

Mixed integer linear programming for quality of service optimization in Clouds

Tom Guérout, Yacine Gaoua, Christian Artigues, Georges da Costa, Pierre Lopez, Thierry Monteil

▶ **To cite this version:**

Tom Guérout, Yacine Gaoua, Christian Artigues, Georges da Costa, Pierre Lopez, et al.. Mixed integer linear programming for quality of service optimization in Clouds. Future Generation Computer Systems, 2017, 71, pp.1-17. 10.1016/j.future.2016.12.034 . hal-01438550

HAL Id: hal-01438550
https://hal.science/hal-01438550

Submitted on 15 Sep 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Mixed integer linear programming for quality of service optimization in Clouds

Tom Guérout[a], Yacine Gaoua[a], Christian Artigues[a], Georges Da Costa[b], Pierre Lopez[a], Thierry Monteil[a]

[a] *LAAS-CNRS, Université de Toulouse, CNRS, INSA, Toulouse, France*
[b] *IRIT Toulouse University, 118 Route de Narbonne, F-31062 TOULOUSE CEDEX 9*

Abstract

The analysis of the Quality of Service (QoS) level in a Cloud Computing environment becomes an attractive research domain as the utilization rate is daily higher and higher. Its management has a huge impact on the performance of both services and global Cloud infrastructures. Thus, in order to find a good trade-off, a Cloud provider has to take into account many QoS objectives, and also the manner to optimize them during the virtual machines allocation process. To tackle this complex challenge, this article proposed a multiobjective optimization of four relevant Cloud QoS objectives, using two different optimization methods: a Genetic Algorithm (GA) and a Mixed Integer Linear Programming (MILP) approach. The complexity of the virtual machine allocation problem is increased by the modeling of Dynamic Voltage and Frequency Scaling (DVFS) for energy saving on hosts. A global mixed-integer non linear programming formulation is presented and a MILP formulation is derived by linearization. A heuristic decomposition method, which uses the MILP to optimize intermediate objectives, is proposed. Numerous experimental results show the complementarity of the two heuristics to obtain various trade-offs between the different QoS objectives.

Keywords: Cloud Computing, Quality of Service, Multiobjective Optimization, Mixed Integer Linear Programming, Genetic Algorithm, DVFS

1. Introduction

Nowadays, the use of Cloud Computing services constantly increases due to their intrinsic facilities proposed to the users. This new paradigm allows a remote access to resources, and the users are able to decide when and how to use these services, without any issue regarding the management of the whole functioning. The main goal of Cloud providers is to attract the maximum number of users by an high level of Quality of Service (QoS), but also to minimize as much as they can the management costs. Indeed, from the perspective of a Cloud provider, the service QoS level management is strongly linked to the users needs, leading to the challenge of continuously providing a QoS level that fit users' expectations in terms of performance, sustainability or dependability. These considerations force the service providers to analyze the characteristics of their systems in order to manage and improve them to find the best trade-off between their own interests and the level of quality of service proposed to the users. The analysis and the management of the global QoS level represent a complex challenge for the provider. Each QoS objective has its own influence on the global behavior of the services, and also on the whole benefits of the providers. Also, the economic aspect of Cloud Computing services not only leads to a financial analysis of the rental cost of services, but also the study of many QoS objectives, which have their influence on the overall performance of the system. The level of quality of service of Cloud Computing services is based on a contract, called SLA (Service Level Agreement). This contract binds the provider of services to its users and contains a set of clauses, which gives information on multiple quality of service areas:

- Hardware performance
- Platform elasticity
- Availability
- Services rent cost, etc.

Email addresses: tguerout@laas.fr (Tom Guérout), ygaoua@laas.fr (Yacine Gaoua), dacosta@irit.fr (Christian Artigues), dacosta@irit.fr (Georges Da Costa), lopez@laas.fr (Pierre Lopez), monteil@laas.fr (Thierry Monteil)

Cloud Computing quality of service objectives can be classified into distinct categories : performance, sustainability, security & data and costs. Each contains different kinds of objectives which have their own influence on the system behavior. The provider must strive to maintain a level of service to first comply with the terms of the SLA promised to users. If some of them are not ensured, it could mean that the providers' platforms are not managed well enough, leading to some potential instabilities during the users utilization. In addition to these QoS objectives that influence the use of the services, the provider has to insure the benefits that he can take from this process. Indeed, the providers' issue is to decrease as much as they can the energy consumption of their platforms which is influenced by the usage of the set of clusters and hosts in their Clouds.

The complexity to manage the level of quality of service relies on the fact that the optimization of each QoS objective can not give an efficient global solution. For example, if the optimization process is only focused on the resources performance, one easy solution is to use a large number of hosts to ensure the stability of the whole clusters in case of any huge peak of load, but it will definitely lead to have a very high energy consumption, which is not an interesting situation in long term for a Cloud provider. Quality of service trade-off are often studied as a conflict relationship between the performance of a system (i.e. response time) and the energy consumption needed to ensure this efficiency. This example is a typical case of conflicting settings resulting in a trade-off between two common QoS objectives. Knowing the level of quality of service promised to the user, the Cloud provider has to be able to take decisions, which do not excessively deteriorate QoS objectives contained in the SLA, while being careful to keep attention on the global yield (performance versus cost, for example) of its services. The example above involves two QoS objectives and is already a complex problem to solve.

Focusing the quality of service optimization only on these two common objectives does not represent a relevant optimization approach, regarding the actual SLA definitions and therefore how a provider has to manage its resources. Taking into account more than the response time and energy consumption metrics could be an interesting approach to improve to the quality of service analysis capabilities of a Cloud provider. Managing in a simple way multiple metrics will be even more important in the near future. In [1], authors evaluate the needs of particular metrics in order to offer real-time characteristics for clouds. They show that adapted metrics, such as the network distance between virtual machines of a single application, are needed to guarantee these characteristics for real-time applications.

In this article, four QoS objectives are taken into account: energy consumption, response time, robustness and dynamism, described in Section 3.2. This set of objectives allows to address the quality of service optimization issue in Cloud while taking into account different QoS categories: performance, dependability and cost. Moreover, most data center operators use simple algorithms such as greedy one to manage their workload. The use of these two multiobjective optimization approaches begets a real investment of the operator to decide to give an overall quality of service level to provide better use conditions to users, while optimizing performance or dependability QoS objectives or to save more energy in order to increase their own benefits and take advantage of this Cloud service rent process. This article proposes a wide analyze of a common heuristic, a Genetic Algorithm (GA), and a MILP based approach, both dedicated to the Cloud Computing context of this article. The elaboration of a MILP based approach is a challenging issue. Indeed, the efficiency of an evolutionary heuristics for a multiobjective optimization is well known due to their fast convergence, even if they do not ensure to return near optimal solutions. The analysis of the MILP based approach compared to the GA allow to study the insights of the two approaches and also to better optimize antagonist Cloud QoS objectives.

The article is organized as follow: Section 2 proposes an overview of quality of service and energy saving studies for Cloud scheduling. Section 3 exposes the problem formulation including the quality of service objectives selected, the modeling of the environment considered and a first mixed integer non-linear programming (MINLP) formulation. Section 4 contains the details of the three solution methods proposed in this article: a Mixed-Integer Linear Program (MILP) obtained by linearizing the MINLP, a MILP-based decomposition method and a Genetic Algorithm. Then, Section 5 presents the experimentation methodology and the results comparison. Finally, conclusions include analyzes on solutions given by the different proposed approaches .

2. Related work

Virtual machine assignment is one of the key issues in cloud computing, in particular because it corresponds to an intractable combinatorial optimization problem. As an illustration, Google organized in 2012 a challenge on a hard practical (virtual) machine reassignment problem in collaboration with EURO/ROADEF operational research societies, for which 30 finalist international teams proposed various optimization methods [2]. The reader can also refer to a recent survey by [3] on virtual machine assignment. In this article, we consider a variant of the virtual machine assignment in which quality of service and energy considerations are put forward. This section describes

few previous works done in the field of quality of service and energy efficiency for virtual machine assignment and scheduling in Clouds. More precisely, the following topics have some similarities with part of our proposal:

- Importance of virtual machine allocation policies for QoS in Clouds [4, 5, 6, 7],
- Decentralized resource allocation approaches for QoS based on negotiation and/or in game theory [8, 9],
- Scheduling, reconfiguration and/or resources allocation algorithms for energy savings under QoS constraints or objectives [10, 11, 12, 13, 14, 15, 16, 17],
- Multiobjective approaches [18, 19, 20, 21],
- Benefits of using DVFS for energy savings in Clouds [22, 23, 24].

Concerning QoS related allocation policies, Islam et al. [4] have developed an elasticity model for cloud instances. They have assumed that each resource type (CPU, memory, network bandwidth, etc.) can be allocated in units and the users are aware of the allocated resources and the relevant QoS metrics for their requests, as in the case of Amazon CloudWatch [1]. Their proposed model combines the cost of provisioned but underutilized resources and the performance degradation cost due to under-provisioning. The consumer's detriment in over-provisioning is the difference between chargeable supply and demand, while the cost of under-provisioning is quantified through the percentage of rejected requests. Authors have also assumed that the customers are able to convert the latter into the estimated financial impact. In [5] the authors introduce a modeling framework called ROAR (Resource Optimization, Allocation and Recommendation System) to simplify, optimize, and automate cloud resource allocation decisions to meet QoS goals for web applications, including complex multi-tier application distributed in different server groups. ROAR uses a domain-specific language to describe the configuration of the web application, the APIs to benchmark and the expected QoS requirements (e.g., throughput and latency), the resource optimization engine uses model-based analysis and code generation to automatically deploy and load tests are applied in multiple configurations in order to derive a cost-optimal resource configuration that meets QoS goals. In [6], Papagianni et al. provide a unified resource allocation framework for networked Clouds. The authors formulate the optimal networked Cloud mapping problem as a mixed integer programming (MIP) problem. Then, they indicate objectives related to cost efficiency of the resource mapping procedure while abiding by user requests for QoS-aware virtual resources. The proposed method for the efficient mapping of resources requests onto a shared substrate interconnecting various islands of computing resources, and adopt a heuristic methodology to address the problem. The efficiency of the proposed approach is illustrated in a simulation/emulation environment, that allows for a flexible, structured, and comparative performance evaluation. In [7], the authors show the impact of the platform choice on the QoS objectives : cost and makespan, that should depend on heterogeneous virtual machines characteristics and on the workload.

In [8] the authors describe a decentralised (P2P) trust model for resource allocation in cloud markets. This model includes mechanisms to allow participants to avoid dishonest behaviour from other peers: each client statistically analyses the external reports about providers and updates the trustworthiness of the peers. The trustworthiness values are used to negotiate prices in later transactions. The trust model is then incorporated in the Service-Level Agreement negotiation and enforcement processes, prioritising trusted clients over non-trusted clients to minimise the consequences of low QoS in relation to the trust of the provider, and incentivise accurate trust reports from the clients. Finally, the authors evaluate and discuss the validity of the trust model under different attacks from dishonest clients and providers. In [9], the authors propose three contributions about QoS optimization domain in a context of Clouds. The fuzzy sets theory is used to express vagueness in the subjective preferences of the customers. The service selection is resolved with the distributed application of fuzzy inference or Dempster-Shafer theory of evidence. The selection strategy is also complemented by the adoption of a game theoretic approach for promoting truth-telling ones among service providers. Experiments results are done through simulations.

Garg et al. [10] have modeled various energy characteristics, such as energy cost, carbon emission rate, workload and CPU power efficiency. Their model considers homogeneous CPUs, a cooling system which depends on a coefficient of performance (COP), and the use of the DVFS. Their performance evaluation includes many metrics and many objectives: average energy consumption, average carbon emission, profit gained, urgency class and

[1] http://aws.amazon.com/fr/cloudwatch/

arrival rate of applications and data transfer cost. Mi et al. [11] have formulated the multi-constraint optimization problem of finding the optimal virtual machine consolidation on hosts while minimizing the power consumption. An application load prediction is computed and they proposed a heuristic based on genetic algorithms in order to find a near optimal reconfiguration policy. The objective function (i.e. fitness) is composed of the power consumption function and a penalty function to keep the CPU utilization between two threshold values. Beloglazov et al. [12] propose a resource management system for efficient power consumption that reduces operating costs and provides quality of service. Energy saving is achieved through the continued consolidation of virtual machines according to resource utilization. The QoS is modeled by the amount of resource needed in Millions Instructions Per Second (MIPS) for the CPU, the amount of Memory (RAM), and by the network bandwidth rate. An SLA violation occurs when a virtual machine cannot have the required three amounts. In the widely cited paper [25], the authors propose multiple energy-aware resource allocation heuristics. Duy et al. [13] design, implement, and evaluate a scheduling algorithm integrating a predictor of neural networks to optimize the power consumption of servers in a cloud. The prediction of future workload is based on demand history. According to the prediction, the algorithm turns off unused servers and restarts them to minimize the number of servers running, thus also minimizing the energy consumption. Srikantaiah et al. [14] study how to obtain consolidation of energy efficiency based on the interrelationship among energy consumption, resource utilization, and performance of consolidated workloads. It is shown that there is an optimal point of operation among these objectives based on the Bin Packing problem applied to the problem of consolidation. Aroca et al. [15] provide a theoretical study of virtual machine allocation problem to minimize the total power consumption on the physical machines. They prove the NP-hardness of the offline variant and perform a competitive analysis of the online version. In [16] the authors propose a framework which shows the capabilities of using different kinds of multi-core CPU (slow/fast cores) for achieving a variety of performance objectives under a power budget and workloads. Their study takes into account different performance goals: large throughput oriented batch jobs and small interactive response-time sensitive jobs. In [17] the authors propose a integer programming based solution approach and a dedicated heuristic to solve a resource constrained scheduling problem corresponding to backup jobs. They consider a trade-off between QoS, performance, and power.

In [18], an energy-aware allocation problem for hosting long-term services or on-demand computing jobs in clusters is addressed. The authors consider two objectives : job performance and energy. Linear programming bounds and polynomial heuristics are proposed to solve this NP-hard problem. About other combinatorial optimization-related literature on the virtual machine allocation problems, the very recent survey [3] reveals that only a few approach deals explicitly with multi objective optimization in the sense of Pareto front computation. A recent purely multiobjective approach uses memetic algorithms [19]. Mezmaz et al [20] propose a new bi-objective hybrid genetic algorithm optimizing two criteria. The energy consumption and the makespan are considered in their optimization process and their approach is based on the island parallel model [26] and the multi-start parallel model. DVFS is also used in their work. In [21], the authors propose to model a data-center and optimize three problems: optimize performance with a bounded power consumption, reduce power consumption with a constraint on the maximum impact on performance or optimize a linear combination of both energy and performance. While this article provides an accurate description of these problems, it is limited by the proposed optimization metrics compared to our proposal.

Abdelsalam et al. [22] have analyzed the mathematical relationship between SLAs and the number of servers used. The energy consumption is also taken into account and their Cloud model uses homogeneous hosts with DVFS enabled, allowing each physical machine to use different frequencies. Their study includes the number of users and their needs, and the average response time of users requests. In [23], Dasgupta et al. pose workload normalization across heterogeneous systems, such as clouds, as a challenge to be addressed. The main differences between the proposed algorithm and other research works are the focus on heterogeneous data-centers and energy benefits provided by active cooling control for unknown sized workload processing. This study takes into account a Return On Investment (ROI) analysis, and also proposes a dynamic power saving case study using the DVFS tools. More recently Ding et al.[24], consider multicore physical machines with DVFS enabled and solve a virtual machine assignment and scheduling problem with deadlines. They propose a scheduling algorithm and give its performance on metrics including energy consumption perfomance-power ratio and execution time among others.

To compare the approach proposed in this article to those describe above, specific and major characteristics of this work, in terms of Cloud architecture, quality of service and optimization methods are the following:

- The QoS objectives taken into account have been selected from the detailed Cloud quality of service analysis presented in [27].

- The objectives allow to involve different QoS aspects defined in SLA Cloud provider : performance, dependability and cost.
- The hosts have the same CPU and memory capacities, but allow the use of the DVFS [28].
- An heterogeneity of hosts power consumption has been considered to approximate the power hosts of real data-center which may be of different generations.
- The DVFS uses the *Userspace* governor, allowing to find the lowest frequency regarding the virtual machines (VM) allocation on physical machines (PM).
- The CPU capacity allocated to the virtual machines is flexible (a decreasing is allowed compared to their maximum CPU capacity).
- A Genetic Algorithm (GA) has been designed to fairly optimize the four QoS objectives.
- A multiobjective decomposition heuristic based on iterative solving of MILP formulations is proposed.

Research work presented in this article has several common aspects with the works cited above, but also ignores some others. An important difference to note is that the notion of a strict violation of SLA is not used in this article. Indeed, SLA violations considered in the work cited above do not fit properly with the providers' SLA clauses. In many cases, they are considered as some performance degradation (i.e. under provisioning of virtual machine) which is not an exact definition of SLA violation. Moreover it implies to define arbitrary thresholds to bind the quality of service accepted level, which is not always easy and relevant. This is why a study of the simultaneous optimization of several QoS objectives was preferred. The main objective of our paper is to assess the performance of the GA and MILP approaches to obtain non-dominant solutions in this above-described context.

3. Problem definition & formulation

Section 3.1 first details the considered Cloud architecture modeling that leads to the definition of several parameters, decision variables and constraints. Then, Section 3.2 defines precisely the four QoS objectives, which are going to be optimized by our approaches. To obtain a more formal and unambiguous definition of the underlying optimization problem to be solved, Section 3.3 presents a multiobjective mixed-integer non-linear programming formulation.

3.1. Cloud Architecture Modeling

In this article we consider a simplified cloud architecture where, only one data center, which contains only one cluster. The cluster is composed of physical machines which can host any of the virtual machines. These physical machines are homogeneous in terms of computing capacity but heterogeneous in terms of power consumption. Let V denote the set of VM and let P denote the set of physical machines. The elementary service running on each VM $v \in V$ is abstracted though an instantaneous maximal CPU capacity requirement c_v and a number of instructions τ_v. In the model we consider, the actual CPU requirement of the virtual machine can be partially controlled as it can take any value above a certain percentage of its maximum requirement. This is called the reconfiguration process. Let $\rho_v \in [R, 1]$ denote the applied decrease rate (to be decided) where $R \in (0, 1]$. The VM actually requires $r_v\rho_v$ of CPU capacity on the allocated PM during $\frac{\tau_v}{c_v\rho_v}$ time units. Furthermore, each VM occupies a fixed memory amount denoted by m_v.

Each PM $p \in P$ has a maximal CPU capacity C_p and a maximum memory capacity M_p. There is a discrete set F of available processor frequencies such that selecting frequency f yields a CPU capacity of $\pi_f C_p$, with a predefined $\pi_f \in [0, 1]$, for each $f \in F$. For a selected frequency $f \in F$, the power consumed by the CPU of a PM p varies proportionally to its utilization rate, with a minimum power $P^f_{\xi(p),\min}$ for a CPU utilization rate of 0% while the power for a full processor utilization if equal to $P^f_{\xi(p),\max}$, where $\xi(p)$ is the type of the PM with $\xi(p) \in \mathcal{T}$ and $\mathcal{T}$ is the set of PM types where $|\mathcal{T}| << |P|$. The PM can be switched off if no VM is assigned to it.

The problem we consider can be informally stated as follows. Each VM has to be assigned one PM. Each PM must be assigned one frequency. For each VM assigned to the same PM the same CPU requirement decrease rate has to be selected (which amounts to select a decrease rate for each PM). The total CPU requirement of the VM assigned to a PM (modulated by the decrease rate) cannot exceed its capacity (determined by the selected frequency). The PM computing capacity is straight related to its frequency, which is chosen among the set of available frequencies. This CPU frequency management and the VM allocation are done only once time at $t = 0$

and then do not change during until the end of the all VM execution. This kind of CPU frequency management allows to reproduce the *Userspace* DVFS governor behavior. Indeed, this is a static approach which can be smartly implemented in the two proposed methods of this article. Also, no decrease rate should be applied to the VM assigned to a PM if the CPU capacity of the PM is not exceeded. Similarly, the total memory requirement of the VM assigned to a PM cannot exceed the PM memory capacity.

3.2. QoS Scheduling objectives

This section lists the four QoS objectives chosen to be optimized during the virtual machine assignment process:

- *Energy Consumption*: The total energy consumption is the sum of each physical machine energy consumption ($E = \sum_{p \in P} E_p$) and its computation uses the following power formula $P^f_{\xi(p)} = \alpha \left(P^f_{\xi(p),\max} - P^f_{\xi(p),\min} \right) + P^f_{\xi(p),\min}$ which correspond to the well known affine power model [29], where f is the current CPU frequency used and α is the CPU load. The energy consumption of each host is computed proportionally to their power values in time and their total execution time. This metric has to be minimized to minimize the energy consumption.

- *Response Time*: The response time, denoted t_{max}, is considered as the execution time of VM, knowing the MIPS capacity of these VM and the number of instructions (in Million of Instructions) they have to execute. It can be expressed (in seconds) as follows: $\frac{\tau_v}{c_v \rho_v}$. Note that the response time metric is here the execution total time the longest VM takes while usually the literature uses this term as the end user response time. This expression has been adopted for the response time metric in this article as the Cloud model used for now is going to be improved in future work with more complex Cloud services representation like composition of several services, DAG execution, etc. Indeed, the metric response time is designed to be used both with simple and more complex services modeling.
  This metric has to be minimized to minimize the Response Time.

- *Robustness*: The robustness, denoted *Rob*, is interpreted as how many VM should be disposed if a host failure happens. In other words, it is the average number of VM per used host. This metric has to be minimized to maximize the Robustness.

- *Dynamism*: The dynamism (or Latent Capacity), denoted *Dyn*, is the average amount of free MIPS on each powered ON host that can be used in a case of a peak of request rate arrival. It is expressed in free MIPS per Host. This metric has to be maximized to maximize the Dynamism.

The aim of this multiobjective problem is to balance the optimization without giving an advantage to one of these four QoS objectives. As explained in each definition of these QoS objectives, the energy consumption, the response time and the robustness have to be minimized, contrarily to the dynamism that has to be maximized. These objectives can be antagonist and, for instance, the minimum energy consumption does not always yield the shortest execution time. Moreover, it is very interesting to add two uncommon QoS objectives (robustness and dynamism) in a context of Cloud to common objectives already in conflict.

3.3. Multiobjective mixed-integer non-linear programming formulation

Below we give a mathematical formulation of the problem using binary assignment variables x_{vp} for each $v \in V$ and $p \in P$ where $x_{vp} = 1$ if virtual machine v is assigned to physical machine p and binary assignment variables $y_{pf} = 1$ if frequency f is selected for PM p. Continuous variable ρ_p gives the decrease rate for the CPU requirement of all VM assigned to PM p. Auxiliary variables t_p, for each $p \in P$ and $t_{\max}$ are used to represent the time to perform all VM on PM p and the global response time, respectively. Auxiliary variable z_p indicates whether PM p is on or off.

$$\min E = \sum_{p \in P} \sum_{v \in V} \sum_{f \in F} \frac{(P^f_{\xi(p),\max} - P^f_{\xi(p),\min})\tau_v}{\pi_f C} x_{vp}\, y_{pf} + \sum_{p \in P} \sum_{f \in F} P^f_{\xi(p),\min}\, y_{pf}\, t_p \tag{1}$$

$$\min t_{\max} \tag{2}$$

$$\min Rob = \frac{|V|}{\sum_{p\in P} z_p} \tag{3}$$

$$\max Dyn = \sum_{p\in P} z_p \frac{(C_p - \sum_{v\in V} c_v\,\rho_p\,x_{vp})}{\sum_{p\in P} z_p} \tag{4}$$

subject to

$$\sum_{p\in P} x_{vp} = 1 \qquad \forall v \in V \tag{5}$$

$$\sum_{f\in F} y_{pf} = 1 \qquad \forall p \in P \tag{6}$$

$$\sum_{v\in V} c_v\,\rho_p\,x_{vp} - \sum_{f\in F} \pi_f\,C_p\,y_{pf} \quad \leq 0 \qquad \forall p \in P \tag{7}$$

$$\sum_{v\in V} m_v\,x_{vp} - M_p \quad \leq 0 \qquad \forall p \in P \tag{8}$$

$$t_p - \frac{x_{vp}\,\tau_v}{c_v\,\rho_p} \quad \geq 0 \qquad \forall v \in V, p \in P \tag{9}$$

$$t_{\max} \quad \geq t_p \qquad \forall p \in P \tag{10}$$

$$z_p \leq \sum_{v\in V} x_{vp} \qquad \forall p \in P \tag{11}$$

$$z_p \geq x_{vp} \qquad \forall p \in P, v \in V \tag{12}$$

$$x_{vp} \in \{0,1\} \qquad \forall v \in V, p \in P \tag{13}$$

$$y_{pf} \in \{0,1\} \qquad \forall p \in P, f \in F \tag{14}$$

$$R \leq \rho_p \leq 1 \qquad \forall p \in P \tag{15}$$

There are three objective functions to be minimized: the energy consumption (1), the response time (2) and the robustness (3) given by the average number of VM assigned to the PM that are switched on. There is one objective function to be maximized, the dynamism (4), given by the average remaining CPU capacity on the PM that are switched on.

Constraints (5) state that each VM has to be assigned to exactly one PM. Constraints (6) state that a frequency has to be selected for each PM. Capacity constraints (7) state that for a PM p, the sum of the CPU requirement of each VM assigned to p must not exceed the CPU capacity of PM p. Note that the first term is non-linear as product $\rho_p\,x_{vp}$ appears. Constraints (8) express the memory limitations. Constraints (9) set the utilization time of PM p to be at least the processing time of each VM. The constraint is non linear as the quotient x_{vp}/ρ_p or alternatively the product $t_p\,\rho_p$ is present. Constraints (10) set the response time to be at least the utilization time of each PM. Constraints (11-12) express z_p as the 0-1 indicator of whether PM $p \in P$ is on or off.

We now explain how we obtain the expression of the energy part of the objective function. We define the consumed energy E as $\sum_{p\in P} E_p$ where

$$\begin{aligned} E_p = &\sum_{f\in F} \frac{(P^f_{\xi(p),\max} - P^f_{\xi(p),\min})y_{pf}}{\pi_f\,C_p} \sum_{v\in V} x_{vp}\,\tau_v \\ &+ \sum_{f\in F} P^f_{\xi(p),\min}\,y_{pf}\,t_p \end{aligned} \tag{16}$$

is the energy consumed by PM p. The right term represents the energy consumed just because the PM is switched-on during t_p time at the minimum power set by the selected frequency. The term is non linear as t_p is

multiplied by y_{pf}. The left term represents the sum of the individual energy consumptions of the VM assigned to p. The total power available for the VM on PM p at a given time depends on the selected frequency and is given by term

$$P_p = \sum_{f \in F} (P^f_{\xi(p),\max} - P^f_{\xi(p),\min}) y_{pf} \tag{17}$$

The power used by a VM assigned to p is equal to $R_{vp}\, P_p$ where

$$R_{vp} = \frac{x_{vp}\, c_v\, \rho_p}{\sum_{f \in F} \pi_f\, C_p\, y_{pf}} \tag{18}$$

represents the percentage of the CPU capacity used by v on p. Finally the energy consumed by VM v on PM p is $R_{vp}\, P_p\, t_v$ where

$$t_v = \frac{\tau_v}{\rho_p\, c_v} \tag{19}$$

is the time needed to complete v if it is assigned to p. Developing this expression the factors $\rho_p\, c_v$ are eliminated, which yields

$$\begin{aligned} E_p = & \sum_{v \in V} x_{vp}\, \tau_v \frac{\sum_{f \in F} (P^f_{\xi(p),\max} - P^f_{\xi(p),\min}) y_{pf}}{\sum_{f \in F} \pi_f\, C_p\, y_{pf}} \\ & + \sum_{f \in F} P^f_{\xi(p),\min}\, y_{pf}\, t_p \end{aligned} \tag{20}$$

we now observe that, as (14) and (6) hold, we have the following additional simplification.

$$\frac{\sum_{f \in F} (P^f_{\xi(p),\max} - P^f_{\min}) y_{pf}}{\sum_{f \in F} \pi_f\, C_p\, y_{pf}} = \sum_{f \in F} \frac{(P^f_{\xi(p),\max} - P^f_{\xi(p),\min}) y_{pf}}{\pi_f\, C_p} \tag{21}$$

which yields the desired expression.

This model (which is NP-hard as it contains a quadratic assignment problem) is intractable for practical instances. Therefore we propose a method based on linearization and decomposition of the MINLP and a metaheuristic (genetic algorithm).

4. Solution Methods

4.1. Towards a Mixed-Integer Linear Program (MILP)

Standard linearization techniques can be applied to the MINLP (1-15). We first perform a discretization of the reconfiguration rate domain $[R, 1]$, obtaining a discrete set of reconfiguration values $\{\alpha_l\}_{l \in L}$ of index set L. A new binary variable $z_{pl} \in \{0, 1\}$ is introduced, that takes value 1 when the reconfiguration rate α_l is applied to all the VM assigned to the PM p. We have to introduce a constraint to select a unique reconfiguration rate for each PM

$$\sum_{l \in L} z_{pl} = 1 \qquad \forall p \in P \tag{22}$$

With this new variable the non-linear constraint (7) can be formulated as the following linear constraint:

$$\sum_{v \in V} \sum_{l \in L} c_v\, \alpha_l\, w_{pvl} - \sum_{f \in F} \pi_f\, C_p\, y_{pf} \leq 0 \quad \forall p \in P \tag{23}$$

where w_{pvl} is standardly set equal to the product of binary variables $z_{pl}\, x_{vp}$ with linearization constraints:

$$w_{pvl} = \begin{cases} x_{vp} + z_{pl} \leq w_{vpl} + 1 & \forall p \in P, v \in V, l \in L \\ w_{vpl} \leq x_{vp} & \forall p \in P, v \in V, l \in L \\ w_{vpl} \leq z_{pl} & \forall p \in P, v \in V, l \in L \end{cases}$$

Another standard linearization process can be applied to constraint (9).

$$t_p - \frac{x_{vp}\,\tau_v}{c_v\,\rho_p} \geq 0 \equiv t_p\,c_v \sum_{l \in L} z_{pl}\,\alpha_l \geq x_{vp}\,\tau_v \qquad \forall v \in V, p \in P \tag{24}$$

Linearizing the product of a binary variable and a continuous variable $e_{pl} = z_{pl}\,t_p$, we obtain the following constraints:

$$\sum_{l \in L} e_{pl}\,\alpha_l\,c_v \;\geq x_{vp}\,\tau_v \qquad \forall v \in V, p \in P \tag{25}$$

$$t_p + T_p(z_{pl} - 1) \leq e_{pl} \qquad \forall p \in P, l \in L \tag{26}$$

$$e_{pl} \leq T_p z_{pl} \qquad \forall p \in P, l \in L \tag{27}$$

$$e_{pl} \leq t_p \qquad \forall p \in P, l \in L \tag{28}$$

where T_p is an upper bound on the time needed to complete all VM on machine p.
For the objective functions, the linearization of $x_{vp}y_{pf}$ can be achieved by introducing a new variable γ_{vpf} set equal to $x_{vp}y_{pf}$ by the following linear constraints:

$$x_{vp} + \gamma_{vpf} \leq y_{pf} \qquad \forall p \in P, f \in F \tag{29}$$

$$\gamma_{vpf} \leq x_{vp} \qquad \forall p \in P, f \in F \tag{30}$$

$$\gamma_{vpf} \leq y_{pf} \qquad \forall p \in P, f \in F \tag{31}$$

Similarly, the linearization of $y_{pf}t_p$ needs variable $\theta_{pf} = y_{pf}t_p$ as expressed by:

$$t_p + T_p(y_{pf} - 1) \leq \theta_{pf} \qquad \forall p \in P, f \in F \tag{32}$$

$$\theta_{pf} \leq T_p y_{pf} \qquad \forall p \in P, f \in F \tag{33}$$

$$\theta_{pf} \leq t_p \qquad \forall p \in P, f \in F \tag{34}$$

This yields a linear expression of energy objective (1). For the robustness objective (3) a linear objective is obtained by maximizing the inverse of the quotient ($\max \sum_{p \in P} z_p/|V|$). The linearization of the dynamism objective is more complex and we do not present it for sake of concision and also as the MILP-based decomposition method (presented in the next section) that is able to solve the realistic instances does not need such linearization.

One of the specification of the problem is that the CPU capacity of the physical machine $\pi_f C_p$ has to be saturated if a reconfiguration rate is applied. The discretization, besides the fact that it may yield a suboptimal solution, may also violate this constraint. Consequently we apply a post-optimization procedure to adjust the reconfiguration rate to continuous values. We will describe the post-optimization procedure in the next section as it is also part of the proposed decomposition procedure.

4.2. MILP-based Decomposition

Since the MILP described above is only able to solve small instances, we present in this section a MILP-based decomposition heuristic that consists of iteratively solving subproblems in which some parts of the problem have been fixed. To simplify the solution process and strengthen the MILP relaxation, we systematically work at fixed robustness. More precisely we heuristically fix at each iteration the number of PM that are switched on in each of the types $\xi \in \mathcal{T}$. Let n_ξ the number of PM of type $\xi \in \mathcal{T}$. At each iteration, we enforce to switch $\overline{n}_\xi$ PM on with $\overline{n}_\xi \leq n_\xi$. As all PM of the same types are equivalent (they have the same CPU and memory capacities), this amounts to consider a reduced set of PM for each type, as the PM that are fixed as switched off can be arbitrarily selected and removed from the problem.

Initially, we set the number of switched on PM to $|P|$ (i.e $\overline{n}_\xi = n_\xi, \forall \xi \in \mathcal{T}$), we also fix the PM frequencies (to their maximum values) and the reconfiguration rates (no reconfiguration). With all these fixed parameters, we solved the VM/PM assignment problem through a first MILP. Once the assignment problem is solved, we fixed the obtained assignment and we solve a second MILP to optimize the discretized reconfiguration rates and the PM frequencies. Then we solve a third MILP with fixed assignment and time to adjust the reconfiguration rate in the continuous domain and recompute the PM frequencies (already mentioned post-optimization phase). Then, we restart the process at the assignment phase with the so-computed reconfiguration rates and frequencies. Globally we perform the first q iterations with the maximum number of switched on PM, then we decrease this number by ν percent for the q subsequent iterations, and then we decrease again the number of switched on PM by ν percent until a maximum number of iterations is reached. We spread the global ν percent decrease evenly over all PM types.

Hence each q iterations the robustness level is changed. We described below the three MILPS and the objective functions that we implemented to tackle the other multiobjective aspects.

For the assignment phase, we work with a fixed reconfiguration rate $\overline{\rho}_p$ and a fixed frequency $\overline{f}_p$, $\forall p \in P$. It follows that, as already mentioned, the robustness is fixed. Furthermore, the dynamism criterion is now linear. We propose to minimize at each iteration a weighted sum of the energy, time and dynamism objective. This gives the following MILP, where w^{EN} is the energy weight, w^{TI} is the time weight and w^{DY} is the dynamism weight. We denote by $\overline{P} \subseteq P$ the set of physical machines that are fixed as switched-on at the current iteration, with $\sum_{\xi \in \mathcal{T}} \overline{n}_\xi = |\overline{P}|$. All other PM are not included in the problem.

$$\begin{aligned} \min\ & w^{EN}\Big(\sum_{p\in\overline{P}} \sum_{v\in V} \frac{1}{\pi_{\overline{f}_p}\, C_p} (P^{\overline{f}_p}_{\xi(p),\max} - P^{\overline{f}_p}_{\xi(p),\min}) \tau_v\, x_{vp} \\ & + \sum_{p\in\overline{P}} P^{\overline{f}_p}_{\xi(p),\min} t_p \Big) \\ & + w^{TI} t_{\max} - w^{DY} \sum_{p\in\overline{P}} \frac{1}{|\overline{P}|} (C_p - \sum_{v\in V} c_v\, \overline{\rho_p}\, x_{vp}) \end{aligned} \tag{35}$$

$$\sum_{p\in\overline{P}} x_{vp} = 1 \qquad \forall v \in V \tag{36}$$

$$\sum_{v\in V} x_{vp} \geq 1 \qquad \forall p \in \overline{P} \tag{37}$$

$$\sum_{v\in V} c_v\, \overline{\rho_p}\, x_{vp} - \pi_{\overline{f}_p}\, C_p \quad \leq 0 \qquad \forall p \in \overline{P} \tag{38}$$

$$\sum_{v\in V} m_v\, x_{vp} - M_p \quad \leq 0 \qquad \forall p \in \overline{P} \tag{39}$$

$$t_p - \frac{x_{vp}\, \tau_v}{c_v\, \overline{\rho_p}} \quad \geq 0 \qquad \forall v \in V, p \in \overline{P} \tag{40}$$

$$t_{\max} \quad \geq t_p \qquad \forall p \in \overline{P} \tag{41}$$

$$x_{vp} \in \{0,1\} \qquad \forall v \in V, p \in \overline{P} \tag{42}$$

As all machines in $\overline{P}$ have to be switched on, we can add constraint (37) that state that at least one VM has to be assigned to each PM. With fixed reconfiguration rates and frequencies, capacity constraints (38) become knapsack constraints just as constraints (39). For the objective function, we have assignment costs in the form of min sum for the variable part of the energy cost and the dynamism cost but also in the form of min max for the fixed part of the energy cost and the time cost. Along the iterations, we alternate four values for the weight vector. A first value is an equal weight for the three objectives after normalization w.r.t. upper bounds of the objective values. The three other values (1,0,0), (0,1,0) and (0,0,1) correspond to successive optimization of each criterion alone.

Once the assignment has been computed by the above-defined MILP, a second MILP is set up to change the reconfiguration rates and the frequencies, given a fixed assignment $\overline{x}_{vp}$. From MINLP (1-15), and the linearization techniques described above, we obtain the following MILP. Let $\overline{V}(p)$ denote the set of VM assigned to PM p.

$$\begin{aligned} \min\ & w^{EN}\Big(\sum_{p\in\overline{P}} \sum_{v\in\overline{V}(p)} \sum_{f\in F} \frac{1}{\pi_f\, C_p} (P^f_{\xi(p),\max} - P^f_{\xi(p),\min}) \tau_v\, y_{pf} \\ & + \sum_{p\in\overline{P}} \sum_{f\in F} P^f_{\xi(p),\min} \theta_{pf} \Big) \\ & + w^{TI} t_{\max} - w^{DY} \sum_{p\in\overline{P}} \frac{1}{|\overline{P}|} (C_p - \sum_{v\in\overline{V}(p)} \sum_{l\in L} c_v \alpha_l\, z_{pl}\,) \end{aligned} \tag{43}$$

$$\sum_{v\in\overline{V}(p)} \sum_{l\in L} c_v\, \alpha_l\, z_{pl} - \sum_{f\in F} \pi_f\, C_p\, y_{pf} \leq 0 \qquad \forall p \in \overline{P} \tag{44}$$

$$\sum_{l \in L} e_{pl}\, \alpha_l\, c_v \;\geq \tau_v \qquad \forall p \in \overline{P}, v \in \overline{V}(p), \tag{45}$$

$$t_p + T_p(z_{pl} - 1) \leq e_{pl} \qquad \forall p \in \overline{P}, l \in L \tag{46}$$

$$t_{\max} \geq t_p \qquad \forall p \in \overline{P} \tag{47}$$

$$e_{pl} \leq T_p z_{pl} \qquad \forall p \in \overline{P}, l \in L \tag{48}$$

$$e_{pl} \leq t_p \qquad \forall p \in \overline{P}, l \in L \tag{49}$$

$$t_p + T_p(y_{pf} - 1) \leq \theta_{pf} \qquad \forall p \in \overline{P}, f \in F \tag{50}$$

$$\theta_{pf} \leq T_p y_{pf} \qquad \forall p \in \overline{P}, f \in F \tag{51}$$

$$\theta_{pf} \leq t_p \qquad \forall p \in \overline{P}, f \in F \tag{52}$$

$$\theta_{pf} \geq 0 \qquad \forall p \in \overline{P}, f \in F \tag{53}$$

$$e_{pl} \geq 0 \qquad \forall p \in \overline{P}, l \in L \tag{54}$$

$$t_p \geq 0 \qquad \forall p \in \overline{P} \tag{55}$$

$$t_{\max} \geq 0 \tag{56}$$

$$y_{pf} \in \{0, 1\} \qquad \forall p \in \overline{P}, f \in F \tag{57}$$

$$z_{pl} \in \{0, 1\} \qquad \forall p \in \overline{P}, l \in L \tag{58}$$

Note that as the assignment is fixed, the above-defined MILP has a reasonable number of binary variables, provided that the set of frequencies and the set of discretized configuration values are not too large.

The last step is to perform the post-optimization phase to obtain adjusted reconfiguration rates, so as to saturate PM capacity whenever reconfiguration is applied. To reintroduce the continuous variable ρ_p, we keep the assignment variables x_{vp} to the values $\overline{x}_{vp}$, defining sets $\overline{V}(p)$, found by the previous MILP. We fix time variable t_p to the value found ($\overline{t}_p$). Note that the time objective does not appear in the cost function as it is upper bounded and that considering the time variable would yield a product of continuous variables $t_p \rho_p$. Hence only the energy and dynamism objectives could remain. However the dynamism objective would tend to decrease the reconfiguration rate even if the capacity constraint is not saturated, which is not permitted. We only keep the energy objective and add a maximization objective on the reconfiguration rate with a weight W setup in such a way that this objective has less importance than the energy one. Observing the obtained MILP shows that, as the capacity constraints are satisfied by the assignment, the solution resorts to selecting the largest reconfiguration value and the smallest π_f satisfying the capacity constraint, which naturally ensures that the capacity constraints are saturated if the reconfiguration rate is strictly smaller than 1.

$$\begin{aligned} \min\; & w^{EN}\Big(\sum_{p \in P} \sum_{v \in \overline{V}(p)} \sum_{f \in F} \frac{(P^f_{\xi(p),\max} - P^f_{\xi(p),\min})\tau_v}{\pi_f\, C} y_{pf} \\ & + \sum_{p \in P} \sum_{f \in F} P^f_{\xi(p),\min} \overline{t}_p\, y_{pf}\Big) - W \sum_{p \in \overline{P}} \rho_p \end{aligned} \tag{59}$$

$$\sum_{v \in \overline{V}(p)} c_v\, \rho_p - \sum_{f \in F} \pi_f\, C_p\, y_{pf} \leq 0 \qquad \forall p \in \overline{P} \tag{60}$$

$$\sum_{f \in F} y_{pf} = 1 \qquad \forall p \in \overline{P} \tag{61}$$

$$\rho_p \geq \frac{\tau_v}{t_p c_v} \qquad \forall p \in \overline{P}, v \in \overline{V}(p) \tag{62}$$

$$y_{pf} \in \{0, 1\} \qquad \forall p \in P, f \in F \tag{63}$$

$$R \leq \rho_p \leq 1 \qquad \forall p \in P \tag{64}$$

All parameters and decision variables used in the three sections 3.3, 4.1 and 4.2 are summarized through two tables in the Appendix (Table 9 and Table 10).

4.3. Genetic Algorithm approach

The Genetic Algorithm [30] used and compared to the linear programming approach in this article has been fully implemented in C++ language. Details of the modeling, illustrated in Figure 1, and the use of the common operators of the GA dedicated to the VM allocation problem and the Cloud QoS objective optimization have already been presented in [27]. This is why this section is focused on the most important characteristics of this GA and highlights how to evaluate and compares the obtained solutions. Note that other multiobjective heuristics with elitist and/or non-dominated sorting could have been used such as NSGA-II [31].

4.3.1. Solution representation & Basic characteristics

A virtual machine allocation solution on hosts is represented by a chromosome, which is composed of a set of genes. The size of a chromosome (i.e. the number of genes) indicates the number of virtual machines. The value assigned to each gene is the index of the host where the virtual machine (represented by the gene) has to be allocated. Indeed, a complete solution is composed of a set of genes associated to a value which represents the host where to each virtual machine has to be executed (Figure 1). Common operators of a GA, mutation and crossover (illustration in Figure 2) have been also implemented. An important phase of the resolution process is to verify the veracity of a such representation. It leads to check if the whole allocation of all virtual machine do not violate the constraints in terms of CPU (depending on the chosen frequency) and the memory capacities. This process takes into account that the virtual machine CPU capacity allocated can be reduced up to 20% (0% to 20%). Indeed, the frequency chosen by the GA for each physical machine is the lowest that yields the selected CPU virtual machine reconfiguration. This important phase has be to done every time before adding a chromosome into the current working population.

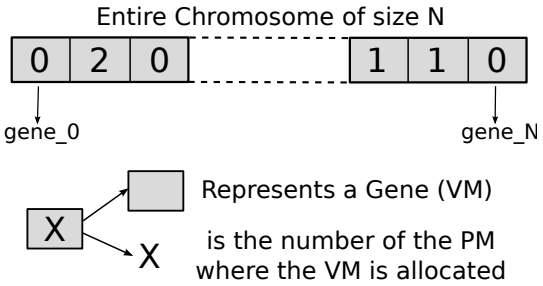


Figure 1: Chromosome solution representation

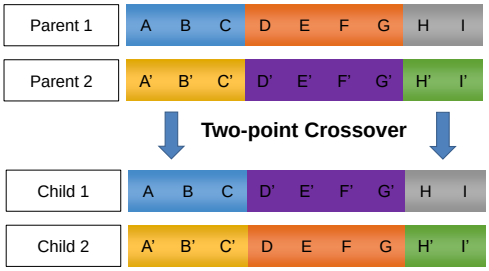


Figure 2: Two-point Crossover operator

The basic characteristics used in this GA implementation are the following:

- The initial population is composed of 1500 chromosomes
- The working population uses 120 chromosomes
- 100 mutations and 100 crossovers are applied during each generation
- The GA is stopped when 600 generations are done

4.3.2. Objective function & chromosome evaluation

The objective function has been defined as follows:

$$F_{obj} = w^{EN} \times E + w^{TI} \times t_{\max} + w^{RO} \times Rob - w^{DY} \times Dyn \tag{65}$$

where w^{EN}, w^{TI}, w^{RO} and w^{DY} are the coefficients of the energy, the response time, the robustness and the dynamism respectively. These coefficients can be modified (one greater than the other for example) to advantage the optimization of one or several QoS objectives. The value given by this objective function is called the *fitness* value and allows to evaluate the quality of the solution obtained.

Each metric has its own real value. Indeed, each metric does not have the same range of value, as so if the fitness value is computed keeping these different ranges of value, the result obtained would make no sense. The standardization (Eq. 66) step of each metric value, in order to compute a reliable chromosome fitness values has been done with the "center and reduce" formula. This gives a set of values with an average value of 0, a variance

value of 1 and standard deviation value of 1. Then, all metric values are comparable therebetween and the fitness value of a chromosome can be computed.

$$v_{std} = \frac{v - \mu}{\sigma} \tag{66}$$

where, v is the value to standardize, μ the average and σ the standard deviation.

4.3.3. Optimization configurations

Five versions of the genetic algorithm have been defined, so as each of them uses different coefficient weights associated to the metrics (Table 1). This leads to have five different kinds of QoS metrics optimization. These five versions are described below:

- *GA_Energy* only optimizes the energy consumption metric,
- *GA_RespT* only optimizes the response time metric,
- *GA_Rob* only optimizes the robustness metric,
- *GA_Dyn* only optimizes the dynamism metric,
- *GA_All* fairly optimizes the four metrics.

Table 1: Coefficient values of the 5 versions of the GA

GA Name	**Coefficients applied to metrics**			
	Energy	**Response time**	**Robustness**	**Dynamism**
GA_All	1	1	1	1
GA_Energy	1	0	0	0
GA_RespT	0	1	0	0
GA_Rob	0	0	1	0
GA_Dyn	0	0	0	1

4.3.4. Analysis and comparison of solutions quality

As explained in the previous section, the quality of an allocation solution given by the GA is represented by its fitness value. With these five different configurations of optimization, the aim was to highlight the relevance to optimize these four metrics simultaneously, and also to show the influence of the optimization on each other. In order to estimate the efficiency of each version of the GA, a comparison has been done using an increasing number of virtual machines (50 to 400) to assign into a fix number of 110 hosts. In the Figure 3, these eight different size problems correspond to the X axis values. The efficiency of each GA is represented by its fitness value in the Y axis. The fitness values normalization have been done using the range [min;max] of each metric of each run. Indeed, the smaller the fitness value, the better the optimization potential regarding the four QoS metrics. The usefulness of this figure is to give relevant analyzes about the optimization quality of version of the GA, in different virtual machines allocation conditions. In a multi-criteria point of view, it is very interesting to note that the *GA_All* version gives a nice trade-off between each metric, with the best fitness value for each of the eight configurations.

The next section presents more detailed results and a comparison between the GA and the MILP-based approaches.

5. Computational Experiments

In this Computational Experiments section, the first part details the methodology adopted and the second one proposes an extended analyzes of the whole results obtained.

5.1. Methodology

This section first exposes the methodology used for all the computational experiments in terms of virtual machines and hosts characteristics (numbers, capacities, power and heterogeneity), and the manner of each optimization method has been exploited.

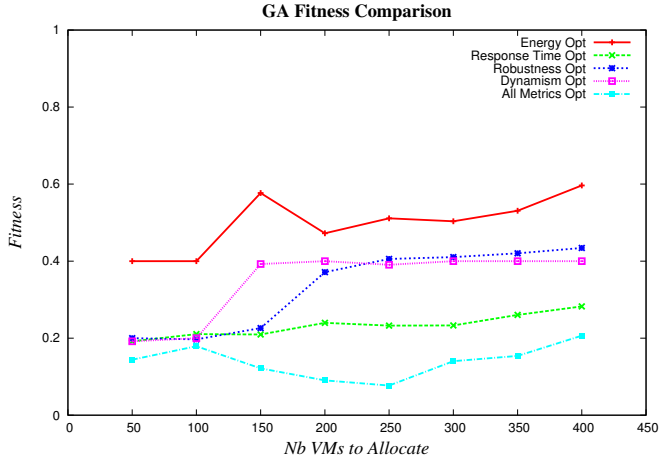


Figure 3: Comparison of Fitness value results between the five GA.

5.1.1. Hosts and virtual machines characteristics

In order to be able make this comparison in safe and relevant conditions, the GA and the MILP-Based Decomposition have been executed using the same input configuration of hosts and virtual machines characteristics. First, five configurations of the allocation problem have been defined:

- 5 hosts / 15 virtual machines
- 25 hosts / 90 virtual machines
- 55 hosts / 200 virtual machines
- 80 hosts / 300 virtual machines
- 110 hosts / 400 virtual machines

About the physical machines, the same CPU and memory capacities of 2000 MIPS and 2500 Mo have been used, respectively. One host is supposed to have one CPU in which five different frequencies are enable. The CPU (in MIPS) and memory (in Mo) capacities of the virtual machines can both take one of the four following values: $[200, 400, 600, 800]$. Regarding the hosts power characteristics, the model of the physical machine of the Grid'5000 [32] Reims site has been used. Measurements on this Grid'5000 cluster have been done to have power values at *idle* and *full* states for the five CPU frequencies. Details on available frequencies, and the corresponding power values are resumed in the Table 2. Concerning the DVFS, it has been used in *Userspace* mode (five modes exist). This mode is a *static* mode and allows to choose one of the frequencies available.

Table 2: Grid'5000 Reims site frequencies and power characteristics

Grid'5000 Reims site						
Available Frequencies (GHz)		0.8	1.0	1.2	1.5	1.7
Power (W)	***Idle***	140	146	153	159	167
	Full	228	238	249	260	272

Based on these real power values, five types [0 to 4] of hosts have been defined in order to represent the variety of an actual Cloud data-center in terms of hosts generation and their energy consumption heterogeneity. It means that an host can deliver more or less power (for the same CPU capacity). Type 2 corresponds to the real base model (0% of heterogeneity). The four other types have an heterogeneity between -20% and +20% compared to the type 2. The Table 3 summarizes this power heterogeneity of the physical machines.

The VM CPU capacity reconfiguration concept is also used (Equation 15). It allows to decrease the CPU capacity of a virtual machine. This tool is used only when the sum of a VM set allocated on physical machine exceed its current capacity (Equation 7). Then, the reconfiguration value ρ_p is computed. The decreasing of the CPU

Table 3: Host heterogeneity

Type	0	1	2	3	4
Heterogeneity (%)	-20	-10	0	+10	+20

capacity of the VMs is applied only if the reconfiguration is lower than 20% ($\rho_p > 0.8$). This threshold of 20% has been chosen to have a more complex virtual machines allocation trade-off while ensuring a fair CPU capacity.

5.1.2. Experiments setup

For each configuration size, ten instances of capacities of virtual machines and hosts have been generated, which means that: hosts types, virtual machines memory and CPU sizes and the number of instructions to execute are different between each instance. This leads to 50 different instances (10 per configuration). Also, the GA has been executed 10 times on each instance while the MILP, which has a deterministic behaviour, was executed once.

An initial MILP computes an upper bound of the energy and time objectives to normalize the weights. At each iteration the assignment MILP (phase 1) is the hardest to solve. The reconfiguration and frequency change MILP (phase 2) is much easier, which is due to the reduced number of variables as already mentioned. Then, the post-optimization (phase 3) is solved immediately. To limit the CPU time, a fixed time limit is set to the assignment MILP. Finally, we experimentally observed that for energy minimization phase (weight vector (1,0,0)) the min max part of the objective function ("fixed" energy part) yielded poor LP relaxations and so poor feasible solutions. Thus we only minimized the variable energy part, which corresponds to a min sum assignment cost and gives much better LP relaxations. The robustness is changed every $q = 10$ iterations and the number of switched on PM is decreased each time by $\nu = 12\%$. The MILP-based methods were coded in C++ using the IBM Cplex 12.5 MILP solver.

In order to keep a set of interesting solutions of these two approaches, only the "non-dominated" solutions are kept, recalling that a "non-dominated" solution is defined as follow:

Let X and Y be two solutions, X dominates Y if, for any criterion j to minimize, we have: $z^j(X) \leq z^j(Y)$, with at least one strict inequality. The set of non-dominated points in the objective space is called the Pareto frontier.

5.2. Results comparison

The first comparison proposed, in Section 5.2.1, analyzes the solution CPU time and the related optimization quality of the two heuristics. The aim is not to highlight that the GA is faster than the MILP but to analyse the evolution of the execution time and the quality of the solutions found by the two heuristics while varying some parameters. About the MILP-Decomposition, two parameters can influence the computation time: the *CPLEX TimeOut* at each iteration and the total number of iterations done (during the execution). Indeed, a very useful step has been done to calibrate the value of the *CPLEX TimeOut* in order to find the one which will give the best ratio between the total execution time and the quality of the solution obtained. Then another set of experiments has been carried out by varying the number of iterations, which also allows to find the most relevant values to use.

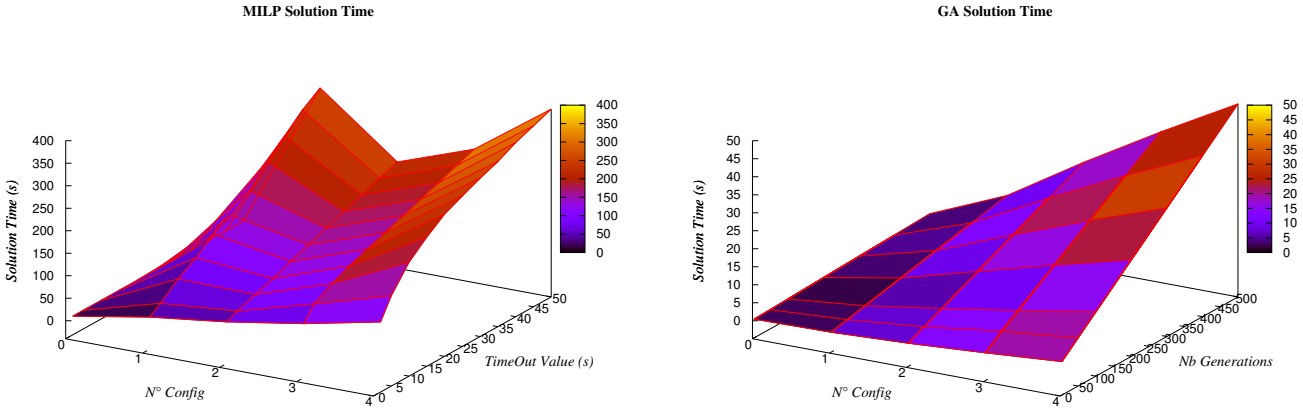
As a comparison metric, the normalized "Hypervolume Indicator" [33, 34] was used. This "Hypervolume Indicator" is a well known metric in the multiobjective optimization domain, which allows to make relevant comparison of the quality of the approximated Pareto frontiers found by several heuristics. The solution time study of the GA has been simply made by varying the number of generations.

The second comparison uses this hypervolume value to analyze the solutions' quality given by the two heuristics. The comparison of multiobjective results is not an easy task, especially when the different approaches which do not use the same normalization function and have also very different optimization behavior. Summaries of results obtained are shown in tables and figures of Section 5.2.2.

Finally, comparisons based on different lexicographic objective orders have been made. This allows to highlight other insight's characteristics of the two heuristics presented in this article. Also, the lexicographic order approach gives numerous results which have been exploited to compare the behaviour of the two heuristics on each metric separately.

5.2.1. Solution time and quality

The solution times of the two heuristics have been compared by varying the number of generations in the GA, and the *CPLEX TimeOut* for the MILP-Decomposition. Seven number of generations have been chosen for the GA $[5, 50, 100, 200, 300, 400, 500]$, and the values chosen for the the *CPLEX TimeOut* (in second) are the following $[2, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50]$. The evolution of the solution time of the heuristics in function of these sets of values is illustrated in Figures 4(a) and 4(b). The two volumes which compose this figure correspond to the execution times of the GA and the MILP (on instance 0). This figure has been generated with the values of all experiments while testing the whole range of *CPLEX TimeOut* values and also the seven number of generations values for the GA, for the five configuration problem sizes and on the 10 instances of each configuration. Figure 4(b) highlights the linearity of the GA w.r.t. to the number of the generation, and a more complex behavior for MILP-Decomposition. This is why, studies on the *CPLEX TimeOut* value and the number of iterations, presented below help to understand the MILP-Decomposition heuristic and its input values' calibration.



(a) 3D solution time illustration of the MILP-Decomposition

(b) 3D solution time illustration of the GA

Figure 4: Solution time comparison between the GA and the MILP-Decomposition

Table 4: TimeOut characteristics comparison

CPLEX TimeOut					
Number	1	2	3	4	5
Value (s)	2	5	10	15	20
Nb best	6	4	**8**	**8**	1
Hyp. avg	4781195	5220331	**5460928**	**5487397**	5458801

6	7	8	9	10
25	30	35	40	45
3	4	1	1	4
5459400	5470382	5487366	5480528	5482471

The analysis of the relation between the total solution time and the quality of the optimization, based on the hypervolume metric, obtained is illustrated in Figure 5 for the Genetic Algorithm and Figures 6 and 7 for the MILP-Decomposition. Each of these figures displays 10 curves that correspond to the 10 instances for the configuration of 100PM and 400VM.

For the GA, Figure 5 gives the hypervolume value in function of the number of generations. The hypervolume increases with number of generations ans shows a stagnation after about 500 iterations. Hence, this value has been chosen for further comparisons with the MILP-Decomposition reported in the two following sections. Indeed, this number of generation appears to be crucial to obtain a nice trade-off between the solution time and the optimization quality.

For the MILP-based method, Figures 6 also shows either an increasing hypervolume in function of the CPLEX time out value with a stagnation (instances 1,2,3,6,8,9) or a bell-shape (instances 0, 4, 5, 7).

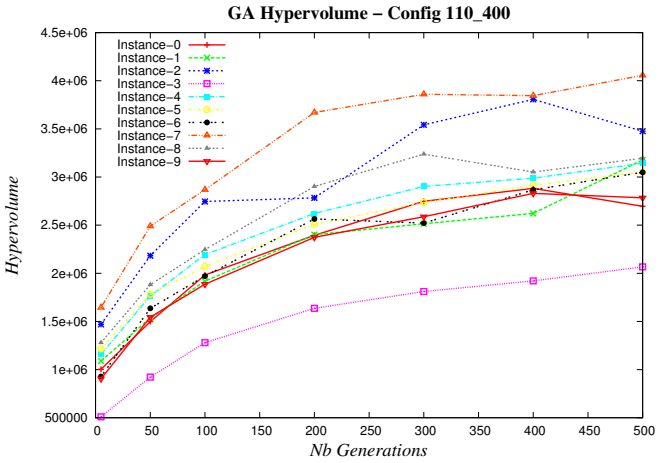


Figure 5: Influence of the number of generations of the GA

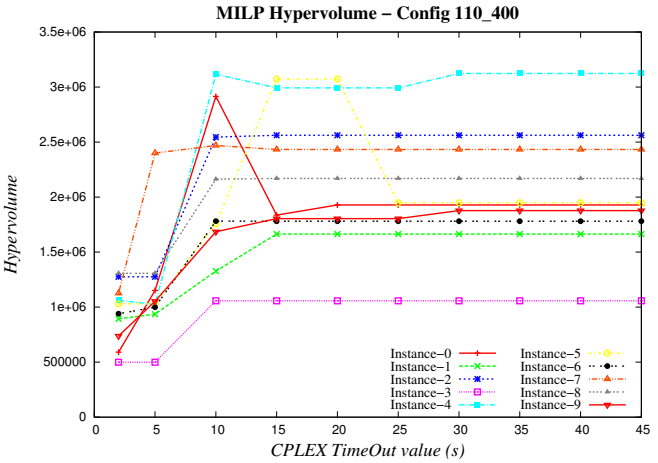


Figure 6: Influence of the *CPLEX TimeOut* (MILP-Decomposition)

The main conclusion of this phase is to see that the best *CPLEX TimeOut* to use is not the higher one, but the third or fourth one. This shows that it it is not necessary to over-optimize a subproblem at a given iteration. A precise analysis of this parameter has been done on each size of problem to determine the value which will give the best optimization in average on all configurations (excluding the smallest one 5_15 because each value gave the same result) and instances. This is summarized in Table 4. The hypervolume average and the number of times that each *CPLEX TimeOut* values gives the best value (for each configuration and each instances) are computed to find the best trade-off. For further computational experiments the 4^{th} value (10s) has been chosen.

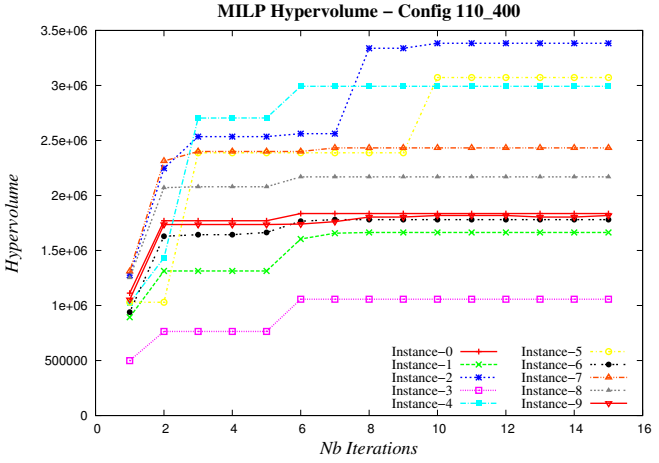


Figure 7: Influence of the number of iterations (MILP-Decomposition)

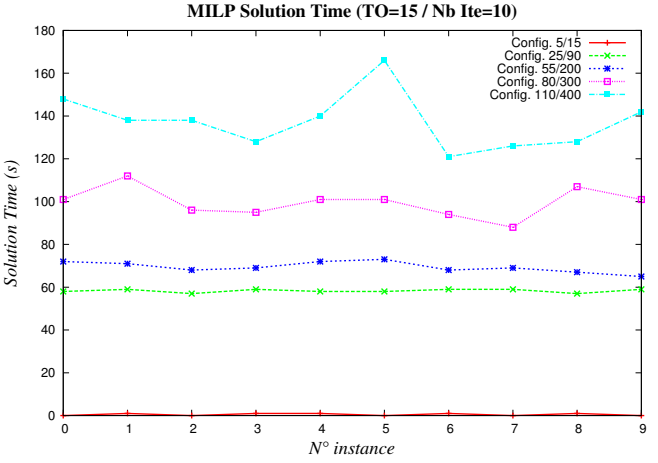


Figure 8: MILP-Decomposition solution time after calibration (on configuration 100_400)

Let us now consider the number of iterations done during the execution of the MILP-Decomposition. A set of 15 values ($\{1, \ldots, 15\}$) has been tested. Figure 7 displays the optimization results (with configuration 100_400) of these different number of iterations to illustrate how exactly this parameter influences the global optimization quality. This phase gives also useful information about how to calibrate the MILP-Decomposition. A stagnation of the optimization quality is clearly visible for each instance since the number of iterations is equal or superior to 10. Hence, to conduct the next computational experiments with the MILP-Decomposition, the number of iterations has been set to 10.

These calibration phases allowed to improve the ratio "optimization quality" versus "solution time" (i.e. lower solution time gives the same solutions quality). Figure 4(a) shows a maximum solving time lower than 400s for the higher problem size. However, after calibration, Figure 8 shows that the solving time for each instance of the five

configurations is always lower than 170s. Consequently, the calibration phases of the *CPLEX TimeOut* and the number of iterations have allowed to obtain an interesting performance/optimization trade-off.

5.2.2. Hypervolume analysis

This section contains two tables (Table 5 and 6) that give the hypervolume values obtained (the higher the value, the better the solution) with the MILP-based Decomposition and the GA, for the three following configurations: 5_15, 55_200 and 110_400. For each configuration, the ten different data instances have been evaluated. The hypervolume values do not have a real interest themselves, but the difference percentage between the two approaches allows to see that the GA is not always but most often better than the MILP-Decomposition approach on the hypervolume criterion. In some configuration instances the MILP-Decomposition can find a better optimization than the GA. This is indicated when the percentage value is lower than 0.

To temper the conclusion suggested by the hypervolume comparison, Table 6 gives the values of the four objective functions of the non dominated solutions found by each method on a particular instance. Although the MILP-Decomposition approach is less efficient in terms of execution time, the table shows that this approach is able to find very efficient solution in terms of response time. For this criterion, the MILP finds strictly better solutions than the GA on this instance, while the GA finds better values for the other objectives. In terms of results consistency the GA seems more stable with lower disparity between the found solutions.

A last question is to know how far the above-described heuristic algorithms are far from the optimal Pareto frontier. Limiting ourselves to the energy and response time criteria, we were able to obtain a lower bound of 112 for the weighted sum objective (Energy + Response time) with the MILP obtained by the linearization techniques described in Section 4.1. With these parameters the GA find a solution of 124,88 (125.55 for the MILP) which means that the gap is less than 12.1% from the optimum for both algorithms for this instance.

Table 5: Hypervolume comparision

Configuration	Instance	Hypervolume Value		Difference (%)
		GA	MILP	
5_15	0	30492332	29879118	2.01
	1	56567726	46304672	18.14
	2	37589617	37732627	-0.38
	3	48667527	42167724	13.35
	4	50178041	44025920	12.26
	5	52577203	46360942	11.82
	6	38015084	37306438	1.86
	7	40419177	35935952	11.09
	8	54010278	55713842	-3.15
	9	38993879	36769489	5.7
55_200	0	7997083	7465146	6.65
	1	8540858	6901692	19.19
	2	8630130	6317714	26.79
	3	8194239	7088897	13.48
	4	9254020	7793537	15.78
	5	8415340	7170061	14.79
	6	12389455	9526466	23.1
	7	8085961	7485145	7.43
	8	11293859	9097126	19.45
	9	11082994	8952591	19.22
110_400	0	2882016	1835744	36.3
	1	3177087	1664197	47.61
	2	3807004	2561813	32.7
	3	2067129	1056971	48.86
	4	3144132	2992586	4.81
	5	3051010	3071987	-0.68
	6	3048988	1780271	41.61
	7	4056095	2432604	40.02
	8	3238128	2168932	33.01
	9	2830638	1818614	35.75

As explained above, the use of the normalized hypervolume value allows to compare the global optimization quality of heuristics, but does not allow to highlight behavior's insight of the heuristics used on each criterion

Table 6: Objective values of Non-Dominated solutions (Config 110/400, Instance 0)

Metrics	Response Time (s)	Energy (Wh)	Robustness ⊘	Dynamism (Free MILPS)
MILP	130.8	591.8	3.6	392
	109	632.4	3.6	327
	109	624.7	3.6	285
GA	123.7	540.3	3.6	403
	129.7	547.3	3.6	424
	123.7	540.7	3.6	414
	122.5	540.8	3.6	414
	120.4	541.6	3.6	408

separately. About the optimization quality, the different analyzes of hypervolume results show the MILP can be close in average to the GA quality: 7.2% for the 5_15 configuration, 16.5% for the 55_200 configuration and about 31.9% for the biggest configuration. The fact that for some instances the two heuristics are very close and that the MILP-Decomposition is sometimes better than the GA, a further analysis which brings out the behaviour of these heuristics on each metric is needed.

This is why, the next section presents the result obtained using a lexicographic comparison.

5.2.3. Lexicographic analysis

The analyzes proposed in this section are based on a lexicographic order comparison. This first leads to number each metric : 1-Response Time, 2-Energy, 3-Robustness and 4-Dynamism. As the multi-criteria problem studied in this article takes into account four Cloud QoS parameters, the number of orders that can be analyzed is equal to $4! = 24$. The idea is to see if one of the two heuristics is better than the other on one or more criteria, and also, for example, if it is always on the same criteria. An example of results is given on Table 7. This table exposes the results on the 24 orders for the five problem sizes for the instance 0[2]. Each table contains 120 results (5×24), and a simple analysis allows to count how many time each heuristic is the best for each order on each problem size. For example, the result of the instance 5 which is shown through the Table 7 is the following: the MILP is 72 times the best, and the GA is 48 times the best. If the same counting is done on the other 9 tables (i.e. 9 other instances), the results can be sometimes quite different. For example, the instance 6 gives the MILP 48 times the best and 72 times for the GA. The instance 1 also gives a different result with the same score for the two heuristics (60 times the best for both).

Another analyzes have been done by taking into account the metric which had the higher importance for a given order. For the four QoS metrics of this article, it leads to determine which heuristics is the best for each order. This has been done by dividing each result table (for the 10 result tables, the orders are the same as in Table 7) every six lines to extract the results for each metric (1xxx, 2xxx, 3xxx and 4xxx). Thus, the computation for each metric is done with 30 result values (for one instance). With the ten instances, a result for a metric is computed with 300 values, and the whole analyzes are illustrated through histograms of Figure 9(a). Another result interpretation is given to show how the two heuristics behave compared to the different problem sizes. This analysis is illustrated in Figure 9(b). Finally, Figure 9(c) shows how many times the GA and the MILP are the best on each instance. This kind of analyzes highlights the importance of doing them on several set of experiments and also that the approach proposed with the decomposition of the MILP is indeed interesting and relevant.

This lexicographic analysis enables to draw more precise conclusions on the heuristic comparison compared with the single hypervolume comparison. This result clearly shows that the GA is better for the dynamism, the MILP-Decomposition is better for the response time, and the two heuristics are very close on the two other criteria (energy & robustness).

5.2.4. Radar Graph & Multiobjective comparisons

This section illustrates the differences between results obtained with the GA and the MILP-Decomposition approaches, through several radar graphs which allow to see the optimization behavior of each approach on the four QoS metrics chosen. These radars graphs are similar to a standard XY graph, except they allow to represent optimization quality of the four metrics. The response time is represented on the positive side of the X axis, the

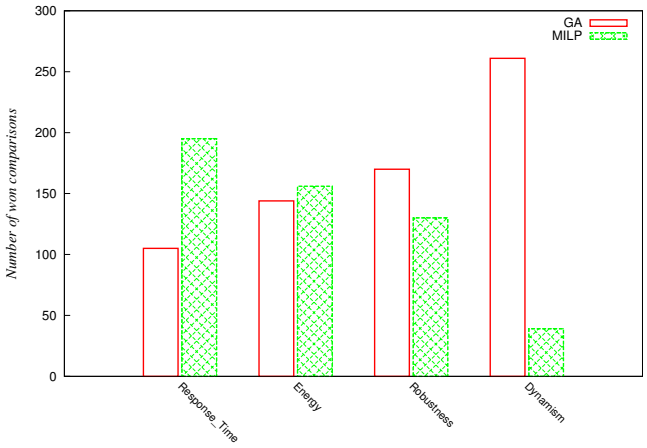
[2]The others 9 instances have also been analyzed but only one is shown in the article.

Table 7: Example of Lexicographic Order Comparaison

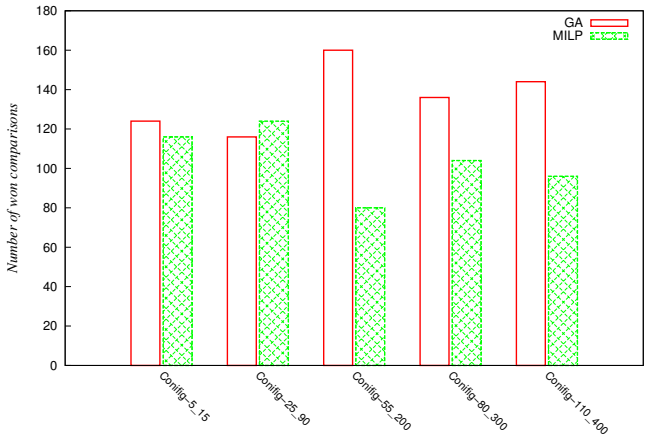
Order	**Instance 5**				
	5_15	**25_90**	**55_200**	**80_300**	**110_400**
1234	GA	GA	MILP	MILP	MILP
1243	GA	GA	MILP	MILP	MILP
1324	GA	GA	MILP	MILP	MILP
1342	GA	GA	MILP	MILP	MILP
1423	GA	GA	MILP	MILP	MILP
1432	GA	GA	MILP	MILP	MILP
2134	MILP	MILP	MILP	GA	MILP
2143	MILP	MILP	MILP	GA	MILP
2314	MILP	MILP	MILP	GA	MILP
2341	MILP	MILP	MILP	GA	MILP
2413	MILP	MILP	MILP	GA	MILP
2431	MILP	MILP	MILP	GA	MILP
3124	GA	GA	MILP	MILP	MILP
3142	GA	GA	MILP	MILP	MILP
3214	MILP	MILP	MILP	GA	MILP
3241	MILP	MILP	MILP	GA	MILP
3412	MILP	MILP	GA	GA	GA
3421	MILP	MILP	GA	GA	GA
4123	MILP	MILP	GA	GA	GA
4132	MILP	MILP	GA	GA	GA
4213	MILP	MILP	GA	GA	GA
4231	MILP	MILP	GA	GA	GA
4312	MILP	MILP	GA	GA	GA
4321	MILP	MILP	GA	GA	GA

energy on the positive side of the Y axis, the robustness on the negative side of the X axis and the dynamism on the negative side of the Y axis. In favor of the readability, each metric value has been standardized (using the minimum and the maximum of each instance for each configuration) leading to have values between 0 and 1. The solutions, of the two heuristics, that composed each radar graph are the Non-Dominated (ND) solutions obtained for each instance. Each solution is a quadrilateral having vertices on the graph axes. This last analysis phase yields of 50 radar graphs, with a number of ND solutions which varies between 3 and 13. In order to not have too much plans in each figure, the instances with the lowest number of ND solutions, of different configurations, have been chosen.

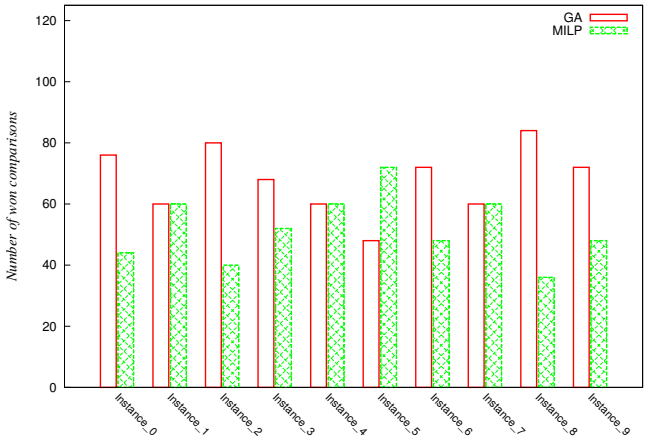
This last analysis allows to see more precisely how the results of an instance are composed. In each graph ND solutions of the GA and the MILP-Decomposition are shown. It is noteworthy that the GA gives a set of solutions (for one instance) which are very close in terms of optimization quality. Indeed the behavior of a GA is known to give good trade-off solutions, whether the number of generation is enough large, despite its intrinsic random process. The MILP-Decomposition gives solutions which can be more extreme (for the same instance) while being a good optimization trade-off. The plans (MILP0, MILP1 and MILP2) of Figure 10(c) well illustrate this observation. The solution MILP1 gives an interesting result for the response time, the solution MILP2 is the best for the energy, but they both do not give interesting result concerning the other objectives, respectively. The solution MILP0 is the one that gives the better trade-off with a robustness optimization very closed to 1, the best response time and the dynamism approximately equal to 0.6. Results of this biggest configuration problem (110_400), clearly show the difficulty for the MILP-Decomposition method to give solutions with a fairly trade-off between the four objectives. Indeed, the different ND solutions can be the best simultaneously up to three metrics while giving worst optimization for the fourth one. The MILP ND solutions of the configuration 55_200, shown on Figure 10(b), represent a mix of interesting trade-off between the four metrics, and solutions with the same optimization disparity as in Figure 10(c). The MILP0, MILP2 and MILP3 ND solutions are very good better than or equal to the GA on the energy, response-time and robustness. The two other MILP ND solutions do not give interesting trade-off, but are clearly the best on the energy objective. The last figure (Figure 10(a)) is composed of three GA ND solutions and two of the MILP-Decomposition. This kind of situation, on a small instance (here 5VM/15PM) allows to see that when the GA finds several non-optimal solution but very close to it, while the MILP is able to give solutions which are very close to the GA on the four objectives.



(a) Number of times GA or MILP are the best for each metric

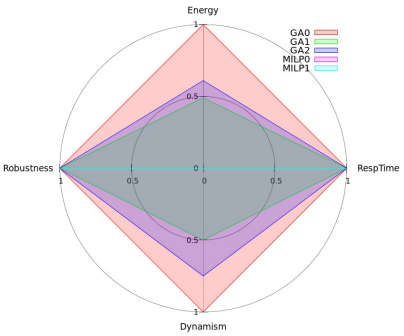


(b) Number of times GA or MILP are the best for each problem size

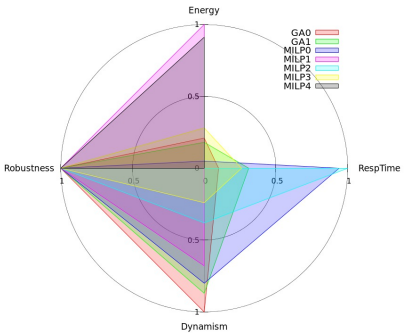


(c) Number of times GA or MILP are the best on each instance

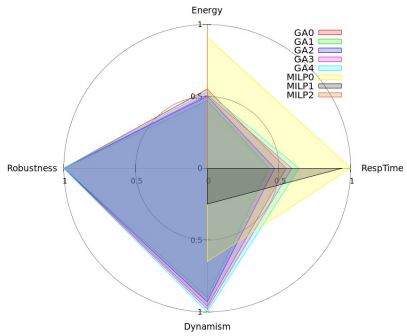
Figure 9: Histograms related to the lexicographic analysis



(a) ND solutions of configuration 5_15, instance 1



(b) ND solutions of configuration 55_200, instance 3



(c) ND solutions of configuration 110_400, instance 5

Figure 10: Illustrations of ND solutions through three radar graphs on instances 5_15, 55_200 and 110_400

Table 8 contains the results of a scoring process, of both heuristics, related to different set of coefficient weights applied to the metrics. The score given to each method has been computed using the real result values of the ND solution of the each instance. Then, the average (column AVG) of these scores are kept. Also the minimum and maximum scores are shown in the columns MIN and MAX, respectively. The computation of the score is done with standardized metric values (using the [min;max]) range of a whole configuration. The standardization applied leads to have values between 0 and 1 (1 is the best). Then, for a ND solution, the standardized metric

Table 8: Optimization average, min and max score : Config 110_400

Coeff Weights				GA			MILP		
RT	E	ROB	DYN	MIN	AVG	MAX	MIN	AVG	MAX
1	0	0	0	4.0	**56.5**	71.0	0	**97.8**	100.0
0	1	0	0	31.0	**56.7**	65.0	0	**40.5**	100.0
0	0	1	0	100.0	**100.0**	100.0	100.0	**100.0**	100.0
0	0	0	1	51.0	**78.9**	100.0	0	**65.8**	81.0
0.25	0.25	0.25	0.25	49.2	**70.6**	79.2	35.2	**61.7**	69.2
0.7	0.1	0.1	0.1	22.2	**62.1**	73.3	14.4	**83.2**	87.7
0.1	0.7	0.1	0.1	38.9	**61.8**	69.1	15.7	**45.0**	84.2
0.1	0.1	0.7	0.1	79.7	**88.2**	91.7	74.1	**84.7**	87.7
0.1	0.1	0.1	0.7	52.7	**73.0**	88.3	16.3	**58.4**	70.0
0.4	0.4	0.1	0.1	32.3	**61.4**	69.4	15.4	**58.5**	64.3
0.4	0.1	0.4	0.1	51.0	**75.1**	82.0	44.4	**83.9**	87.7
0.4	0.1	0.1	0.4	37.8	**66.7**	78.8	20.4	**64.1**	74.2
0.1	0.4	0.4	0.1	59.3	**74.9**	79.7	45.4	**64.1**	84.2
0.1	0.4	0.1	0.4	45.8	**66.6**	77.9	22.5	**49.3**	65.3
0.1	0.1	0.4	0.4	66.2	**80.1**	89.9	46.0	**68.8**	75.7
0.3	0.3	0.3	0.1	48.1	**70.4**	76.8	35.3	**67.1**	72.1
0.3	0.3	0.1	0.3	39.1	**64.7**	75.1	22.3	**54.1**	63.1
0.3	0.1	0.3	0.3	51.8	**73.8**	83.1	38.3	**70.9**	78.7
0.1	0.3	0.3	0.3	57.1	**73.8**	82.5	40.3	**59.5**	71.6

values are multiplied by the weights and finally multiply by 100 to obtain a score between 0 and 100 related to each weight combination. The more the score is closed to 100 the better. As we can see in the Table 8, 19 different weight combinations have been defined. The first four ones represents a mono-objective optimization and the fifth represents a fair multiobjective optimization. The values of the 14 other combinations have been chosen in order to represent different kinds of optimization which could represent the needs of a Cloud provider. Indeed, these combinations mix the four objectives with various weights. The last study lead to have five tables (related to the five configurations) but only one is displayed here.

5.2.5. MILP / GA pros and cons

The proposed extended results comparisons have been done in order to have a large and rich overview of the GA and MILP-Decomposition behaviors. One of the main challenges of this work was to propose a heuristic, based on a MILP approach, with the intent to obtain interesting result in terms of multiobjective optimization dedicated to a Cloud environment. It is well known that a genetic algorithm is very efficient in terms of convergence rates and solution times. The strength of a MILP-Decompostion is to be able to give optimal intermediate optimization. Despite the decomposition of the objectives optimization, the combination of these phases can have disadvantage compared to the GA for the above given reasons. As the problem studied in this article has complex non-linear objectives, it is understandable that the MILP-Decomposition is not good as the GA in terms of solution time. Moreover, it was quite tricky to anticipate which objective will be the most difficult to optimize. The elaboration of the MILP-Decomposition is the result of numerous experiments and improvements phases to increasingly raise a good overall optimization. There is a clear different behavior of the MILP-Decomposition and the GA in terms of the obtained trade-offs between the four objectives. In our setting, the MILP has the ability to optimize one objective individually (especially response time and energy) but may have difficulties in obtaining a smart trade-off solution. This allowed to better understand the role and the intrinsic properties of each Cloud QoS metric used.

6. Conclusions & Perspectives

This article proposed a Cloud quality of service multiobjective optimization, using two different optimization methods. As the considered Cloud virtual machine allocation problem has a huge complexity, even higher in this article due to the use of the DVFS and the virtual machine reconfiguration process, the exact Mixed-Integer Linear Program cannot be used on the tested instances. Thus, an alternative MILP-Based Decomposition, which iteratively solves sub-problems in order to give the best possible solution for the global problem, has been presented.

The comparison with the GA allowed to highlight weaknesses and strengths of both approaches. Also, the MILP-Decomposition has given interesting informations about the complexity to solve the different sub-problems and also about the optimization complexity of the Cloud QoS metrics analyzed.

Another aspect of the results analyzes has been the comparison of these two approaches, first using the "Hypervolume indicator" which allows to ignore how these methods optimize, normalize and compute their solutions. It provides an aggregate view of the quality of the solutions from a multiobjective aspect. Using this hypervolume value, another interesting phase study was to highlight how the heuristic proposed through the decomposition of the MILP has been improved with the calibration of two input parameters: the *CPLEX TimeOut* value, and the number of iterations executed. This study showed the impact of these two input parameters, and the importance to find the right values before comparing this MILP-Decomposition with the GA. It allowed to decrease the solution time of the MILP-Decomposition while keeping solutions quality very close or equal to the solutions obtained with a longer solution time. The GA most often dominates the MILP approach for the hypervolume criterion.

Then, the lexicographic order gave a very different point of view of the results obtained as it allows to compare the two heuristics through the optimization quality of each QoS objective. This comparison approach has been chosen to propose an alternative view of the whole the ten instances of each five problem sizes. The analysis of the 24 orders have allowed to show different characteristics of the two heuristics and mainly to highlight their behaviour and their global performance on each QoS metric (Figure 9(a) and Figure 9(c)). This allowed to draw the conclusion that MILP is better when priority is given to the response time, while the GA is better when priority is given to dynamism.

The last method used to analyze the results allows to have an overview of the whole Non-Dominated solutions of the two heuristics. Indeed, this analysis uses radar graphs, allowing to show the optimization quality for each metric, which was not possible only with the Hypervolume and the lexicographic comparisons. Finally, in order to be closer to the reality of QoS objectives trade-off that a Cloud provider would want to have, the last table proposes 19 different combinations of coefficient weights. Thus, it gives an overview of the optimization behavior of both heuristics, and also the correlation or the antagonism between the metrics. As explained in the introduction, one of the challenges of this article was to develop an heuristic based on a MILP and then to better understand how a multi-criteria approach, focusing on the optimization, could be interesting in a Cloud Computing context. The selected QoS objectives, the experiments and analyzes proposed in this article result on different conclusions and trade-off given by the GA and the MILP-Decomposition.

Future work concerns both applicative and physical areas. It includes the extension of the input parameters. This can concern the jobs running on the virtual machines which could have different quality of service needs. Moreover, each job can be associated to a type which defines its priority or whether a job can be delayed or stopped for a certain period of time. To implement these new features and also to be able to deal with larger size of instances, a detailed analysis of the scalability of both approaches will be performed. Another perspective is to take into account different kind of energy sources. Green and renewable energy sources could be considered, each with their own productivity rate. The management of these sources will lead to interesting researches with the intention of increasing the use of clean energy and improve the manner of how to use them as they could have a limited lifetime.

Acknowledgement

The authors are grateful to the referees for their helpful remarks, comments and references.

References

[1] M. Garca-Valls, T. Cucinotta, C. Lu, Challenges in real-time virtualization and predictable cloud computing, Journal of Systems Architecture 60 (2014) 726 – 740.

[2] H. M. Afsar, C. Artigues, E. Bourreau, S. Kedad-Sidhoum, Machine reassignment problem: the ROADEF/EURO challenge 2012, Annals of Operations Recearch 242 (2016) 1–17.

[3] F. Lopez-Pires, B. Baran, Virtual machine placement literature review, arXiv preprint arXiv:1506.01509 (2015).

[4] S. Islam, K. Lee, A. Fekete, A. Liu, How a consumer can measure elasticity for cloud platforms, in: Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering, ICPE '12, ACM, New York, NY, USA, 2012, pp. 85–96.

[5] Y. Sun, J. White, S. Eade, D. C. Schmidt, ROAR: A QoS-oriented modeling framework for automated cloud resource allocation and optimization, Journal of Systems and Software 116 (2016) 146 – 161.

[6] C. Papagianni, A. Leivadeas, S. Papavassiliou, V. Maglaris, C. Cervello-Pastor, A. Monje, On the optimal allocation of virtual resources in cloud computing networks, IEEE Transactions on Computers 62 (2013) 1060–1071.

[7] Z. Zhang, L. Cherkasova, B. T. Loo, Optimizing cost and performance trade-offs for mapreduce job processing in the cloud, in: IEEE Network Operations and Management Symposium (NOMS), 2014, pp. 1–8.

[8] M. Macas, J. Guitart, Analysis of a trust model for SLA negotiation and enforcement in cloud markets, Future Generation Computer Systems 55 (2016) 460 – 472.

[9] C. Esposito, M. Ficco, F. Palmieri, A. Castiglione, Smart cloud storage service selection based on fuzzy logic, theory of evidence and game theory, IEEE Transactions on Computers 65 (2015) 2348–2362.

[10] S. K. Garg, C. S. Yeo, A. Anandasivam, R. Buyya, Environment-conscious scheduling of HPC applications on distributed cloud-oriented data centers, Journal of Parallel and Distributed Computing, Special Issue on Cloud Computing 71 (2011) 732–49.

[11] H. Mi, H. Wang, G. Yin, Y. Zhou, D. Shi, L. Yuan, Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers, in: IEEE International Conference on Services Computing (SCC), 2010, pp. 514–521.

[12] A. Beloglazov, R. Buyya, Energy efficient resource management in virtualized cloud data centers, in: 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid), 2010, pp. 826–831.

[13] T. V. T. Duy, Y. Sato, Y. Inoguchi, Performance evaluation of a green scheduling algorithm for energy savings in cloud computing, in: IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010, pp. 1–8.

[14] S. Srikantaiah, A. Kansal, F. Zhao, Energy aware consolidation for cloud computing, in: Proceedings of the 2008 conference on Power aware computing and systems, volume 10, San Diego, California, 2008, pp. 1–5.

[15] J. A. Aroca, A. F. Anta, M. A. Mosteiro, C. Thraves, L. Wang, Power-efficient assignment of virtual machines to physical machines, Future Generation Computer Systems 54 (2016) 82–94.

[16] F. Yan, L. Cherkasova, Z. Zhang, E. Smirni, Optimizing power and performance trade-offs of mapreduce job processing with heterogeneous multi-core processors, in: IEEE 7th International Conference on Cloud Computing, 2014, pp. 240–247.

[17] L. Cherkasova, A. Zhang, Optimizing qos, performance, and power efficiency of backup services, in: Sustainable Internet and ICT for Sustainability (SustainIT), 2015, pp. 1–8.

[18] D. Borgetto, H. Casanova, G. D. Costa, J.-M. Pierson, Energy-aware service allocation, Future Generation Computer Systems 28 (2012) 769 – 779. Special Section: Energy efficiency in large-scale distributed systems.

[19] F. L. Pires, B. Barán, Multi-objective virtual machine placement with service level agreement: A memetic algorithm approach, in: IEEE/ACM 6th International Conference on Utility and Cloud Computing, 2013, pp. 203–210.

[20] M. Mezmaz, N. Melab, Y. Kessaci, Y. C. Lee, E.-G. Talbi, A. Y. Zomaya, D. Tuyttens, A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems, Journal of Parallel and Distributed Computing 71 (2011) 1497–1508.

[21] D. Borgetto, H. Casanova, G. Da Costa, J.-M. Pierson, Energy-aware service allocation, Future Generation Computer Systems 28 (2012) 769–779.

[22] H. S. Abdelsalam, K. Maly, R. Mukkamala, M. Zubair, D. Kaminsky, Analysis of energy efficiency in clouds, in: Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, COMPUTATIONWORLD'09, 2009, pp. 416–421.

[23] G. Dasgupta, A. Sharma, A. Verma, A. Neogi, R. Kothari, Workload management for power efficiency in virtualized data centers, Communications of the ACM 54 (2011) 131–141.

[24] Y. Ding, X. Qin, L. Liu, T. Wang, Energy efficient scheduling of virtual machines in cloud with deadline constraint, Future Generation Computer Systems 50 (2015) 62–74.

[25] A. Beloglazov, J. Abawajy, R. Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing, Future generation computer systems 28 (2012) 755–768.

[26] D. Whitley, S. Rana, R. B. Heckendorn, Island model genetic algorithms and linearly separable problems, in: Evolutionary computing, Springer, 1997, pp. 109–125.

[27] T. Guérout, S. Medjiah, G. D. Costa, T. Monteil, Quality of service modeling for green scheduling in clouds, Sustainable Computing: Informatics and Systems 4 (2014) 225–240.

[28] T. Guérout, T. Monteil, G. Da Costa, R. Neves Calheiros, R. Buyya, M. Alexandru, Energy-aware simulation with DVFS, Simulation Modelling Practice and Theory 39 (2013) 76–91.

[29] S. Rivoire, P. Ranganathan, C. Kozyrakis, A comparison of high-level full-system power models, in: Proceedings of the 2008 conference on Power aware computing and systems, HotPower'08, USENIX Association, 2008, pp. 3–3.

[30] M. Srinivas, L. Patnaik, Genetic algorithms: a survey, Computer 27 (1994) 17–26.

[31] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE transactions on evolutionary computation 6 (2002) 182–197.

[32] F. Cappello, E. Caron, M. Dayde, F. Desprez, Y. Jegou, P. Primet, E. Jeannot, S. Lanteri, J. Leduc, N. Melab, G. Mornet, R. Namyst, B. Quetier, O. Richard, Grid'5000: a large scale and highly reconfigurable grid experimental testbed, in: 6th IEEE/ACM International Workshop on Grid Computing, 2005. DOI: 10.1109/GRID.2005.1542730.

[33] C. M. Fonseca, L. Paquete, M. López-Ibánez, An improved dimension-sweep algorithm for the hypervolume indicator, in: IEEE International Conference on Evolutionary Computation, 2006, pp. 1157–1163.

[34] N. Beume, C. Fonseca, M. Lopez-Ibaez, L. Paquete, J. Vahrenhold, On the complexity of computing the hypervolume indicator, IEEE Transactions on Evolutionary Computation 13 (2009) 1075–1082.

Appendix - Summarized tables of parameters and decision variables

Table 9: PARAMETERS

Notation	Description
	SETS
P	Set of PM
V	Set of VM
F	Set of PM available frequencies
$\mathcal{T}$	Set of PM types
L	Discretized set of VM CPU reconfiguration values
$\overline{P}$	Fixed set of switched on PM (in the decomposition)
$\overline{V}(p)$	Fixed set of VM allocated to PM p (in the decomposition)
	PHYSICAL MACHINE
C_p	PM maximum CPU capacity
M_p	PM Memory capacity
π_f	PM CPU capacity decrease rate when assigned frequency f
$\xi(p) \in \mathcal{T}$	PM type
$P^f_{\xi(p),\min}$	PM Minimum power for a frequency f
$P^f_{\xi(p),\max}$	PM Maximum power for a frequency f
T_p	Upper bound on the PM response time
n_ξ	Number of type ξ PM
$\overline{n}_\xi$	Enforced switched on PM of type ξ (in the decomposition)
$\overline{\rho_p}$	Fixed reconfiguration rate for all VM on p (in the decomposition)
$\overline{f}_p$	PM fixed CPU frequency (in the decomposition)
$\overline{t}_p$	PM fixed response time (in the decomposition)
	VIRTUAL MACHINE
c_v	VM CPU capacity requirement
τ_v	VM's number of instructions to execute
r_v	VM CPU capacity
m_v	VM Memory capacity
R	Minimum decrease rate value (ρ_v)
$\alpha_l \in L$	l^{th} VM discretized CPU reconfiguration value
	OBJECTIVE WEIGHTS
w^{EN}	Energy objective weight
w^{TI}	Response Time objective weight
w^{RO}	Robustness objective weight
w^{DY}	Dynamism objective weight

Table 10: VARIABLES

Notation	Description
E	Total energy consumption
E_p	Energy consumption of PM p
t_{max}	Maximum response time
Rob	Robustness value
Dyn	Dynamism value
x_{vp}	Binary assignment variable of VM v to PM p
y_{pf}	Binary assignment variable of frequency f to PM p
ρ_v	Decrease rate applied on VM v
t_p	Response time of PM p
z_p	Binary switched on indicator of PM p
P_p	Power of PM p
R_{vp}	Percentage of CPU capacity used by VM v on PM p
t_v	Time needed to complete VM v
z_{pl}	Binary assignment variable of l^{th} reconfiguration to PM p
w_{pvl}	Binary assignment variable of l^{th} reconfiguration of VM v on PM p (product $z_{pl}x_{vp}$)
e_{pl}	Response time of PM p with the l^{th} reconfiguration (product $z_{pl}t_p$)
γ_{vpf}	Binary assignment variable of frequency f to PM p which runs VM v (product $x_{vp}y_{pf}$)
θ_{pf}	Response time of PM p with frequency f (product $y_{pf}t_p$)