
{alba.amato,beniamino.dimartino,salvatore.venticinque}@unina2.it

Abstract. Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources offered by commercial providers according to specific service level agreements. Research effort has been spent to address the lack of Cloud interoperability that is a barrier to cloud-computing adoption because of the vendor lock-in problem. In fact the ability to easily move workloads and data from one cloud provider to another or between private and public clouds can improve performance, availability and reduce costs. In this paper we explore the potential use of multiobjective genetic algorithms in the field of a brokering service, whose aim is to acquire resources from multiple providers on the basis of SLA evaluation rules finding the most suitable composition of Cloud offers that satisfy users' requirements.

1 Introduction

Cloud computing, as defined by the National Institute of Standards and Technology (NIST) [10], is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. However users see the lack of cloud interoperability as a barrier to cloud-computing adoption because organizations fear vendor lock-in. In fact the ability to easily move workloads and data from one cloud provider to another or between private and public clouds is often hampered by several factors. Sometimes users are not able to find the most suitable configuration of resources for their applications and to compare it with the several offers of different providers. In fact service options are often not comparable owing to their heterogeneity in terms of resources, technology and service levels. In order to support users in service negotiation we have designed and developed Cloud Agency, a Multi-Agent System (MAS) that have the main task of selecting dynamically a set of Cloud resources, from different vendors, that best fits users requirements. Cloud agency service is compliant with the NIST definition of cloud broker [11], that is an entity that manages the use, performance and delivery of cloud services, and negotiates relationships between Cloud Providers and Cloud Consumers. In previous research [2] [3] we addressed the brokering problem choosing the best offer from a single provider

D. an Mey et al. (Eds.): Euro-Par 2013 Workshops, LNCS 8374, pp. 55–64, 2014.

performing experiments using a single Service Level Agreement (SLA) of a single provider, and the constraints defined on a single service. In this paper we aim at investigating the brokering problem applied to the composition of services from different providers that consists of brokering n services, in a set of m proposals from different providers, which can be composed according to a specific workflow. In this case complexity grows exponentially with n and it will be more difficult to treat because it is necessary to define global criteria among different SLAs and to consider multiple solutions of composition. Here we explore the potential use of multiobjective genetic algorithms in the broker module, whose aim is to acquire resources from providers on the basis of SLA evaluation rules finding the most suitable Cloud provider that satisfy users' requirements. The paper is organized as follows: in Section 2 some relevant works are presented, Section 3 describes the application context and a solution for the brokering problem. Section 4 briefly describes the proposed algorithm; conclusions are drawn in Section 5.

2 Related Work

The current role of multi-objective optimization in various sectors is becoming increasingly relevant. As with most problems, the objectives to be considered are many and often contradict each other. The classical methods of solution are based on the idea of transforming the original problem into one with a single objective function (scalarization) [9]. Although this technique is easy to implement it presents a series of disadvantages, first of all the fact to return a single optimal solution hypothesized. The first problem is constituted by the fact that it is not always clear where this solution is placed compared to the Pareto Front, and in some cases this solution may even prove to be a solution dominated. Besides knowing this variety of solutions on the Pareto Front is extremely useful to have a wider choice available. The Pareto-optimal formulation of the multi-objective problem, in which are not considered scalarization and where all objectives in contrast are kept separate is certainly a more valid and reliable method. Evolutionary algorithms [13] have become some of the main methods for the Exploration of Pareto optimal front in multi-objective optimization problems. This is due not only to the fact that there are few alternatives for finding multiple Pareto optimal solutions in large spaces that could be "intractable". But also to the fact that thanks to parallelization, to their capability to explore the similarity of the solutions with the operation of recombination, these algorithms are able to approximate the Pareto optimal front in a single optimization run. This has resulted in a use of multi-objective evolutionary algorithms in more and more numerous applications and a rapidly growing interest in this type of algorithms. Evolutionary Multi-objective Optimization Methods are based on the principle of natural selection and can be seen as a combination of the evolutionary computation and traditional multiple criteria decision making. A considerable number of evolutionary multi-objective optimization techniques have been developed in recent years [6], [5]. In particular in this paper we discuss the utilization of

NSGA-II (Non-dominated Sorting Genetic Algorithm II) that represents one of the most applied methods in the selection literature. NSGA-II [7] is a computationally fast and elitist MOEA (Multiobjective Evolutionary Algorithm) based on a non-dominated sorting approach that alleviates some difficulties of NSGA and is capable of outperforming many other genetic optimization algorithms. In the first step, NSGA-II constructs a space of solutions, then performs sorting based on non-domination level, and applies the crowded comparison operator to create a new pool of offspring. It applies a fast non-dominated sorting approach which has $O(MN^2)$ computational complexity, where M is the number of objectives and N the population size. NSGA [12] was proposed by Srinivas and Deb. The algorithm modifies the ranking procedure originally proposed by Goldberg [8] based on several layers of classifications of the individuals.

3 The Brokering Problem

The brokering problem consists of choosing the best proposal among the number of offers, which have been received from different providers, who answer to the same call. Here we define Call For Proposal (CFP) the document to be prepared by the customer to specifies his requirements. The CFP includes the list of resources to be acquired and the rules/policies to be used for defining resource brokering strategies, *i.e.* based on prices, availability, etc. As shown in Fig.1, the CFP is composed of two documents. The first one is the *SLA Template* that is described according to the XML SLA@SOI schema (Available at: http://sla-at-soi.eu/). The second document composing the CFP is the *Broker Policy*, containing a set of rules, to be enforced by the brokering algorithm, in order to choose among the different proposals offered by the Cloud market. The *SLA Template* expresses the configuration of resources that are necessary for the user and consists of a set of virtual resources that may include compute, network and storage. In particular the SLA template [1] is composed of *Service Properties*, that define the technical requirements for user's applications; and

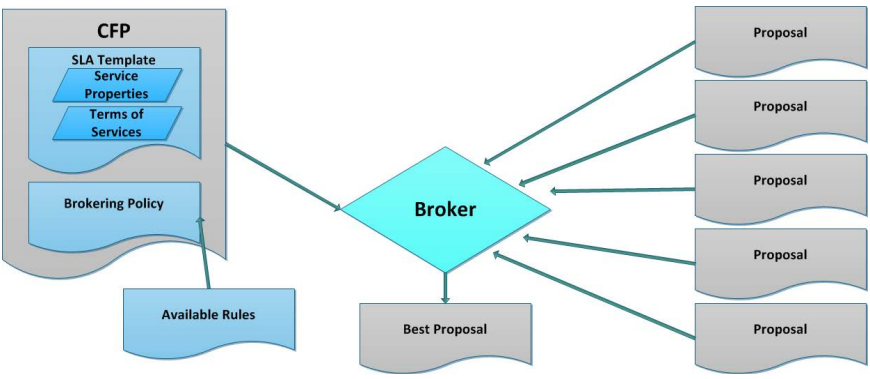

Fig. 1. Architecture of the mOSAIC Broker

the correspondent desired *Service Levels*, such as availability, reliability, performance; *Terms of Service* that include the contract duration, data location and billing frequency, etc. Brokering policy sets *constraints* and *objectives* on multiple parameters such as the best price, the greatest number of cores, the best accredited provider or the minimum accepted availability. We propose a model to formulate the application requirements into constraints that can be architectural constraints and service level constraints. Besides this constraints can be set *hard* or *soft*. Hard constraints refer to the fact that the Cloud offer must have the required condition, otherwise it is to be excluded. Soft constraints refer to desired requirements that can make a provider preferred with respect to another. Simple constraint rules are: exact matches, value in a set, greater/less then, value in a range. Of course not every constraint can be applied to any SLA parameters. For example cpu architecture should match a certain type (x86, amd64) or could match any values in a set. Billing frequency can be set equal to *per_month*, *per_year* or *any*. Price can be set less or equal to a certain value. Availability and Reputation can be set greater or equal to a certain value. The same parameters of the SLA can be considered as input of objectives functions to be optimized. There may be no one single objective for the optimization, and no single optimal solution. For example conflicting goals could be minimization of the cost and maximization of memory. In this case, a multi-objective approach should be adopted and one of the solutions on the Pareto front (that is a set of all those solutions that are considered to be optimal in multi-criteria optimization) should be chosen.

3.1 One-Cloud Evaluation and Brokering

Our broker [3] finds in a set of SLA proposals $\{P_1, \ldots, P_m\}$ the ones that optimize a multi objectives function and satisfy a number of constraints. Each proposal is provided by a Cloud vendor complementing the terms of the SLA template $T = \{t_1, \ldots, t_n\}$, received from the customer, with the correspondent offered values $Pj = \{(t_1; v_{j,1}), \ldots, (t_n; v_{j,n})\}$. In order to evaluate the best proposal the broker can use a set of rules $R = C \cup O$, which can be constraints rules $cr \in C$ or objectives rules $or \in O$ rules. Constraints rules are boolean expression that can represent soft or hard requirements:

$$cr : (t_i, c_i, m_i) \rightarrow [true, false] \tag{1}$$

t_i is an SLA term, c_i is a boolean expression, and m_i specifies that the constraints is hard (when 1) or soft (0). Objectives rules assign a score between 0 and 1 to the compliance of the SLA value of the term t_i with the correspondent user's requirements

$$or : (t_i, f_i, o_i) \rightarrow [0, 1] \tag{2}$$

For each objective rule the user has to select a mapping function f_i between the t_i values and the correspondent score, and has to specify if that rule is an explicit ($o_i = true$) or implicit objective ($o_i = false$). The mapping function change the way to evaluate that objective. For example logarithmic, linear, and exponential

function can be used to define the relevance of that objective according to the value $v_{j,i}$ offered by the provider and the one desired by the customer. The broker policy will be a subset of rules $R' \subset R$ that is defined for those terms of the SLA template, which are relevant for the user's requirements. To solve the brokering problem we have to perform the following computation for each received proposal by replacing t_i with the correspondent value $v_{j,i}$:

- $M_j = \prod_{i=1}^{n} \neg(\neg c_i \wedge m_i) \, \forall j = 1, \ldots, m$
  that is used to check if the SLA proposal can be considered as a valid candidate for the SLA , in fact at least a false mandatory constraint invalidates that offer.
- $Opt_j = \sum_{i=1}^{n} (\neg m_i * (c_i = true)) \, \forall j = 1, \ldots, m$
  evaluates how many soft constraints are met. It can contribute to the evaluation of the proposal.
- $V_j = \sum_{i=1}^{n} ((o_i = false) * f_i(v_{j,i})) \, \forall j = 1, \ldots, m$
  represents an overall evaluation for all those terms which have not to be negotiated independently.

All objectives rules which have $o_i = true$ will be considered independent objectives. In general the best proposals will be the ones which solve the following equations:

$$max_{j=0}^{m}(or_i) : o_i = true \; and \; M_j = 1 \, \forall i = 1, n \tag{3}$$

An additional criteria will be:

$$max_{j=0}^{m}(Opt_j) : M_j = 1 \tag{4}$$

All the defined criteria can be grouped if the user set $o_i = false \forall i = 1, \ldots, n$, and $m_i = false \, \forall i = 1, \ldots, n$. In this case the result of brokering will be:

$$max_{j=0}^{m}(V_j) : M_j = 1 \tag{5}$$

that means the best proposal are the ones with the best overall score. The policy edited will be translated in a language supported by programs for symbolic computation like Octave, Matlab or Maple and will be run by replacing t_i parameter with actual values of each proposal.

3.2 Multi-cloud Brokering

In the case of Multi-Cloud brokering problem it needs to define global criteria among different SLAs. Brokering rules defined in the Section 3.1 have to be extended to the entire composition in order to check that the composition still fulfill the consumer's requirements. Moreover, in this case, there is the need to treat the computational complexity that would occur if it were decided to explore all the solutions. In fact several solutions consisting of different combinations of service composition should be evaluated. For this reasons, it is necessary to use algorithms aimed at finding the optimal solution which have a sustainable computational complexity. Some QoS parameters (such as cost, response

time, the availability, etc.) can be composed in accordance with a predetermined function, while others may not have a linear dependence on the QoS of each single component. In this preliminary work, given the user service composition, the requirements and their QoS constraint descriptions, we assume that it is possible to express the QoS of n services that make up a composite service as $QoS(composite) = f(QoS(s_1), ..., QoS(s_n))$ where QoS (s) represents the value of the QoS level for a service s, composite is the composite service and s_1, s_2, .. s_n are the services that make up the composite service. As an example, let the negotiation target be a cloud virtual machine service and the parameters of SLA are: Hard Disk, CPU, RAM, Cost, Availability. The approach for computing a the QoS of a composite service is those proposed by [4] but differently from [4] we consider both the indexes of availability promised in the SLA, in the raw **Availability (A)**, and the benchmarking results by a third party (e.g Cloud Harmony available at: www.cloudharmony.com) in the raw **Availability (A_b)**. In the case that different providers will be used as several replicas to increase availability or reliability the aggregation function will be the same of a switch construct shown in Table 1. In the same way if we have a multi-tier Cloud application, that uses services from different providers to optimize the cost or performances the related aggregation function will be equivalent to a sequence. Table 1 shows an aggregation function for each pair work flow construct and QoS attribute.

Table 1. Aggregation functions

QoS Attr.	Sequence	Switch	Flow	Loop
Time (T)	$\sum_{i=1}^{m} T(t_i)$	$\sum_{i=1}^{n} p_{ai} * T(t_i)$	$Max(T(t_i)_{i\in(1...p)})$	k*T(t)
Cost (C)	$\sum_{i=1}^{m} C(t_i)$	$\sum_{i=1}^{n} p_{ai} * C(t_i)$	$\sum_{i=1}^{p} C(t_i)$	$k * C(t)$
Availability (A)	$\prod_{i=1}^{m} A(t_i)$	$\sum_{i=1}^{n} p_{ai} * A(t_i)$	$\prod_{i=1}^{p} A(t_i)$	$A(t)^k$
Availability (A_b)	$\prod_{i=1}^{m} A_b(t_i)$	$\sum_{i=1}^{n} p_{ai} * A_b(t_i)$	$\prod_{i=1}^{p} A_b(t_i)$	$A_b(t)^k$
Reliability (R)	$\prod_{i=1}^{m} R(t_i)$	$\sum_{i=1}^{n} p_{ai} * R(t_i)$	$\prod_{i=1}^{p} R(t_i)$	$R(t)^k$

4 Multiobjective Optimization with NSGA-II

In general, a genetic algorithm allows to find a set of optimal solutions for the problem that is represented by an individual of the space of possible solutions. To operate, the algorithm needs to represent in some way the individuals so to each individual is assigned an appropriate string that represents it, said chromosome: a chromosome is formed by multiple components such genes each of which represents a single feature of the solution. The number of components present in each string is equal to the dimension of the space of solutions. In a multi-objective genetic algorithm, pursuing multiple goals, an individual is considered to be more or less appropriate in relation to most criteria. Besides, as seen before, also the result will be a set of optimal solutions, so the genetics algorithms are used to explore the Pareto front. To determine whether a solution is more suitable than another, a mechanism ranking is needed, different from the value

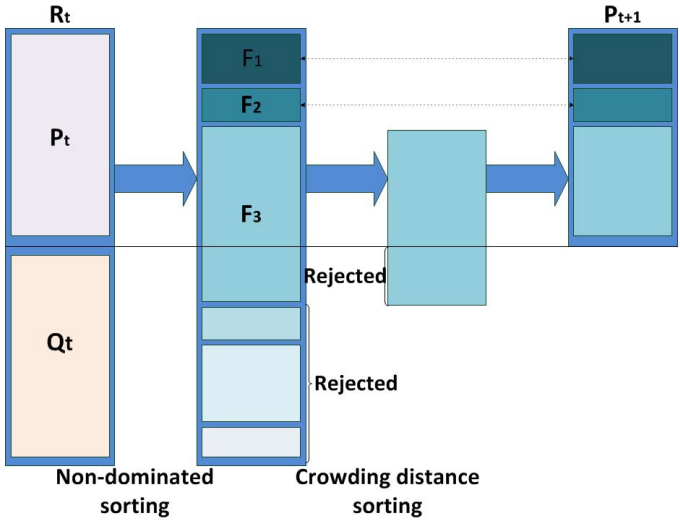

Fig. 2. Structure of NSGA II algorithm

of its own fitness, that have to take into account the presence of more goals. The ranking may be based on dominance, or it can be based on the preference of an objective respect to another, for example by assigning appropriate weights to each objective. In particular NSGA II is based on strict rules of classification of individuals. Before the selection, the population is classified according to the concept of dominance: all individuals that are not dominated, are classified into groups or classes, called rank. In order to maintain the diversity of solutions, individuals classified are shared based on the value that the fitness functions assume in them, and, once formed the first group, it is momentarily ignored for proceed to the creation of other groups of non-dominated lower order. The process continues until all individuals are classified. A key feature of NSGA II is to preserve the diversity among the solutions of the same non-dominated front, in order to maintain a good distribution of solutions. To do this, the NSGA II uses the crowding-distance, a measure of the distance of an individual from its "neighbors", and is used by the algorithm to ensure adequate distribution of individuals, in order to lead the population to adequately explore the whole space of objectives. At first iteration, the usual binary tournament selection, recombination, and mutation operators are used to create a offspring population Q_0 of size N. This population is then combined with the first random generated population P_0 of size N. As shown in Fig. 2 [7], for each iteration a combined population R_t is formed as union of current population Q_t and the previous population P_t. Then, the population R_t, of size $2N$, is sorted according to non domination. After that individuals with the lowest rank are chosen and, in case of equal rank, with a high value of crowding-distance so forming the population P_{t+1}. The new population of size N is then used for selection, crossover,

and mutation to create a new population P_{t+1}. The iteration proceeds until the number of generations required has been reached. In our approach we list the SLA templates offered by those providers who answered to a call for proposal for a Cloud service. Each proposal (a gene of individual) is referenced by its index in the list and is a candidate to be a component of the final SLA. An individual corresponds to an array of integers whose length is equal to the number of services of the composition. For each individual values of constraints and objectives are computed by the model defined in Section 3.1 on each proposal. All values, if the rules has been specified to be global, are aggregated according to the function defined in Section 3.2. NSGA II pseudo code of multi-cloud brokering is described in Algorithm 1. In order to evaluate performance and

Algorithm 1. NSGA II for Brokering

Data N individuals, g generation, $f_i(x_i)$
Result N results

```
Initialization of Population P of size N
Evaluation of Constraints
Evaluation of Objectives
Ranking Based on Pareto dominance
Generation of Child generation
Tournament Selection
Recombination and Mutation
for i = 1 → g do
    for each Parent and Child in Population do
        Ranking Based on Pareto
        Dominance Crowing Distance
    end for
    Creation of next generation
    Tournament Selection
    Recombination and Mutation
end for
```

feasibility of the proposed approach we used Jmetal, an API that provides an implementation of many genetic algorithms. We developed a multi-cloud broker the actually support the evaluation of general reduction functions on any parameters of a cloud proposal. It supports mean, maximum, minimum value, sum, product reduction operations, but also a number of workflows, which allow to combine services by a switch or sequence operator. In order to test the brokering algorithm with realistic cloud proposals we evaluated 41 different VM offers from Amazon, Rackspace and IBM. For each offer we considered amount of memory, cpu speed, number of cores, availability cost and others. We imagine that a cloud customer want minimize costs and maximize the number of cores of his system by spreading the load of n different machines, possibly allocated in different geographical areas to better support customers in different parts of the world based to geographic jurisdictions. To do this he needs a load balancer to address the tasks to different VM, and a function of ip to redirect the client

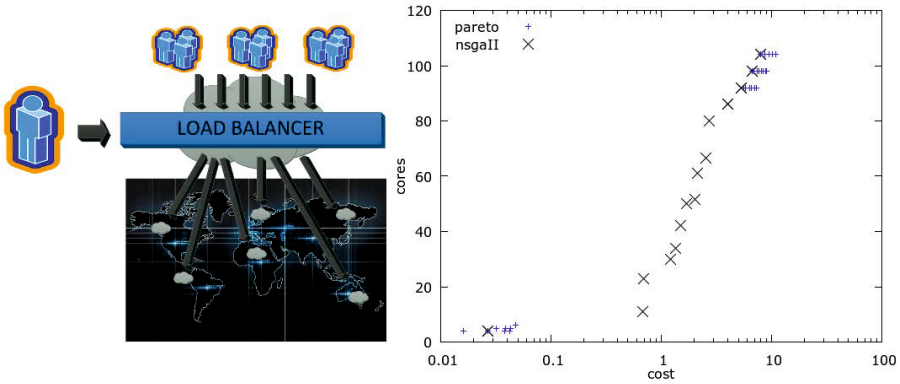

Fig. 3. Example of NSGA II utilization for multiple broker

to the server the corresponding geographical area. So while some tasks may be executed on the same cloud, some others may run on different cloud platforms. The allocation of workloads is used to improve resource utilization and reduce costs. It means that the cloud customer would improve the performance of its composed service. These are really conflicting objectives. Figure 3 shows that in the true Pareto front there are either cheap solutions with few cores or expensive solutions with many cores. We observe a great diversity between solutions that optimize different objectives. In this case the total cost will be computed as: $C_T = \sum_{i=1}^{n} C(t_i)$ and the totalnumber of cores as $CORE_T = \sum_{i=1}^{n} CORE(t_i)$.

5 Conclusion

In this paper we proposed the use of multiobjective genetic algorithms in the field of a Multi-Cloud brokering service, to allow for the evaluation of a composite service by different providers. In particular we proposed the utilization of NSGA II to consider multiple SLAs simultaneously whose composition satisfy the user's requirements. A preliminary example of NSGA II utilization for multiple broker has been shown. The evaluation of the Multi-Cloud brokering service implementation and the discussion of performance results are matter of future work.

Acknowledgments. This work has been supported by PRIST 2009, *Fruizione assistita e context aware di siti archelogici complessi mediante terminali mobile*, founded by Second University of Naples and by the mOSAIC project (EU FP7-ICT programme, project under grant #256910).

References

1. Amato, A., Liccardo, L., Rak, M., Venticinque, S.: Sla negotiation and brokering for sky computing. In: CLOSER, pp. 611–620 (2012)
2. Amato, A., Di Martino, B., Venticinque, S.: Evaluation and brokering of service level agreements for negotiation of cloud infrastructures. In: ICITST, pp. 144–149 (2012)
3. Amato, A., Venticinque, S.: Multi-objective decision support for brokering of cloud sla. In: The 27th IEEE International Conference on Advanced Information Networking and Applications (AINA 2013), March 25-28. IEEE Computer Society, Barcelona (2013)
4. Canfora, G., Di Penta, M., Esposito, R., Villani, M.L.: An approach for qos-aware service composition based on genetic algorithms. In: Proceedings of GECCO 2005, pp. 1069–1075. ACM (2005)
5. Carlos, A.: Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. Knowledge and Information Systems 1(3), 129–156 (1999)
6. Dastjerdi, A.V., Buyya, R.: A taxonomy of qos management and service selection methodologies for cloud computing. In: Cloud Computing: Methodology, Systems, and Applications, pp. 109–131. CRC Press (2011)
7. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
8. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning, 1st edn. Addison-Wesley Longman Publishing Co., Inc., Boston (1989)
9. Marler, R.T., Arora, J.S.: Survey of multi-objective optimization methods for engineering. Structural and Multidisciplinary Optimization 26(6), 369–395 (2004)
10. Mell, P., Grance, T.: The nist definition of cloud computing. Tech. rep., National Institute of Standards and Technology (2011)
11. NIST: NIST cloud computing reference architecture - special publication 500-292 (2011), `http://www.nist.gov/`
12. Srinivas, N., Deb, K.: Multiobjective optimization using nondominated sorting in genetic algorithms. Evol. Comput. 2(3), 221–248 (1994)
13. Van Veldhuizen, D.A.: Multiobjective evolutionary algorithms: classifications, analyses, and new innovations. Ph.D. thesis, Wright Patterson AFB, OH, USA, aAI9928483 (1999)