Generic Cloud Platform Multi-objective Optimization Leveraging Models@run.time

Donia El Kateb
Security, Reliability and Trust Interdisciplinary Research Center, SnT & Laboratory of Advanced Software SYstems (LASSY)
University of Luxembourg
Luxembourg
donia.elkateb@uni.lu

François Fouquet
Security, Reliability and Trust Interdisciplinary Research Center, SnT
University of Luxembourg
Luxembourg
francois.fouquet@uni.lu

Grégory Nain
Security, Reliability and Trust Interdisciplinary Research Center, SnT
University of Luxembourg
Luxembourg
gregory.nain@uni.lu

Jorge Augusto Meira
Security, Reliability and Trust Interdisciplinary Research Center, SnT
University of Luxembourg
Luxembourg
Jorge.Meira@uni.lu

Michel Ackerman
EBRC
Luxembourg
michel.ackerman@ebrc.com

Yves Le Traon
Security, Reliability and Trust Interdisciplinary Research Center, SnT & Laboratory of Advanced Software SYstems (LASSY)
University of Luxembourg
Luxembourg
yves.letraon@uni.lu

ABSTRACT

Cloud computing promises scalable hosting by offering an elastic management of virtual machines which run on top of hardware data centers. This elastic management as a cornerstone of PaaS (Platform As A Service) has to deal with trade-offs between conflicting requirements such as cost and quality of service. Solving such trade-offs is a challenging problem. Indeed, most of PaaS providers consider only one optimization axis or ad-hoc multi-objective resolution techniques using domain specific heuristics.

This paper aims at proposing a generic approach to build cloud optimization by combining modeling and search based paradigms. Our approach is two-fold: 1) To reason about a cloud environment, we use a Models@run.time approach to have an abstraction layer of a cloud configuration that supports monitoring capabilities and represents cloud intrinsic parameters like cost, load information, etc.. 2) We use a search-based algorithm to navigate through cloud candidate configuration solutions in order to solve the *C*loud *M*ulti-objective *O*ptimization *P*roblem *(CMOP)*.

We validate our approach based on a case study that we define with our cloud provider partner EBRC [1] as representative of a dynamic management problem of heterogeneous distributed cloud nodes. We implement a prototype of our PaaS supervision framework using Kevoree, a Models@run.time platform. The prototype shows the efficiency of our approach in terms of finding possible cloud configurations in reasonable time. The prototype is flexible since it enables an easy reconfiguration of the cloud customer optimization objectives.

Keywords

Models@run.time, Multi-Objective Optimization, SaaS, Search based Algorithms

1. INTRODUCTION

A cloud infrastructure is a set of physical machines which are clustered in geographically distributed data centers. Virtual machines (VMs) can be started or stopped on those physical machines to host software that is delivered as a service. VM management is an orthogonality constrained problem: Cloud customers aim at minimizing cost by consolidating software elements on VMs and reducing active VMs, at the same time, to achieve software isolation or fault tolerance they tend to replicate software on different VMs and thus have to increase the number of their active VMs. This local optimization that can be provided by a cloud provider to a cloud customer is typically one of the major components of a platform as a service (PaaS) hosting. Current cloud platforms usually focus on one dimensional optimization. For instance, RedHat OpenShift [2] PaaS provides a rule-based engine to deal with horizontal computational power and disk space scalability by starting and stopping gears (computation nodes). In this engine, rules are uncorrelated and represent independent mono-objective optimization axis. In addition, such rule-based engine is not suitable to handle non deterministic choices such as selecting one action in case of several available gears types that can achieve horizontal scalability.

[1] http://www.ebrc.com/

[2] https://www.openshift.com/wiki/architecture-overview#_3._Horizontal_scaling_Beta

SAC'14 March 24-28, 2014, Gyeongju, Korea

This paper tackles cloud multidimensional and orthogonal optimization. Our goal is to provide a supervision approach supported by a decision making tool that can be reconfigured according to evolving customer optimization objectives. In the presence of conflicting objectives, our approach aims at calculating an optimal cloud configuration, given a set of resources that are dedicated to a cloud customer.

In a nutshell, our approach focuses on generating a customer optimized PaaS layer. We model a cloud supervised infrastructure using Kevoree, a models@run.time [24] platform that simulates possible space of cloud reconfigurations and adaptations. This enables to reason at the abstract level, on a dynamically updated model of a cloud configuration, and then propagate the configuration changes at the different technical layers. We model the cloud management problem as a multi-objective search problem where customers requirements are captured through SLA (Service Level Agreements) [25]. We use genetic algorithms applied on a given system configuration snapshot, to resolve the *Cloud* Infrastructure Management *Multi-objective Optimization Problem (CMOP)*, by dynamically choosing optimal configurations among the space of possible cloud configurations alternatives. We validate our approach through a use case defined with our partner EBRC.

The use case relies on heterogeneous cloud infrastructure (different categories of virtual machines). In this use case, we address two research questions. 1) The first research question aims at showing the ability to reconfigure a cloud infrastructure at runtime in the presence of conflicting objectives. In this research question, we have explored the feasibility of our optimization approach at the level of the architectural model that we have built using the models@Runtime platform Kevoree. 2) The second research question is related to the performance of our approach with regards to the customer infrastructure scalability.

This paper is organized as follows. Section 2 describes the key concepts related to this paper. Section 3 describes the approach that we have used to solve the multi-objective problem. Section 4 details the experiments that we have run to validate our approach. Section 5 discusses related work. Finally, Section 6 presents our conclusion and future work.

2. PROBLEM DEFINITION AND MODELING

This section defines the *Cloud* Infrastructure Management *Multi-objective Optimization Problem (CMOP)* and motivates the use of Models@run.time for cloud abstraction.

2.1 CMOP Description

A cloud infrastructure environment can be strongly abstracted as a set of Physical Machines (PMs) hosting Virtual Machines (VMs). Cloud customers benefit from virtualization technology to host their data/applications according to Service Level Agreements (SLA) [27] which provide the key terms of a contract to regulate a service between a cloud customer and a cloud provider. Cloud environments are dynamic environments in which customers and providers are expressing evolving requirements towards cost saving, QoS evolution, etc... PaaS providers mostly manage VM allocation (i.e. by starting VMs or shutting them down) to regulate the quality of service of customers services.

In this work, we explore optimization from cloud customers perspective. A cloud customer has to adapt to evolving needs that might be conflicting. Given a cloud customer infrastructure which consists in a dedicated set of allocated physical resources, we aim at developing a decision support that takes as input SLA objectives and a snapshot of a cloud configuration and provides decision capabilities to place software components in VMs in the objective to maintain compliance with SLA requirements and to adjust costs to specific customers needs. We believe that having a business model that handles in a fine-grained way customers deployment costs in an elastic cloud is one of the main push factors towards cloud technology. The fine-grained customer cloud management that we propose in this paper, allows cloud customers to reconfigure their optimization objectives and to adjust their resources provisioning to deployment costs.

2.2 CMOP Inputs

VM placement is an active research area [10] that illustrates the trade-offs that should be taken into consideration to manage a cloud infrastructure. As an illustrative example, to reduce costs, customers have to consolidate (VMs) that have to be started on the physical machines. However at the same time, they have to host the loads of different end users on different VMs to achieve isolation.

VMs heterogeneity in terms of deployment cots or in terms of hosting geographical location adds more complexity to VM management problem and motivates the need to have well-defined reasoning techniques for VM allocation.

In this paper, we consider scenarios that present an actual interest to EBRC, a cloud provider in Luxembourg, which offers a complete range of tailored services in hosting and managing a cloud infrastructure. EBRC is interested in improving the supervision of its customers environments. In this paper, we explore resolution strategies to deploy an optimized cloud environment for EBRC customers. Customers expectations are commonly captured through SLA (Service Level Agreements) which consist in defined terms in the form of a contract between a service consumer and a service provider. As defined in the SLA Handbook [4]: "It is the Service Level Agreement that defines the availability, reliability and performance quality of delivered telecommunication services and networks".

Most SLA introduced in the literature have focused on the aspects related to performance and have not investigated the aspects related to security. In this work, we consider the isolation property in the context of a single customer infrastructure. Depending on the confidentiality of their data, we consider that customers are able to express some constraints related to the isolation of their workloads/data in the cloud environment. For example, a customer may require that his applications/data are hosted in dedicated VMs. In this work, we also consider cost reduction as an optimization objective. Next section shows how we model the different customer objectives through multi-objective fitness functions.

2.3 Models@run.time for Cloud Abstraction

As shown in Figure 1, PaaS customers use cloud infrastructure resources to host applications and data that can be accessed by end users. (1) A need for customers local cloud optimization is triggered by some application increasing workload, a change in security requirements or to new energy reduction objectives. (2) To achieve local optimization, the cloud customer recurs to our framework to trigger a VM reconfiguration at the level of his own infrastructure. The request is interpreted at the PaaS level so that VM placement can be taken into account at the level of physical machines. Our framework aims at reconfiguring a cloud infrastructure, at a given time, to find an optimal configuration given the presence of several conflicting constraints. The framework is based on a genetic algorithm to solve *CMOP* that has been simulated in a Models@run.time [5] platform. We recur to Models@run.time tech-

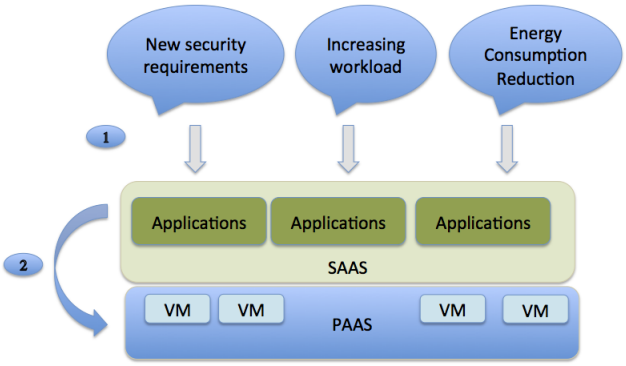

Figure 1: Positioning with regards to cloud Service Models

niques since they offer an abstraction layer to reason about cloud adaptations and about the different trade-offs that have to be envisioned at runtime. Models@run.time paradigm is an evolution of Model-driven engineering (MDE) [19] that permits to reason about the system at design time and extends the reasoning to the system while it is in a running state. It takes into consideration design time information at runtime for continuous reevaluation, to reevaluate requirements satisfaction while the system is evolving. Cloud infrastructure is thus a typical environment that can take advantage of Models@run.time techniques since they can provide a global view that reduces cloud infrastructure management complexity. Models@run.time provide high level abstraction for Clouds through a standard MOF-based metamodel [28]. Our approach bridges the gap between any MOF-based tool and real infrastructures. The framework is thus able to provide a supervision layer for a customer infrastructure by identifying optimal cloud local configurations in some evolving contexts like an increasing workload. We use Kevoree as a platform to run our implementation. A physical machine is abstracted as a Kevoree Infrastructure Node, a VM is abstracted as a Kevoree Child Node and a component is abstracted as a software service. The architectural model can be easily deployed in a real large-scale production environment [12] [11] like a cloud infrastructure.

3. PROBLEM RESOLUTION

A supervision problem formulated by SaaS cloud providers can be stated as follows: "Given a dedicated set of virtual machines allocated by a PaaS provider, how to optimize software placement in VMs to reduce costs and to maintain compliance with SLA requirements". We have opted for a search-based approach to study the effectiveness of the supervision framework. Search Based Software Engineering (SBSE) research [15] has been widely explored in the last decades in several problems in software engineering and particularly in multi-objective optimization problems. The applicability of search based approaches in the domain of cloud engineering has been motivated in [16]. This usage has been supported by techniques such as meta-heuristics and evolutionary algorithms [7]. Genetic Algorithms [14] belong to search-based evolutionary approaches that are commonly applied for multi-objective optimization problem resolution. Genetic algorithms maintain a pool of solutions, called a population, which evolve from one generation to another. The selection of best individuals that serve for producing a new generation is based on a fitness function evaluation that evaluates solutions to determine the ones that should be selected in the next generation of solutions to better achieve an objective. In each generation, the genetic algorithm constructs a new population using genetic operators. The population is normally randomly initialized. As the search evolves, the population includes fitter and fitter solutions, and eventually it converges. Genetic algorithms have proven to be efficient in several areas of software engineering such as software testing problems [3] [1] [2].

3.1 CMOP Modeling as a Search-based Problem

In this paper, we use genetic algorithms to resolve cloud supervision problem. Formally, we define *CMOP* by the following triplet *(I,F,CO)* such that:

1. I denotes a cloud infrastructure model. A cloud infrastructure model I is an abstraction of a set of VM. Each VM hosts n software components C [17].
2. F is a vector of objective functions $F(X) = (f_1(x), f_2(x), ..., f_m(X))$, that have to be minimized.
3. CO denotes a set of possible configurations in I that satisfy F. A configuration $co \in CO$ is obtained through a mapping function that maps the components (C) to (VMs). A configuration co is represented like the following: co=($VM_1(c_1...c_k)$,...,$VM_m(c_1...c_l)$), for example $co = VM_1(c_1, c_2, c_3)$ denotes a configuration with a single virtual machine VM_1 hosting 3 components c_1, c_2, c_3.

We model the key elements in our genetic algorithm as follows:

- **Individual:** A solution vector $x \in X$ that corresponds to a cloud infrastructure model.
- **Genes:** A gene corresponds to a component in our context.
- **Population:** A population corresponds to a set of cloud infrastructure models.
- **Genetic operator:** Two types of genetic operators are used in genetic algorithms to maintain genetic diversity: (1) The mutation operator operates on one individual (unary operator), (2) The crossover operator combines two existing solutions into new ones (binary operation on two solutions that results in new solutions). In this paper we do not consider crossover and the evolution is based on mutation operators only.
- **Fitness Function:** The fitness function is required to assign a value to each individual that reflects its potential ability to contribute to the problem solution.

3.2 Fitness Functions Definition

The vector F(X) is composed of the following 5 objective functions:

1. $f_1(x)$= Cost(x) where Cost(x) is calculated based on Amazon pricing model[3].
2. $f_2(x)$=Security(x) where x = {(C)}, security(x) defines the security level (classification or clearance) associated with a component [20]. If a component with a security level 4 shares a VM with a component with a security level 2, then a security violation of degree 2 is reported, if the component shares a VM with a component with a security level 1, then a security violation of degree 3 is reported.

[3]http://aws.amazon.com/ec2/pricing/

3. $f_3(x)$=Completeness(x) where x=(C_1,....,C_n) denote the software components that have to placed in a cloud infrastructure. The output value of $f_3(x)$ is 100 if no component is placed and 0 if all C_1,....,C_n are placed.

4. $f_4(x)$=Overload(x) where x = {(C)}. Overload(C) defines the gap between the required and the observed % of virtual CPU for each software element.

5. $f_5(x)$=SLAPerformance(x) where x={(C)} defines the required CPU for every component as stated in the SLA.

3.3 CMOP Algorithm

Table 1 presents our set of mutation operators O.

CMOP is based on NSGA-II which is a non dominated genetic algorithm that is based on the concept of dominance-based selection [9]. At each generation, the algorithm identifies non dominating solutions. The algorithm is characterized by a crowding distance that calculates a distance measure for an individual and its neighbors. This distance is used as a diversity measure within populations. Solutions selection is thus preformed by fitness values evaluation and by the crowding distance. In the validation section, we compare two variants of *CMOP* to assess their effects on the algorithm convergence. The two *CMOP* variants differ in their fitness function:

1. An ε-dominance (CMOP-epsilon Dominance) [21] that evaluates populations based on the values of fitness functions and the crowding distance [18].

2. A composite fitness that is the average of objectives scores: The mean value of $f_1(x)$ to $f_5(x)$ scores (CMOP-composite) [26].

Algorithm 1 describes the CMOP-epsilon Dominance.

Algorithm 1 <CMOP Algorithm>

Input: (Population-Size int, Generation-Number int)
Generate an initial population P
Evaluate $f_i(x)$ on all cloud configuration $co \in CO$ {//Individuals ranking based on the objectives functions evaluation}
Order $co \in CO$ according to $f_i(x)$ scores
Create an Archive A {//A contains ε-non dominated solutions $co \in CO$ of P }
while Stopping Criterion is not reached **do**
  Select solutions from A and P
  Apply randomly a set of mutation operators O on P to get P_{new}
  Evaluate P_{new} against P and A to select solutions $co \in CO$
end while

4. VALIDATION

To show the effectiveness of our approach, we have implemented CMOP (with and without epsilon dominance pareto front selection) on Kevoree platform and we have compared it to random and mono-objective solutions. The validation of our approach is related to the scenarios of interest for EBRC provider and focuses on the following research questions:

- RQ_1**:** Comparison of mono-objective vs. multi-objective optimization. *What is the benefit of using multi-objective optimization, such a CMOP-epsilon, compared to mono-objective one? How faster a good solution is found using a mono-objective algorithm compared to a multi-objective algorithm?*

- RQ_2**:** Comparison of multi-objective algorithms in terms of objectives satisfaction and scalability. Which algorithm, among CMOP-epsilon Dominance, CMOP without Epsilon Dominance and Random, better satisfies the objectives while scaling to more complex cloud infrastructure model?

In the validation, our goal is to show the effectiveness of a search-based approach in finding possible solutions at the level of the architectural model and to show that such approach scales. Comparison with other multi-objective optimization approaches is beyond the scope of this paper. In our scenario, EBRC allocates to a customer X a cluster composed of 5 physical machines to host its on-line stock trading website that is composed of the following components: A database to store items, a load balancer, a database for payment, a database to manage users and a web Front-end. Virtual CPU assignment depends on the physical machines which are either Intel Xeon or ARM processors based. In this scenario, we explore the supervision problem by considering optimization at the level of a single customer infrastructure.

4.1 What is the benefit of using multi-objective optimization compared to mono-objective one? (RQ_1)

Nowadays, most cloud providers use mono-objective optimization for horizontal scaling. For instance RedHat OpenShift [4] only considers the performance of a customer front-end, by an adaptation algorithm defined in the load balancer. In the same manner as RedHat OpenShift, Amazon [5] defines trigger levels based on performance measures to start or stop virtual machines. Both approaches only consider one objective (i.e. the front-end performance) to achieve optimization at the level of cloud customer infrastructure and do not take into account other optimizing factors like isolation. The first experiment that we have conducted tries to evaluate the potential quality gain (e.g. optimization quality) by considering all fitness functions during the optimization.

4.1.1 Setup

Here is the description of the cloud infrastructure model corresponding to the 5 physical machines that we deploy in Kevoree:

- 3 Low power consumption ARM based Infrastructure Node (1 Virtual Machine abstracted as child node where each node is 1GHz)

- 2 High power consumption Xeon based Infrastructure Node (8 Virtual Machine abstracted as child node where each node is 1GHz)

Table 2 defines some parameters that are relevant for the estimation of the fitness function:

- The required CPU (GHz) for every component

- The required security level for each component

- The CPU Load property which defines the required virtual CPU percentage for each component to run

Two algorithms for 10 initial populations are thus compared on this basic configuration:

- A mono-objective algorithm that minimizes SLAPerformance fitness without taking into consideration other fitness functions.

[4]https://www.openshift.com/wiki/architecture-overview#_3._Horizontal_scaling_Beta
[5]http://aws.amazon.com/autoscaling/

Operators	Description
AddComponent(c,A)	Adds a component c in the the virtual machine A
RemoveComponent(c,A)	Removes a component c in the virtual machine A
MoveComponent(c,A,B)	Moves a component c from the virtual machine A to the virtual machine B

Table 1: Mutation Operators

- A multi-objective CMOP-epsilon algorithm.

We have thus performed 20000 generations per run before selecting the best configuration and have used the techniques presented in Kevoree to increase the performance of our search based algorithm especially for the different operations defined in the previous section.

4.1.2 Results

For configurations found as solutions of the search-based problem, the value of the fitness vector for (Completeness, Consumption, Overload, Security, SLAPerformance) is illustrated in Table 3 and the evolution of the fitness scores of all objective functions given by CMOP are given in Figure 2. As already demonstrated by Frey *et al* [13], the multi-objective algorithm maximizes the satisfaction of the fitness functions through reaching a mean that is better compared to a mono-objective algorithm. However, it is interesting to notice that the 4 fitness functions reach a perfect score with the multi-objective search while only the SLAPerformance reaches a perfect score for the mono-objective algorithm. The consumption fitness in both cases could not reach 0 because at least one machine must be started to host a software component, however we observe a significant consumption reduction while using the multi-objective algorithm. The mono-objective search (SLAPerformance objective in our example) achieves the best score in 600 ms, CMOP-epsilon stabilizes a nearly optimal solution after 1700 ms and introduces then a time overhead compared to the mono-objective search. However the time overhead remains reasonable when running CMOP-epsilon, and presents a time duration value that is acceptable for a cloud infrastructure reconfiguration. This time overhead will be investigated in the next section.

4.2 Comparison of multi-objective algorithms (objectives satisfaction and scalability) (RQ_2)

Several algorithmic techniques are able to explore search based problems and to resolve the multi-objective trade-offs. Firstly, our algorithm can leverage epsilon-dominance for pareto-front selection and can operate without the epsilon-dominance as well. Thus we explore in the current section two NSGA II variants. Secondly, cloud models can be created randomly and evaluated through fitness functions. In what follows, we explore the scalability issues of these different methods.

- 1) Scalability in width: How the quality of results is impacted by the size of the search domain? Does each solution continue to find qualitatively good results with the increase of the search domain?

- 2) Scalability in depth: How the size of the search domain impacts the convergence speed?

In all the experiments, we evaluate the results quality in terms of fitness scores. The execution time of our algorithms is only used to verify our compliance with the time constraints of our cloud case study.

4.2.1 Setup

To answer this research question, we use NSGA II with and without epsilon-dominance and random generation:

- The random generation of 500 solutions using our mutation operators. We consider the best random solutions (over 500 configurations).

- The multi-objective genetic algorithm with 1000 generations on 10 populations (thus 10000 generations) per run, before returning the best solution (over 1000).

- The same multi-objective genetic algorithm but without the boxed epsilon dominance selection. The genetic algorithm considers the mean value of all fitness functions.

In this experiment, we make the infrastructure scale as follows: RQ_2 configuration (scale 1 : 5 infrastructure hosts), 3 times bigger scale (15 hosts), and so on with scale 4, 5, 8 and scale 12 (60 hosts). The random uses 500 generations while the genetic uses 1000 since it is slower, we then adapt random generation to normalize the resolution time. This normalization is necessary since we compare qualitatively results after the same search time for each algorithm.

4.2.2 Results

The results shown in Figure 3 correspond to scale 4 runs and compare the evolution of mean scores for the CMOP-epsilon Dominance [18], CMOP-without Epsilon Dominance and Random. They show the progress of the algorithms to satisfy the objectives. It appears that CMOP-without Epsilon Dominance performs better, which is not surprising since it performs evaluation without using a boxed dominance for solution comparison which makes it faster. However, CMOP-without Epsilon Dominance algorithm often converges to a solution that minimizes one or two objectives perfectly, but degrades the other (starvation of some objectives like the SLA requirements). Thus the CMOP-epsilon appears to be the best solution to avoid having some privileged objectives and to ensure an uniform optimization distribution. For the first scale, the comparison between the three algorithms demonstrates that the best scores are obtained without Epsilon Dominance, with the average of 5.21%, than with CMOP-with Epsilon Dominance algorithm with the average of 6.4%. Random search gives the worst fitness scores, since we achieve an average value of 18%. These experiments with these specific configurations reveal the feasibility of a bounded-in-time reasoning on an abstract representation of a cloud infrastructure. In terms of algorithms comparison, multi-objective optimization outperforms random algorithms. Finally, CMOP-without Epsilon Dominance achieves the best trade-off in terms of computation time, and objectives satisfaction, while it is shown that it is not optimal in terms of mean objectives values. The lesson learned from this experiment is that the scalability can be greatly improved by using an hybrid approach combining 2 steps of CMOP, one with and one without epsilon Dominance to firstly generate good solutions and secondly to refine them with the multi-objective epsilon dominance based-search.

Component	Required CPU (GHz)	Security Level	CPU Load (VCPU %)
ItemDB	1.2	2	40
LoadBalancer	0.4	0	20
PaymentDB	0.6	4	60
UserDB	0.4	3	40
WebFrontend	1.2	1	40

Table 2: Services in the SLA Model

Algorithm	Completeness	Consumption	Overload	Security	SLAPerformance
Mono-objective	0	100.0	24.00	70.0	**0.0**
Multi-objective	0.0	**43.47**	0.0	**0.0**	**0.0**

Table 3: Fitness Vector Values: Mono-objective vs Multi-objective (The values related to the Multi-objective row are better). The row mono-objective only considers SLAPerformance fitness while the row Multi-objective considers all fitness functions. It is worth to note that despite the SLAPerformance (in bold) presents a perfect score for both mono-objective and multi-objective algorithms, the multi-objective reaches better scores for security and global consumption.

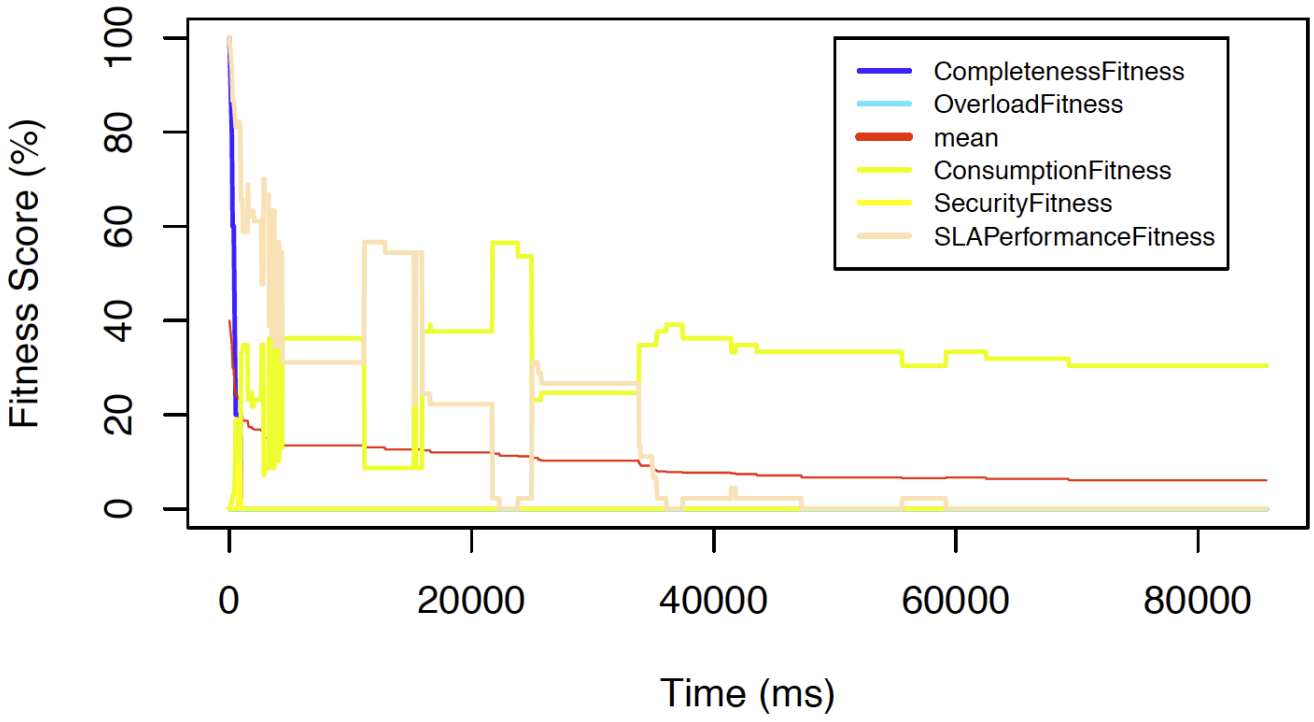

Figure 2: Evolution of the fitness scores of all objective functions and the evolution of the average function(mean) (lower functions correspond to better scores). It is important to notice that in order to reach a global mean (shown in red), the search algorithm satisfies better some objectives to the detriment of others objectives in order to globally reduce the global mean. For instance the consumption at the T=20000ms needs to be increased in order to reduce the mean and customer SLA satisfaction. The consumption increase is explained by the algorithm that tries to find available solutions to host a software component.

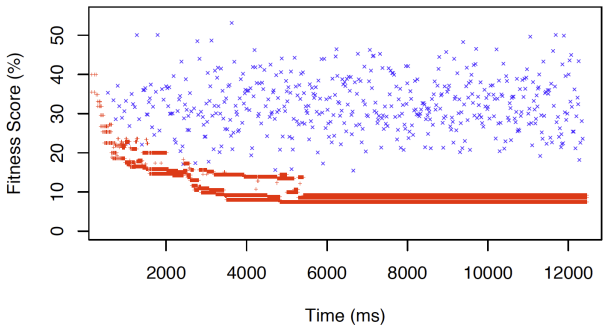

Figure 3: Comparison of multi-objective algorithms, (lower is better). In red, the 5 best mean value of the ε-pareto. In gray the mean value without ε-pareto and in blue the mean value of random solutions. It is important to notice that genetic search is always better than random and that ε-pareto converges slower than CMOP without epsilon Dominance.

Scale	1	3	4	5	8	12
Random	18	16	14.7	16	14.9	14
w/o ε-pareto	6.4	5.9	7.9	7.1	6.69	8.3
w ε-pareto	5.21	6	8.9	8.8	9.4	11

Table 4: Comparison of multi-objective algorithms for the different scales

4.3 Threats to validity

In the present paper, there are many factors that represent a threat to the validity of the obtained results. Internal validity concerns the possible courses of bias of the experiments that were conducted. More precisely, those threats are related to the:

- Number of generations and populations selected at the initial setup.
- Stopping criterion that has been chosen.
- The modeling of fitness functions that have been chosen.

In future work, we intend to conduct experiments on a varied range of populations size. In the current work, the stopping criterion has been based on a timing constraint, we plan to consider other stopping criteria that are based on generation comparison. This comparison will be based on similarity criteria that we aim to define on architectural models. We also plan to consider different other representations of fitness functions and to evaluate solutions quality through quality metrics [6]. External validity are related to the pertinence of the optimization axis that we have chosen and to the resolution strategies that have been used in this paper. Table 4 shows the average of all fitness functions for the three scales and for the different algorithms. Since the configuration combinations are larger, all algorithms tend to find better trade-offs when the scale of the problem increases.

5. RELATED WORK

The dynamic nature of cloud systems presents new challenging issues related to the design and the management of cloud infrastructure. In this work, we have presented the optimization of cloud infrastructure configuration as a multi-objective optimization problem and we have shown how models@run.time is used to provide an abstraction layer to resolve *CMOP*. In [8], the authors have advocated the interest of using model driven engineering to resolve the problem of multi-objective optimization in the cloud. They built an automated model that helps the extraction of available resources in the cloud as a first step to select optimal configurations. In this paper, we go a step forward and we advocate the use of models@run.time to consider different cloud configurations snapshots. In [13], the authors have tackled the problem of software migration in the cloud. They have identified the different criteria, named cloud deployment options (CDOs) to migrate software. They have shown that their simulation-based genetic algorithm CDOXplorer is able to find optimal solutions and thus to optimize CDOs. In [23], the authors have proposed a model for virtual machines allocation and distribution in the cloud environment, based on aspects of performance and budget. They aimed at enabling enough provisioning to finish applications within a desired deadline and at achieving VM consolidation to dispatch workloads so that the global cost is reduced. In our work, we consider problems that go beyond software migration and cost, to cover several cloud supervision aspects such as security. In [29], genetic algorithms have been used to optimize QoS attributes when deploying cloud configurations. Performance configuration has been assessed through queuing theory and history of mean arrival rates, however the authors have not mentioned how their approach can be applied to handle variable aspects in the cloud, like increasing workload. In [31], the authors have tackled a cloud optimization problem aiming at configuring cloud infrastructure for composite SaaS. The authors have used evolutionary algorithms to improve SaaS performance. Their resolution framework differs from our framework since it focuses more on SaaS specific features rather than virtualization environment characteristics.

In [22], the authors have proposed CloudOpt algorithm to optimize applications deployments in the cloud. CloudOpt relies on a combination of bin packing, mixed integer programming and performance models in order to take decisions for a scalable environment and conflicting goals. Compared to our approach, the authors have used integer programming to solve it.

6. CONCLUSION AND FUTURE WORK

In this paper, we have investigated supervision aspects by providing to the customer a decision making framework as a supporting tool to reconfigure his local cloud infrastructure. We have explored the effectiveness of search-based approaches to resolve a cloud multi-objective optimization problem. The framework has been implemented using Kevoree as a Models@run.time platform. The results have shown that the algorithm is able to find nearly-optimal solutions in reasonable time, considered acceptable in the case of cloud dynamic reconfiguration. We are currently extending this work to handle the following aspects:

Stopping criteria and improvements auto evaluation. Defining a stopping criteria for a multi-objective problem is a tedious task. Currently we only consider time constraints or generations number as stopping criteria. We will use ε-box dominance as an indicator to detect the absence of improvements and as a stopping criterion. We also plan to define a stopping criterion based on a defined acceptable values that we will assign to our fitness functions.

Fuzzy logic to express a fitness functions model. Fitness score comparison can introduce a bias in the global optimization process. Even if we have normalized each fitness into a percentage value, the gain represented by each percentage is highly dependent on the fitness implementation. In a future work, we aim at limiting this bias using fuzzy logic techniques as they have been already

successfully used for multi-objective optimization [30].

7. ACKNOWLEDGEMENT

This research is supported by the Fonds National de la Recherche, Luxembourg C12/IS/4011170 in the context of TOOM Core project.

8. REFERENCES

[1] Benoit Baudry, Franck Fleurey, Jean-Marc Jézéquel, and Yves Le Traon. Automatic test cases optimization using a bacteriological adaptation model: Application to .net components. In *Proceedings of ASE'02 (Automated Software Engineering), Edimburgh*, 2002.

[2] Benoit Baudry, Franck Fleurey, Jean-Marc Jézéquel, and Yves Le Traon. Automatic test case optimization: A bacteriologic algorithm. *IEEE Software*, pages 76–82, 2005.

[3] Benoit Baudry, Vu Le Hanh, Jean-Marc Jézéquel, and Yves Le Traon. Building trust into oo components using a genetic analogy. In *ISSRE*, pages 4–14, 2000.

[4] Diana Berberova and Boyan Bontchev. Design of service level agreements for software services. In *Proceedings of the International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing*, pages 26:1–26:6, 2009.

[5] Gordon Blair, Nelly Bencomo, and Robert B France. Models@ run. time. *Computer*, pages 22–27, 2009.

[6] Lucas Bradstreet. *The Hypervolume Indicator for Multi-objective Optimisation: Calculation and Use*. University of Western Australia, 2011.

[7] Edmund K Burke and Graham Kendall. Search methodologies: introductory tutorials in optimization and decision support techniques. Springer, 2005.

[8] Kleopatra Chatziprimou, Kevin Lano, and Steffen Zschaler. Towards a meta-model of the cloud computing resource landscape. In *1st International Conference Model-Driven Engineering and Software Development, Barcelona, Spain, Feb*, pages 19–21.

[9] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast elitist multi-objective genetic algorithm: Nsga-ii. pages 182–197, 2000.

[10] E. Feller, C. Rohr, D. Margery, and C. Morin. Energy management in iaas clouds: A holistic approach. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 204–212, June.

[11] François Fouquet, Erwan Daubert, Noël Plouzeau, Olivier Barais, Johann Bourcier, and Jean-Marc Jézéquel. Dissemination of reconfiguration policies on mesh networks. In *DAIS*, pages 16–30, 2012.

[12] François Fouquet, Brice Morin, Franck Fleurey, Olivier Barais, Noël Plouzeau, and Jean-Marc Jézéquel. A dynamic component model for cyber physical systems. In *CBSE*, pages 135–144, 2012.

[13] Sören Frey, Florian Fittkau, and Wilhelm Hasselbring. Search-based genetic optimization for deployment and reconfiguration of software in the cloud. In *Proceedings of the 2013 International Conference on Software Engineering*, pages 512–521, 2013.

[14] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[15] Mark Harman. The current state and future of search based software engineering. In *2007 Future of Software Engineering*, pages 342–357, 2007.

[16] Mark Harman, Kiran Lakhotia, Jeremy Singer, and Shin Yoo. Cloud engineering is search based software engineering too. *Journal of Systems and Software*, 2012.

[17] George T. Heineman and William T. Councill, editors. *Component-based Software Engineering: Putting the Pieces Together*. 2001.

[18] Christian Horoba and Frank Neumann. Benefits and drawbacks for the use of epsilon-dominance in evolutionary multi-objective optimization. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 641–648, 2008.

[19] Stuart Kent. Model driven engineering. In *Integrated Formal Methods*, pages 286–298. Springer, 2002.

[20] Leonard J LaPadula and D Elliott Bell. Secure computer systems: A mathematical model. *Journal of Computer Security*, pages 239–263, 1996.

[21] Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. Combining convergence and diversity in evolutionary multiobjective optimization. volume 10, pages 263–282, 2002.

[22] Jim Zw Li, Murray Woodside, John Chinneck, and Marin Litoiu. Cloudopt: Multi-goal optimization of application deployments across a cloud. In *Proceedings of the 7th International Conference on Network and Services Management*, 2011.

[23] Ming Mao, Jie Li, and Marty Humphrey. Cloud auto-scaling with deadline and budget constraints. pages 41–48, 2010.

[24] Brice Morin, Olivier Barais, J-M Jézéquel, Franck Fleurey, and Arnor Solberg. Models@ run. time to support dynamic adaptation. *Computer*, pages 44–51, 2009.

[25] A. Ranabahu P. Patel and A. Sheth. Service level agreement in cloud computing. Technical report, Conference on Object Oriented Programming Systems Languages and Applications, Orlando, Florida, 2009, USA.

[26] Mandavilli Srinivas and Lalit M Patnaik. Genetic algorithms: A survey. pages 17–26, 1994.

[27] Le Sun, Jaipal Singh, and Omar Khadeer Hussain. Service level agreement (sla) assurance for cloud services: a survey from a transactional risk perspective. In *Proceedings of the 10th International Conference on Advances in Mobile Computing & Multimedia*, pages 263–266, 2012.

[28] Arie van Deursen, Paul Klint, and Joost Visser. Domain-specific languages: An annotated bibliography. *SIGPLAN Not.*, pages 26–36, 2000.

[29] Hiroshi Wada, Junichi Suzuki, Yuji Yamano, and Katsuya Oba. Evolutionary deployment optimization for service-oriented clouds. *Software: Practice and Experience*, pages 469–493, 2011.

[30] Reay-Chen Wang and Tien-Fu Liang. Application of fuzzy multi-objective linear programming to aggregate production planning. *Computers & Industrial Engineering*, pages 17–41, 2004.

[31] Zeratul Izzah Mohd Yusoh and Maolin Tang. Composite saas placement and resource optimization in cloud computing using evolutionary algorithms. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 590–597, 2012.