MT-EA4Cloud: A Methodology for testing and optimising energy-aware cloud systems

Pablo C. Cañizares[1] , Alberto Núñez[1], Juan de Lara[2] and Luis Llana[1]

[1]*Dept. Sistemas Informáticos y Computación, Universidad Complutense de Madrid, Spain*

e-mail: pablocc@ucm.es, alberto.nunez@pdi.ucm.es, llana@ucm.es

[2]*Dept. Ingeniería Informática, Universidad Autónoma de Madrid, Spain*

e-mail: juan.delara@uam.es

Abstract

Currently, using conventional techniques for checking and optimising the energy consumption in cloud systems is unpractical, due to the massive computational resources required. An appropriate test suite focusing on the parts of the cloud to be tested must be efficiently synthesised and executed, while the correctness of the test results must be checked. Additionally, alternative cloud configurations that optimise the energetic consumption of the cloud must be generated and analysed accordingly, which is challenging.

To solve these issues we present MT-EA4Cloud, a formal approach to check the correctness – from an energy-aware point of view – of cloud systems and optimise their energy consumption. To make the checking of energy consumption practical, MT-EA4Cloud combines metamorphic testing, evolutionary algorithms and simulation. Metamorphic testing allows to formally model the underlying cloud infrastructure in the form of metamorphic relations. We use metamorphic testing to alleviate both the reliable test set problem, generating appropriate test suites focused on the features reflected in the metamorphic relations, and the oracle problem, using the metamorphic relations to check the generated results automatically. MT-EA4Cloud uses evolutionary algorithms to efficiently guide the search for optimising the energetic consumption of cloud systems, which can be calculated using different cloud simulators.

Keywords: Cloud modelling, Metamorphic testing, Simulation, Evolutionary algorithms, Energy-aware systems

1. Introduction

Cloud computing platforms are currently increasing their role to perform large-scale computational analysis [1, 2]. The ever-growing amount of online data requires the use of a large number of computational resources – like CPUs and storage devices – for its efficient processing. However, the speedup obtained by exploiting the parallelism offered by these resources requires an extremely high energetic cost [3]. As an example, according to the most recent survey (in November 2018) of the fastest 500 computers in the world, the top supercomputer Summit reaches a performance of 200,794.9 Teraflop/s with 2,397,824 cores [4]. This system requires 9,783 kW of power, a cost of 1,858€ per hour if we assume a cost of 0.19€ per kW.

In the last decade, IT companies have invested much effort to reduce energy costs. This has led to developing new approaches for improving the efficiency of energy consumption on cloud computing systems [5, 6]. However, testing these systems requires a high level of expertise, not only to design appropriate test suites but also to find a compromise between the overall performance and the efficiency to detect faults. Unfortunately, testing these approaches is costly, challenging and, in some cases, unfeasible.

Although testing is currently the most widely used technique to validate the correctness of computing systems, it presents some drawbacks when it is applied to cloud systems. First, the underlying architecture of the system under test is needed for checking its correctness. Moreover, the access to the cloud settings is usually restricted (i.e. allocation policy of virtual machines, network topology) which hampers the testing process. This is so as testing involves executing a large test suite over the system under test, which requires dedicated access to the system during a long period of time. Finally, providing an appropriate test suite (including the oracle) for determining the correctness of a system is complex, and particularly challenging if we target energy consumption. Hence, conventional testing techniques are not suitable to face the problem of checking the correctness and optimising energy consumption in cloud systems.

In order to alleviate these issues, we propose a formal methodology, called `MT-EA4Cloud`, to efficiently check the correctness of energy-aware cloud systems, and automatically propose improvements to their designs. `MT-EA4Cloud` is founded on a novel combination of metamorphic testing [7, 8], evolutionary algorithms [9] and simulation. `MT-EA4Cloud` requires the user to provide the cloud model under study, and hence it focuses on the testing and optimisation phases. This way, metamorphic testing is applied in the testing phase, EAs are applied to the optimisation phase and the energy consumption of each cloud model is calculated using different simulators.

Metamorphic testing (MT) is a testing technique that alleviates the oracle problem [7, 8] and the reliable test set problem [10]. Basically, MT models the properties of the system under test as *metamorphic relations* (MRs). The essential idea is that instead of checking the output o_1 produced when testing with one input x_1, we test with a second (follow-up) input x_2, observing output o_2, and check that o_1 and o_2 are related as specified by the MRs. `MT-EA4Cloud` uses MRs – which formally model the underlying cloud infrastructure – to automatically generate test cases focusing on the features reflected in these relations. Moreover, the MRs are applied to automatically check the correctness of the results provided by the execution of the test cases.

Evolutionary Algorithms (EAs) are inspired by the principle of natural selection and genetics [9]. The idea consists of simulating the evolution of different individuals, each one representing a potential solution to the target problem, which are evaluated using a fitness function. In this work, each individual represents a cloud configuration and the fitness function is designed by using a catalogue of MRs modelling the underlying behaviour of cloud systems regarding energy consumption. The main motivation to apply EAs for optimising the energy consumption of cloud systems is that EAs focus on an adaptive global search in the space of possible solutions and provide near-optimal solutions to complex and hard optimisation problems, where the execution time represents a significant constraint. Thus, `MT-EA4Cloud` applies EAs to efficiently generate alternative cloud configurations (individuals) to optimise the energetic consumption.

During the last years, simulation has been widely adopted by the research community as a cost-effective technique to model and analyse cloud computing systems [11, 12, 13]. Simulation presents several advantages in this domain: the access to the underlying architecture of the system under test is not required; experiments can be easily reproduced and parallelised, which significantly reduces the total execution time; and a wide spectrum of cloud configurations can be easily generated.

Overall, our work makes the following four contributions:

- Providing a methodology for deciding the most appropriate simulator to model and simulate cloud infrastructures. In general, each simulator focuses on modelling and simulating a specific part of the cloud like the storage system, virtual machine (VM) migration, resource provisioning and energy consumption, among others. The high number of existent cloud simulators makes it difficult to select an appropriate one for a specific purpose. In this work, we use MT to alleviate this issue. We accurately model the cloud in the form of MRs, which represent the underlying behaviour of the cloud. Thus, our methodology allows measuring the adequacy of each simulator for simulating cloud systems. This process is semi-automatic, in the sense that the test cases are automatically generated, executed and checked, but the MRs must be manually designed by the expert.

- Proposing a methodology for optimising the energy consumption of cloud infrastructures. One of the main difficulties in optimising cloud systems lies in the high complexity of constructing an accurate model of the system. In order to check and optimise the energy consumption of cloud systems, we propose a novel approach that combines MT and EAs. The idea is to use our EA to evolve cloud systems efficiently. This is achieved by using MRs, which guide the search for finding an optimised cloud. This way, each new offspring of individuals (clouds) are generated using the constraints defined in the MRs. Thus, the proposed EA not only reduces the search space but improves the overall efficiency.

- Providing an automatic method to execute the testing process. Optimising the energy consumption of cloud systems requires to generate an appropriate test suite for determining the correctness of the system under test (known as the reliable test set problem [10]) and to decide if the outputs of a test suite are correct (known as the oracle problem [14]). We face these challenges by combining MT and simulation. Our methodology allows to automatically generate quality test cases using the provided MRs and therefore the created test suite focuses on testing the features reflected in the MRs. Additionally, the outputs of the simulations are automatically checked using MRs. Thus, we provide an automatic method for optimising cloud systems without the intervention of a (human) oracle.

- Evaluating our proposed methodology to study its applicability to optimise cloud systems from an energy consumption point of view. We present a thorough study to optimise three different cloud systems using seven well-known cloud simulators. First, we use our methodology to check the adequacy of each simulator to analyse the cloud systems. Second, we use our novel approach that combines MT and EAs for optimising the energy consumption of the clouds. Finally, we show that our method outperforms an approach that uses MT and random testing for generating test cases.

The rest of the paper is structured as follows. Section 2 presents an overview of energy-aware cloud computing and introduces the main concepts of MT. Section 3 analyses related works. A detailed description of our methodology is presented in Section 4. In Section 5 we present a catalogue of MRs to analyse energy consumption in cloud systems. Section 6 shows our EA, and Section 7 presents tool support. Section 8 describes a thorough experimental study using our proposed methodology with different simulation tools, and the threats to its validity are discussed in Section 9. Finally, Section 10 finishes with the conclusions and prospects for future work.

2. Background

In this section we provide an introduction to some of the main concepts used in this work: energy-aware cloud computing, and metamorphic testing.

2.1. Energy-awareness in cloud computing

Currently, the energy consumption in cloud systems is gaining attention as one of the main important concerns in this field. In general, a high energy cost can be considered as a threat since it decreases the Return of Investment (ROI) and increases the Total Cost of Ownership (TCO) of the cloud infrastructures. The economic impact of running these systems cannot be neglected by its users, but moreover, the whole world is affected. The exponential increase of energy consumption has produced significant changes in the global environment, which has a negative environmental impact and in the well-being of the inhabitants of the world. On the one hand, the data centres – which in general are overprovisioned – constantly operate under their maximum capacity [15]. On the other, the developers of the applications that are executed on them generally do not take the energy consumption into consideration [16, 17]. In some cases, a single data centre is able to produce 170 million metric tons of carbon per year [18]. The carbon emissions of data centres worldwide are expected to reach 670 million metric tons by 2020 [19]. In particular, the average concentration of carbon dioxide in the atmosphere has increased from 280 ppm in the year 1750, to more than 400 ppm in 2018 [20].

Several factors motivate the interest in detecting and reducing the main causes of energy consumption, such as carbon footprint reduction [21], savings in IT electricity bills [22], and increase of the life time of some devices [23]. *Green computing* [24], or sustainable computing, has become the focus attention of initiatives such as Green Grid [25], a global consortium dedicated to advancing energy efficiency in data centres and business computing ecosystems. As demonstrated by the successful emergence of the Green500 list [26], which provides a ranking of the most energy-efficient supercomputers in the world, energy consumption has become as significant as performance.

2.2. Metamorphic testing

Conventional testing methods require checking whether the output(s) returned by the system under test are the expected ones. Schematically, let S be a system, I the input domain, $\mathcal{X}$ a test selection strategy and $\mathcal{T} = \{t_1, t_2, \ldots, t_n\} \subseteq I$ the set of tests generated using $\mathcal{X}$. When these tests are sequentially applied to the system S we obtain a sequence of outputs $S(t_1)$, $S(t_2)$, …, $S(t_n)$. Therefore, if we have an oracle, called f, to predict the expected output of S when exercised with any test in $\mathcal{T}$, then we find an error if there exists $t_i \in \mathcal{T}$ such that $S(t_i) \neq f(t_i)$.

In general, testing faces two fundamental problems. First, the *oracle problem* [14], which refers to the availability of a mechanism to distinguish between the correct behaviour and potentially incorrect behaviours of the system under test. Unfortunately, in some situations – like testing cloud systems – an oracle is not available or its application is computationally too expensive and alternative approaches must be used. The second issue is the *reliable test set problem* [10], which consists in providing an appropriate test suite for determining the correctness of a system. Since it is normally not feasible to execute all possible test cases over the system under test, a subset needs to be selected. However, selecting the most optimal subset is challenging.

Metamorphic testing, unlike the major part of the testing techniques, can be applied for both test case generation and test result verification, making it suitable to face both fundamental problems of testing [27, 28, 29]. The main difference between traditional testing techniques and MT

lies in the comparison of the obtained outputs. Hence, while traditional techniques compare the output of each individual test case with the one obtained from the oracle, MT checks the relation between multiple test inputs and their outputs.

MT uses expected properties of the target system, relating multiple test inputs with the corresponding outputs obtained from the system under test. These properties are formulated as metamorphic relations (MRs). An MR is a property of the analysed system that involves multiple inputs and their outputs. We represent an MR as a formula $\mathsf{i}(MR) \implies \mathsf{o}(MR)$, where $\mathsf{i}(MR)$ refers to the relation between the source test case and the follow-up test case, and $\mathsf{o}(MR)$ refers to the relation that must be fulfilled by the outputs obtained from the source test case and the follow-up test case. Generally, follow-up test cases are automatically generated by applying modifications over a source test case.

Let us illustrate this method with an example. Consider the problem of testing an implementation of the trigonometric *sine* function. Any implementation of this function will be an approximation, where the process for checking the correctness of an expected output for a given input can be complex and error-prone. However, we can define MRs encoding our knowledge of the domain, like the fact that $sine(x) = -sine(-x)$. Then, we provide a test input t, say 0.3, and call the function with it. Next, we calculate a follow-up test f that permits exercising the MR ($f = -0.3$). Finally, we check whether the two obtained outputs are related as expected by the MR ($sine(t) = -sine(f)$). If the output relation does not hold, a failure has been detected.

3. State of the art

In this section, we review works related to each technique used in our methodology: energy-aware cloud systems (Section 3.1), MT for cloud systems (Section 3.2), EAs in cloud design and operation (Section 3.3), and simulation of cloud systems (Section 3.4).

3.1. Energy-aware cloud systems

Current studies have shown that an idle data centre consumes around 70% of the power with respect to the same servers running at maximum CPU capacity [30]. Therefore, for energy efficiency reasons, it is necessary to devise techniques to hibernate idle nodes for reducing the overall consumption [31]. Next, we review some of them.

Faragardi and collaborators present an energy-aware resource provisioning mechanism to optimise the energy consumption in the data-centres supporting the cloud [32]. In particular, this approach serves real-time periodic tasks following the SaaS model. Based on an experimental study, the authors show that their approach outperforms an energy-aware version of RT-OpenStack.

Sayadnavard and collaborators propose an approach that takes into account the reliability of each physical machine (PM) to reduce the number of active PMs at the same time [33]. Then, a Markov chain model is designed to analyse the reliability of PMs, which are prioritised based on the CPU usage and the reliability status. The effectiveness of this work has been evaluated by performing an experimental study using the CloudSim toolkit.

Mohhamad and collaborators present a VM placement method based on the balance-based cultural algorithm for virtual machine placement (BCAVMP) to decrease the energy consumption in cloud data centres [34]. The proposed algorithm provides a novel fitness function to estimate VM allocation solutions. Haghighi and collaborators propose a virtualisation technique for resource management [35]. For this, the authors suggest a hybrid technique based on k-means

for mapping task and dynamic consolidation and a micro-genetic algorithm. The proposed technique provides a good trade-off between reducing the energy consumption and the quality of service of data centres. There exist several methodologies for energy-efficient computing and networking, which are focused on reducing the energy consumption of security protocols and frameworks [36]. Samy and collaborators propose secure energy-aware provisioning of cloud computing resources on consolidated and virtualised platforms [37]. This work is based on a dynamic round-robin provisioning mechanism, which powers down the subsystems of a host that are not required by the requested VMs. The experimental evaluation of these works has been conducted using the CloudSim simulator.

Kharchenko and collaborators propose a method to examine the energy efficiency of computational tasks in hybrid clouds, taking into account data privacy and security aspects [38]. For this purpose the authors examine a high computational demanding application by using mathematical models based on Markovian chains and queuing theory.

Overall, these approaches focus on saving energy in cloud systems by applying run-time techniques, like VM migration, server consolidation and VM placement. However, the goal of `MT-EA4Cloud` is to optimise the design of the cloud infrastructure for energy efficiency.

3.2. Metamorphic Testing for cloud systems

During the last years, MT [8, 7] has been successfully applied as an effective approach to alleviating the oracle problem to a wide variety of domains including web services [39] and embedded systems [40], among others. Remarkably, MT was able to detect new faults [41, 42] in three out of seven programs in the Siemens suite [43], which has been studied in major software testing research projects for 20 years. Similarly, Le and collaborators [44] discovered over one hundred faults in two popular C compilers (GCC and LLVM) using MT. Chan and collaborators [45] used MT to check both the functional behaviour and the energy consumption of wireless sensor networks. The authors of this work present two MRs for detecting failures, which are focused on data consolidation and equivalent consumption of sensor nodes that are close in proximity. Although MRs focusing on sensor networks must constantly be dealing with energy-saving issues, these cannot be applied to cloud computing systems. First, sensor networks are provided with limited batteries that considerably restrict the operations for computation and transferring information. On the contrary, the cloud uses computing and storage nodes that are provided with a constant power supply. Second, the cloud focuses on virtualising the hardware of computing nodes, allowing several users to share the resources of the same machine, which cannot be applied to sensor networks due to computing power limitations. Finally, the cloud is deployed using a predefined network topology, while the topology of the sensor networks may be built at run-time, allowing variations when the nodes have low battery.

Although there is currently work in the literature that combines MT with simulation techniques, to the best of our knowledge, MT has not been appropriately applied to check the correctness of energy consumption in cloud systems. Núñez and Hierons [46] combine the iCanCloud simulator with MT to detect unexpected behaviours when simulating cloud provisioning and usage. Although that contribution provides interesting ideas for checking the correctness of cloud systems, it also presents some limitations. That approach only provides a single MR for checking the correctness of the energy consumption. Moreover, a reduced number of test cases were applied during the testing process. Murphy and collaborators [47] present an approach to systematically test simulation software, specifically focusing on the domain of health care, with the aim of discovering defects in the implementation. Ding and collaborators [48] investigate the

effectiveness of MT to test a Monte Carlo modelling program for heterogeneous media. Additionally, they evaluate the adequacy of testing coverage criteria to measure the quality of the MT process, to guide the creation of MRs, in order to generate test inputs and investigate the exceptions found. Chen and collaborators propose the application of MT to check the conformance between network protocols and network simulators [49], and apply MT to discover faults in open queuing network models [50].

3.3. Evolutionary Algorithms in cloud design and operation

EAs have been successfully applied to solve real-world problems in a wide spectrum of fields, like swarm robotics, hardware design and fault tolerance and reconfigurability [51, 52]. In particular, several works applying EAs to cloud systems can be found in the literature as well. Keshanchi and collaborators proposed an improved genetic algorithm for static task scheduling for cloud environments [53], for assigning subtasks to processors. A profile-based approach was developed by Vasudevan and collaborators for energy-efficient application assignment to VMs with consideration of resource utilisation [54]. The approach is based on a Repairing Genetic Algorithm (RGA) to solve the large-scale optimisation problem. Xiao and collaborators [55] proposed a novel algorithm based on evolutionary game theory that successfully addresses the challenges faced by the dynamic placement of VMs. In this work, the authors demonstrate that the energetic consumption of a cloud is reduced by dynamically adjusting VM placement. A dynamic task scheduling algorithm that uses an Integer Linear Programming (ILP) model focusing on minimising the energy consumption in a cloud data centre was developed by Ibrahim and collaborators [56]. They also propose an Adaptive Genetic Algorithm (AGA) to reflect the dynamic nature of the cloud environment, which provides a near-optimal scheduling solution that minimises energy consumption.

Gabaldon et al. presented a multi-objective Genetic Algorithm based on a weighted blacklist for reducing the energy consumption and the makespan [57]. In this work, the proposal achieves better resource utilisation in comparison with the *first fit*, *best fit* and *worst fit* strategies. A VM allocation model based on collective intelligence was introduced by Zhang et al. [58]. This proposal considers the problem of redundant power consumption resulting from idle resource waste of physical machines. As a result, an optimal VMs placement scheme was designed based on feature metrics, multi-objective optimisation, and an EA algorithm. Joda et al. proposed an Energy-Efficient VM allocation technique using the Interior Search Algorithm (ISA) for reducing the energy consumption and resource underutilisation of a data centre [59]. A Particle Swarm Optimisation heuristic focused on finding the optimal VM placement to reduce power waste was presented by Abdessamia et al. [60]. Soltanshahi et al. [61] presented a solution for the VMs placement to reduce data centre energy using the Krill Herd algorithm, which is considered as the fastest collective intelligence algorithm. The experimental phase to analyse the performance of the proposal was conducted using the CloudSim simulator, achieving a reduction of 35% of the energy consumption.

These works use EAs to optimise cloud systems but focus on VM allocation and scheduling tasks. Our approach uses MRs – previously designed by an expert – to adapt the search of an optimised cloud configuration using an EA. In particular, we provide the design of cloud infrastructure, including its hardware architecture, to optimise the energetic consumption of the cloud. To the best of our knowledge, there are only a few works in the literature combining EAs and MT. Segura and collaborators presented a proof of concept to automate the detection of performance bugs by combining MT and search-based techniques [62]. Rounds and Kanewala identified 17 MRs for testing a GA and show, through MT, that these relations are more effective

at finding defects than traditional unit tests based on known outputs [63]. Arora and Bassi [64] show that using GAs increases the efficiency of MT to detect faults in software. However, these works focus on applying MT to check and test EAs, while we focus on the evaluation of the energy consumption of cloud systems.

The current literature reports different techniques based on EAs, such as Genetic Algorithms (GAs), Genetic Programming (GP), Ant Colony Optimisation (ACO), Cat Swarm Optimisation (CSO), Particle Swarm Optimisation (PSO) and Honey-Bees Mating Optimisation (HBMO), among others. Despite the diversity, all the EA variants are based on a common working scheme, where the performance and accuracy obtained strongly depend on the type and the complexity of the problem [65, 66]. Thus, the task of selecting an EA technique to solve a computational problem is vital to successfully design a valid solution.

In our proposal, we need to model complex systems that require a high number of inter-related parameters and therefore, we need flexibility to generate a wide spectrum of cloud configurations (individuals), and high performance to evaluate them efficiently. Techniques inspired by ACO, CSO, PSO and HBMO are based on swarm intelligence, where the focus is on cooperation. In essence, swarm intelligence focuses on systems containing many individuals, which coordinate using decentralised control and self-organisation. Since our individuals must compete to obtain the best result (energy consumption), we discarded these approaches. On the contrary, GAs promote competition, where the individuals more adapted to the environment propagate their *genetic information* to the next generation. Thus, the best individuals of each generation have a higher probability of being selected for reproduction. GAs have been used in the past to solve problems related to the cloud [67, 68]. Moreover, we think that mutation and crossover techniques are suitable to face the problem of optimising energy consumption in cloud systems. For this, we propose a hybrid encoding – combining GAs and GPs – based on graphs and integer representation that eases the processing of the large structures that conform the cloud.

3.4. Simulation of cloud systems

The research community has developed a vast collection of tools for modelling and simulation of cloud systems. However, only a small subset of them focuses on analysing the energy consumption of the cloud [69]. This set includes, among others, CloudSim [11], DCSim [70], GreenCloud [71], SimGrid [13], iCanCloud [72, 12] and DISSECT-CF [73].

CloudSim is an extensible and open-source Java simulator, which enables modelling cloud computing systems and application provisioning environments. CloudSim is considered the *de facto* standard cloud simulation platform due to its capabilities for simulating cloud systems, such as VM allocation and provisioning, energy consumption, federated clouds and the possibility to model different types of clouds like public, private, hybrid and multi-cloud environments. One of the key features of CloudSim is the possibility to include new functionalities using extensions like *cloudSimStorage*, which supports modelling the energy consumption of the storage system [74].

DCSim, also known as *The Data Centre Simulator*, is a Java extensible simulation framework for simulating a data centre hosting. In essence, DCSim focuses on the IaaS layer for providing services to multiple tenants.

GreenCloud is an open-source tool for simulating data centres focusing on data communication and energy cost in cloud computing. GreenCloud provides a wide range of network and communication configurations for simulating data centres.

SimGrid is a tool for simulating algorithms and distributed applications in distributed computing platforms. The resources are modelled by their latency and service rate, and the topology

is configurable by the users. Initially, SimGrid targeted grid environments. However, the current version of SimGrid supports a variety of cloud computing use cases including multi-purpose network representation, VM abstraction, live migration, VM support and storage.

iCanCloud is a simulation platform aimed to model and simulate cloud computing systems by providing different functionalities like resource provisioning. Additionally, the framework *E-mc*^{2} [12] can be used for analysing energy consumption. The main goal of iCanCloud is to predict the trade-offs between cost and performance of a given set of applications executed in specific hardware.

DISSECT-CF is a simulator targeted to evaluate the energy consumption of IaaSs. DISSECT-CF offers two major benefits: a unified resource-sharing model, and a complete IaaS stack simulation, which includes VM image repositories, storage and in-data-centre networking.

In general, the current approaches for modelling and simulating cloud systems are suitable to represent the behaviour of cloud architectures. However, each simulation tool focuses on a specific part of the cloud (e.g. storage system, VMs allocation policies, energy consumption) and, unfortunately, there is no common solution that satisfies the entire research community. Moreover, these tools lack a formal approach to represent cloud systems, which complicates the automation of the testing process. Our approach aims at alleviating these issues. First, using different simulation tools increases the features of the cloud infrastructure that can be modelled and simulated. Second, combining MT and simulation allows to formally model the main features of the cloud and, therefore, automating the testing process. Finally, our methodology can propose cloud optimisations from the energy point of view, which is not supported by these tools.

In relation to energy consumption, the previous simulators are not focused on modelling and simulating the cooling system, but are more focused on the global cloud infrastructure (e.g. network, virtualisation, storage), where the cooling system is not taken into account. However, there are several approaches [75, 76] to model the cooling system using some of these cloud simulators. Unfortunately, their source code and binaries are not available. Consequently, the cooling system must be computed separately using a specific simulator for this purpose. For instance, CoolSim [77] provides models to manage and design the airflow in data centres. DCWorms [78] models the cooling system of distributed systems, but it has not been designed to represent the underlying behaviour of the cloud.

4. Methodology

This section describes our proposed methodology, called `MT-EA4Cloud`, which combines MT, simulation and EAs to check the correctness of energy-aware cloud systems. The main steps of `MT-EA4Cloud` are depicted in Figure 1.

Initially, the features having a relevant impact on the energy consumption must be carefully analysed, like the computing system, the storage system, network features and the workload to be processed, among others. Next, these features are used to design the MRs (label ① in the Figure). The idea is to provide a formal and accurate model – in the form of MRs – that represents the underlying behaviour of the cloud. The set containing the provided MRs, which we refer to as *catalogue*, is denoted by C and will be discussed in detail in Section 5.

Our methodology does not target a specific tool, moreover, it is desirable to use several simulators in the testing process. To that end, the tester must choose the simulators that offer capabilities to model and simulate the features formulated in C (label ②). In this step, the simulators are not executed, but their specifications are analysed to determine whether they can be used in the testing process. The chosen set of simulators in this step is denoted by $\mathcal{S}$.

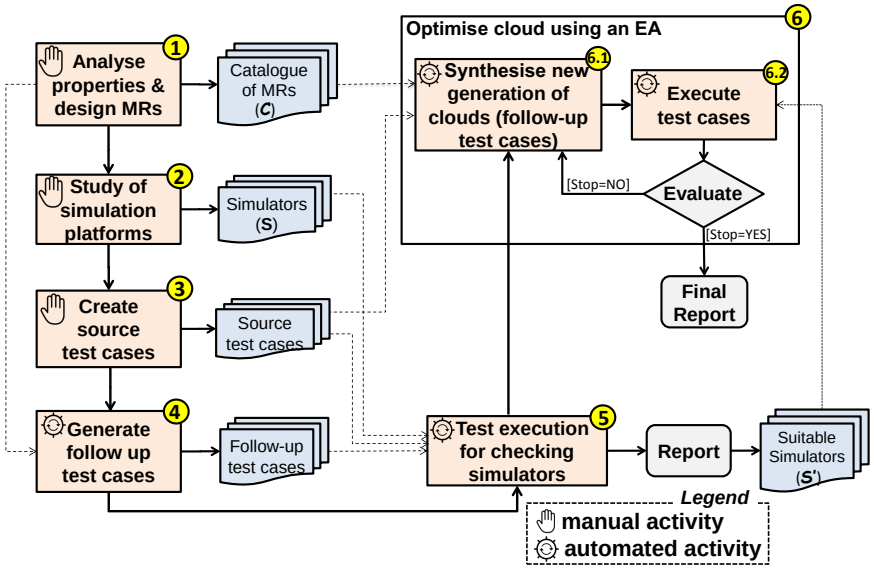


Figure 1: Scheme of the `MT-EA4Cloud` methodology.

We define a test case as a tuple (m, ω), where m refers to a cloud model and ω is the workload to be executed over m. The cloud model contains details about the underlying architecture of the system, while the workload represents the operations performed by the cloud. Similarly, a follow-up test case is denoted by (m', ω'). Executing a workload ω over a cloud model m is carried out by simulation. We denote by $S(m, \omega)$ the result of the simulation – using the simulator S – for executing the workload ω over m. In Section 5 we formally define when a test case $t = (m, \omega)$ and a follow-up test case $f = (m', \omega')$ satisfy an MR in a simulation performed by simulator S, which is denoted by $(t, f, S) \models MR$.

Next, in step ③, the tester manually designs a reduced number of source test cases. This set is called $\mathcal{T}$. The main difference between a source test case and a follow-up test case lies in the way the test case is generated. While a source test case is manually designed, a follow-up test case is automatically generated by using a source test case and an MR. In step ④, we apply a procedure, described in Section 5.3, to generate a set $\mathcal{F}$ of follow-up test cases.

The goal of the next step (label ⑤) is two-fold: first, to analyse the adequacy of each MR; second, to investigate how appropriate is each simulator to represent the behaviour of cloud systems, focusing on energy consumption. For this purpose, the source and the follow-up test cases are executed on the simulators chosen in step ②. We measure the adequacy of an $MR \in C$ by calculating the percentage of follow-up test cases f, generated from each source test case $t \in \mathcal{T}$ and executed using the simulator S, that fulfil $(t, f, S) \models MR$. The adequacy of a metamorphic relation MR using a simulator S and the test selection strategy $\mathcal{T}$, written $adq_{\mathcal{T}}(MR, S)$, is a number between 0 and 1 calculated as follows:

$$adq_{\mathcal{T}}(MR, S) = \frac{\sum_{t \in \mathcal{T}} |\{(t, f, S) \mid f \in followUp(t) \ \wedge \ (t, f, S) \models MR\}|}{\sum_{t \in \mathcal{T}} |\{(t, f, S) \mid f \in followUp(t)\}|} \quad (1)$$

where $followUp(t)$ is the set of generated follow-up test cases for t. Hence, `MT-EA4Cloud` can be applied to compare different simulators for representing the behaviour of cloud systems focusing

on energy consumption.

Once all the tests are executed over the simulators, a report containing the adequacy of the MRs is generated. The tester uses this report to create a new list of simulators, discarding those that do not appropriately represent the behaviour of cloud systems. As a result, a new list of simulators, denoted by $\mathcal{S}'$, is generated.

The quality of the cloud designs – focusing on energy consumption – that were created by the tester in step ③, is optimised in the last step (label ⑥). In order to accomplish this task, we use an EA, which is discussed in detail in Section 6. The EA evolves an initial population of cloud models generated from the source model, which is provided by the user. Such evolution involves applying different operations to create a new generation of clouds, until one of the individuals fulfils the stop criteria (e.g. its energetic consumption has been reduced a 5%).

5. Metamorphic relations for modelling energy-aware cloud systems

In this section, we provide a catalogue of novel MRs that are specially designed to analyse the proper use of energy in cloud architectures. For this purpose, we first introduce some notation (Section 5.1), then we present the catalogue (Section 5.2) and finish by describing how to generate follow-up test cases using the MRs (Section 5.3).

5.1. Notation

Clouds. One of the main challenges in using MRs for accurately analysing complex systems [7], like clouds, lies in appropriately representing – with enough fidelity – the high number of interrelated parameters that determine the behaviour of the system under test, such as, in the case of cloud systems, data centres containing a large number of physical machines, communication networks, concurrent access of different users and virtualisation, among others. We use the parameters provided by each simulator to model and estimate these systems, like the network latency or the CPU speed. Hence, the level of accuracy obtained directly depends on how each simulator processes these parameters to simulate the underlying behaviour of each subsystem. In order to provide a flexible and accurate configuration of the cloud, we use the following notation:

- We represent a processor *cpu* as a pair (s, n) where $s \in \mathbb{N}$ is the speed of *cpu* measured in MFlops and $n \in \mathbb{N}$ is the number of cores.
- A hard disk *hd* is a tuple (s, r, w) where $s \in \mathbb{N}$ denotes the total size in GBytes and $r, w \in \mathbb{N}$ are the read and write bandwidth of the disk, measured in Mbps, respectively.
- The RAM memory *mem* is a tuple (s, r, w) where $s \in \mathbb{N}$ is the total size measured in MBytes and $r, w \in \mathbb{N}$ are the read and write bandwidth in MBps, respectively.
- A *node* is a tuple (cpu, hd, mem) where *cpu* is its processor, *hd* is its hard disk, and *mem* is its RAM memory.
- A *board node* is a tuple $(Id, nodes = \{n_j\}_{j \in J})$ where $Id \in \mathbb{N}$ is its identifier, and *nodes* is the set of the actual nodes.
- A *rack* is a tuple $(Id, boards = \{b_i\}_{i \in I})$ where $Id \in \mathbb{N}$ is its identifier and *boards* is a set of actual board nodes.

- A network connection *net* is a tuple (src, dst, bw, lat) where $src, dst \in \mathbb{N}$ are the identifiers of the source and destination racks of *net*, $bw \in \mathbb{N}$ is the bandwidth of *net* measured in MBps and $lat \in \mathbb{N}$ is the latency measured in μs.
- We define a *cloud model* as a connected non-directed graph $m = (RS, C)$ where RS is the set of racks and C is the set of network connections.

Given a board node b, we use $b.nodes$ for its set of nodes, and similarly for other tuples.

Cloud metrics. To measure energy consumption, we define the following metrics:

- $\Delta(m_{cpu})$ denotes the overall CPU performance of the cloud model m. Formally:

$$\Delta(m_{cpu}) = \sum_{r \in m.RS} \sum_{b_i \in r.boards} \sum_{n_j \in b_i.nodes} n_j.cpu.s \cdot n_j.cpu.n$$

- $\Delta(m_{IO})$ is the overall I/O performance of the cloud m. Formally $\Delta(m_{IO}) = (r, w)$ where:

$$r = \sum_{r \in m.RS} \sum_{b_i \in r.boards} \sum_{n_j \in b_i.nodes} n_j.hd.r$$
$$w = \sum_{r \in m.RS} \sum_{b_i \in r.boards} \sum_{n_j \in b_i.nodes} n_j.hd.w$$

- $\Delta(m_{NET})$ denotes the overall network performance of the cloud model m. Formally:

$$\Delta(m_{NET}) = \frac{1}{|C|} \cdot \sum_{c \in m.C} c.bw$$

- $|m|$ denotes the number of physical machines contained in the cloud model m. Formally:

$$|m| = \sum_{r \in m.RS} \sum_{b_i \in r.boards} |b_i.nodes|$$

- $|vm|$ denotes the number of virtual machines contained in the cloud model m.

Workload. A workload ω is a trace of operations to be executed on the cloud system, that is, requests of VMs to be deployed in physical machines, storage and computing operations. The trace elements are taken from OP, the set of basic operations, and so $\omega \in \mathsf{OP}^*$. We say that $\omega \subseteq \omega'$ if ω is a subtrace of ω', and define similarly a strict subtrace $\omega \subset \omega'$.

Test case. As mentioned in Section 4, a test case is a tuple (m, ω) made of a cloud m and a workload ω, while a follow-up test case is denoted by (m', ω'). The execution of ω over m is performed by simulation, and the result of the simulation is denoted by $S(m, \omega)$. We are especially interested in energy consumption, and hence write $\Omega(S(m, \omega))$ to represent the overall energy consumption required to execute the workload ω over m, which is calculated using the simulator S. We can assume – without losing generality – that this is an integer number because power measures like $12.345W$ can be seen as $12345mW$ in a smaller power unit.

Metamorphic relation. Intuitively, an MR can be seen as a tuple (MR_i, MR_o), where MR_i is a relation over the source test case and a follow-up test case, and MR_o is a relation over the results obtained from the execution of these test cases. These test cases must fulfil the input relation MR_i. This way, a *metamorphic* relation MR_S for m and ω using S, can be formally represented as a set of 4-tuples:

$$MR_S = \Big\{ \langle (m,\omega),(m',\omega'),S(m,\omega),S(m',\omega')\rangle \mid MR_i((m,\omega),(m',\omega')) \Rightarrow MR_o(S(m,\omega),S(m',\omega'))\Big\} \tag{2}$$

5.2. Catalogue of Metamorphic Relations

In order to accurately model the underlying cloud infrastructure, an expert with deep knowledge in cloud systems must define a catalogue of MRs. The catalogue we propose in this work is depicted in Figure 2. Next, we discuss its intuitive meaning, where the parameters not mentioned in the explanation of the rules remains the same.

$$\begin{array}{ll}
\textbf{MR}_1 & \Delta(m_{cpu}) > \Delta(m'_{cpu}) \wedge \omega = \omega' \implies \Omega(S(m,\omega)) \leq \Omega(S(m',\omega')) \\
\textbf{MR}_2 & |m| \geq |m'| \wedge \omega = \omega' \implies \frac{|m|}{|m'|} \geq \frac{\Omega(S(m,\omega))}{\Omega(S(m',\omega'))} \\
\textbf{MR}_3 & \frac{|m|}{|m'|} \geq \frac{\Delta(m_{cpu})}{\Delta(m'_{cpu})} \wedge \omega = \omega' \implies \frac{|m|}{|m'|} \geq \frac{\Omega(S(m,\omega))}{\Omega(S(m',\omega'))} \\
\textbf{MR}_4 & \Delta(m_{IO}) > \Delta(m'_{IO}) \wedge \omega = \omega' \implies \Omega(S(m,\omega)) \leq \Omega(S(m',\omega')) \\
\textbf{MR}_5 & \Delta(m_{NET}) > \Delta(m'_{NET}) \wedge \omega = \omega' \implies \Omega(S(m,\omega)) \leq \Omega(S(m',\omega')) \\
\textbf{MR}_6 & \Delta(m_{RAM}) > \Delta(m'_{RAM}) \wedge \omega = \omega' \implies \Omega(S(m,\omega)) \leq \Omega(S(m',\omega')) \\
\textbf{MR}_7 & |m| = |m'| \wedge |vm| > |vm'| \wedge \omega = \omega' \implies \Omega(S(m,\omega)) \geq \Omega(S(m',\omega')) \\
\textbf{MR}_8 & m = m' \wedge \omega \subseteq \omega' \implies \Omega(S(m,\omega)) \leq \Omega(S(m',\omega'))
\end{array}$$

Figure 2: Catalogue of MRs for checking the correctness of cloud simulators.

MR$_1$: If the CPU of m has better performance than the CPU of m', and ω and ω' are equal, then the amount of energy required to execute ω over m should be less than or equal to the one required to execute ω' over m'.
MR$_2$: If the model m contains more machines than m', and ω and ω' are equal, then the ratio between the number of machines of m and m' should be greater than or equal to the ratio between the energy consumption required to execute ω over m and the one required to execute ω over m'.
MR$_3$: If the ratio between the number of machines of m and m' is greater than or equal to the ratio between the CPU performance of m and the CPU performance of m', and ω and ω' are equal, then the ratio between the energy consumption required to execute ω over m and the one required to execute ω' over m' should be less than or equal to the ratio between the number of machines of m and m'.
MR$_4$: If the I/O performance of m is better than the I/O performance of m', and ω and ω' are equal, then the energy consumption required to execute ω over m should be less than or equal to the one required to execute ω' over m'.
MR$_5$: If the network performance of m is better than the network performance of m', and ω and ω' are equal, then the energy consumption required to execute ω over m should be less than or equal to the one required to execute ω' over m'.

MR$_6$: If the RAM memory performance of m is better than the RAM memory performance of m', and ω and ω' are equal, then the energy consumption required to execute ω over m should be less than or equal to the one required to execute ω' over m'.
MR$_7$: If the number of machines used in m and m' is equal, the number of virtual machines deployed in m is greater than the number of virtual machines deployed in m', and ω and ω' are equal, then the energy consumption required to execute ω over m should be greater than or equal to the one required to execute ω' over m'.
MR$_8$: If m and m' are equal and the workload ω is a subtrace of ω', then the energy required to execute ω over m should be less than or equal to the one required to execute ω' over m'.

Please note that we did not include any MR dealing with the cooling system. This is so as the considered set of cloud simulators do not consider this system. However, as explained in Section 4, both the catalogue and the set of simulators is open, and so other MRs affecting the energy consumption can be added in future work.

5.3. Generation of follow-up test cases

MT can be used to alleviate the *reliable test set problem* [10]. First, the follow-up test cases are generated using a source test case as basis. Source test cases are manually created by the tester and, therefore, these test cases should be designed to test specific features of the system under test. Second, since generated tests should fulfil previously defined constraints in the form of MRs, these are focused on testing *sensible* parts of the system for providing relevant information.

Given a set of source test cases, we automatically generate the follow-up test cases, by copying the source test case and applying a slight modification in the replica. The key to automatically generate appropriate follow-up test cases is to calculate the cloud parameters that are most likely to be selected to perform the modifications. For this, we use the provided catalogue of MRs.

In our approach, a small number of source test cases must be provided by the tester. Additionally, the tester must select one or several MRs and the number of follow-up test cases to be generated. Since each generated test case must fulfil the selected MRs, we focus on those parameters that are used in the MRs (i.e. CPU parameters when MR_1 is involved). Next, we generate a random value within a specific range, to be assigned to the selected parameter. This way, we avoid generating follow-up test cases with unreal values like e.g. a communication network with a bandwidth of 1Mbps, or a disk drive with a capacity of 5 MB.

6. An Evolutionary Algorithm to optimise energy-aware cloud systems

Cloud computing systems usually consist of thousands of components and therefore, making random modifications on its underlying architecture, like CPU, memory and network, hampers the search for optimal models. In order to alleviate this issue, we propose an EA to explore complex search spaces efficiently. In particular, the EA will evolve the cloud systems using the catalogue of MRs to conduct the search towards the optimisation of the energetic consumption. For this purpose, it is necessary to provide a precisely defined cloud configuration with the main features of the system, as we defined in Section 5.1. The main steps of our proposed EA are depicted in Figure 3, and are explained next.

Initialisation. The algorithm requires as input a set of MRs modelling the underlying behaviour of the cloud, a cloud model and the maximum size of the population. Each individual in the population represents a cloud configuration. In this phase, the tester designs manually a cloud

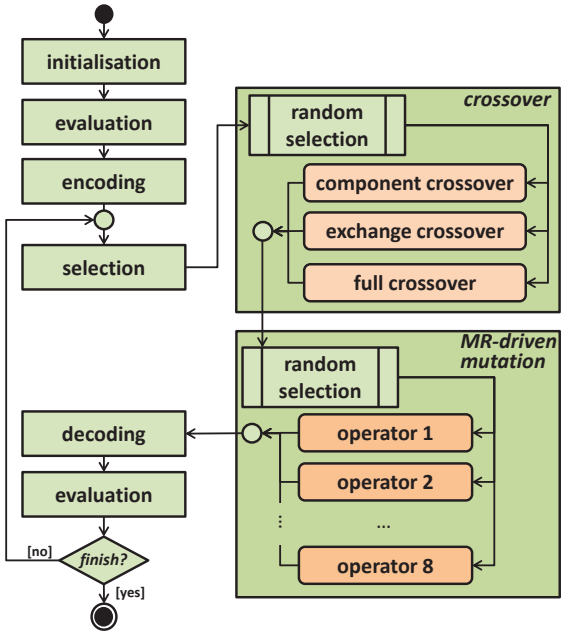


Figure 3: Main steps of the proposed evolutionary algorithm.

model, which is used as a seed to generate the initial population. Since the algorithm performs a guided search, each individual is synthesised by mutating the initial cloud according to the constraints reflected in the MRs (see the mutation step of the algorithm).

Evaluation. The individuals are evaluated using a fitness function, which provides a numeric value representing the quality of the candidate solution. This value is provided by a cloud simulator that calculates the energy consumption of each cloud model.

Encoding. In order to create the offspring, clouds are encoded in a way that facilitates its manipulation during the next stages of the algorithm. Clouds are complex systems made of a high quantity of connected components. Thus, it is necessary to design an encoding that appropriately represents all the required features and minimises dangling combinations requiring repairing or substitutions.

For this purpose, we studied the encodings proposed by two well-known EAs: classic Genetic Algorithm (GA) and Genetic Programming (GP). We discarded the use of the classic GA encoding because, even though it allows describing the cloud components, it has limitations to represent their interconnections. However, in GP, the relations between the different features of the models can be described using a tree encoding, which helps in representing the network connections. Nonetheless, we have identified several drawbacks complicating some steps of the algorithm. The first one is related to performance issues: the crossover and mutation steps require creating new versions of the cloud model, which is a computationally expensive task. Second, since we use MT to reduce the search space of the EA, it is necessary to use optimised structures for representing the individuals. Hence, we avoid the generation of dangling individuals to improve the overall performance of the algorithm.

For these reasons, we have designed a hybrid encoding based on graphs and integer representations, which facilitates the management of the large structures conforming the cloud, and reducing the generation of incorrect models. This way, we encode a cloud using two different vectors. The first one represents the physical components of the cloud (the racks), and the second one the network connections. For this purpose, we use the tuple-based representation for clouds explained in Section 5.1.

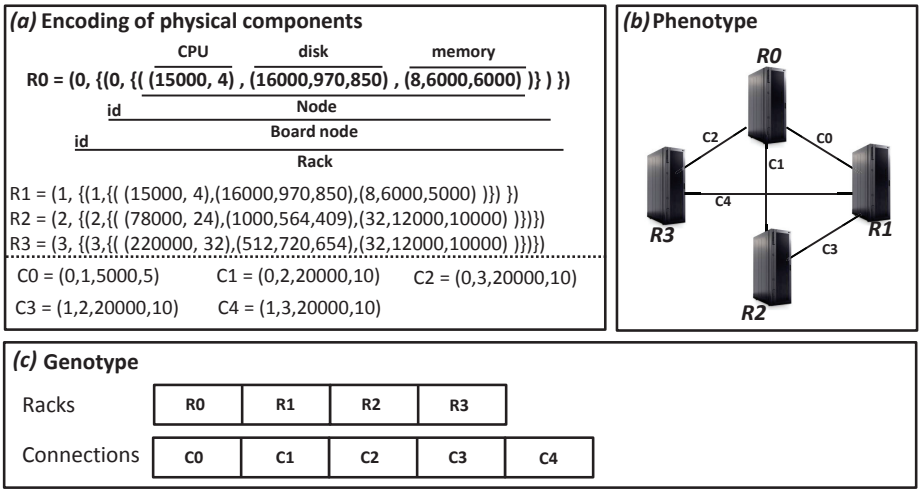


Figure 4: Encoding a cloud system: genotype and phenotype.

Example. Figure 4(a) shows the encoding of a cloud made of 4 racks (R0..R3), each one allocating a blade node, and 5 network connections (C0..C4). Rack R0 has a single board with one node. This node has a CPU with a speed of 15000 MFlops and 4 cores, a disk with 16000 GBytes providing a write and read bandwidth of 970 and 850 MBps, respectively, and a RAM memory of 8 GBytes providing a write and read bandwidth of 6000 MBps. C0 represents a network connection between racks R0 and R1, providing a bandwidth of 5000 MBps and latency of 5 μs. The genotype, i.e., the vectors containing the racks and the connections of the cloud, is depicted in Figure 4(c). The phenotype, representing the topology of the cloud, is shown in Figure 4(b).

Selection. In this phase, the best individuals – based on an energy-aware criterion – have a higher probability to be selected for generating a new offspring. We use *roulette wheel* as a selection method, which gives high probabilities to those clouds models with low energy consumption. These probabilities are based on the width of the slice of a hypothetical roulette wheel, where wider slices provide higher probabilities to be selected.

Crossover. In this phase, individuals are combined to generate a new offspring following the principles of biological reproduction. Crossover mainly depends on the encoding and, therefore, using an appropriate crossover design improves the performance of the EA. Since the inclusion of multiple crossover mechanisms prevents the premature convergence and improves the performance of the algorithm [79], we propose three crossover techniques focused on spe-

cific knowledge of cloud systems, where for each iteration, the technique to be applied is randomly selected. The crossover operators we use are explained next:

- *Mix crossover*: This crossover merges the information of two physical machines to generate a new offspring. Initially, the tester must provide the percentage of machines involved in the crossover. Hence, one part of the cloud will remain unmodified, unless the tester indicates a 100%. Next, for each pair of clouds (parents) from the actual population, two new individuals (offspring) are generated. For this purpose the components of the physical machine are encoded in binary. Hence, the information of each component (CPU, disk, memory, ...) from both parents is combined using the standard one-point crossover. Figure 5 illustrates this crossover, where $CPU1$ (blue) and $CPU2$ (red) are combined to generate two new CPUs, each one containing information from both parents.

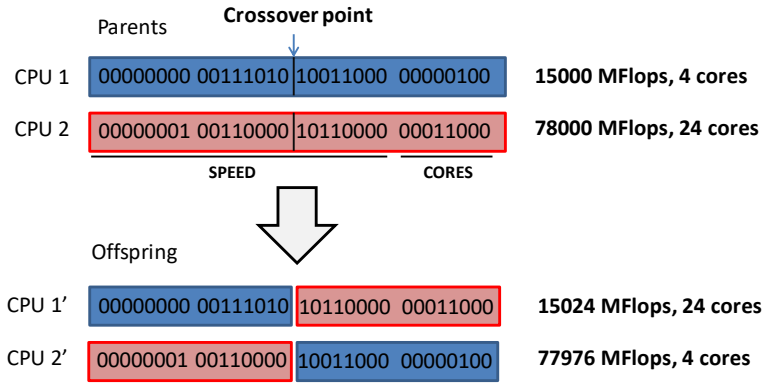


Figure 5: Example of mix crossover.

- *Swap crossover*: In this case the operator combines unmodified components from the parents to generate the offspring. Hence, the binary information of each component is not modified but placed in another individual. Similar to the mix crossover, the tester must specify the percentage of physical machines involved in the crossover. Thus, the machines of the new offspring are a combination of components from both parents.
- *Full crossover*: This crossover is inspired by a NASA contribution to evolve graph topologies [80]. The idea is to randomly divide two clouds, each one in two fragments, and combine them to generate two new individuals. Figure 6 shows an example, where two different graphs $G1$ and $G2$ are combined to generate two new individuals, $G1'$ and $G2'$. To allow recombination, the graphs $G1$ and $G2$ are partitioned in such a way that $G1_a$ and $G2_a$ have the same number of edges cut (2 in this case).

MR-driven mutation. In this step, random mutations are seeded into the individuals. The idea is to explore those cloud models that provide better energy consumption without enhancing its underlying components. In some cases, a mutation is applied towards degenerating a specific part of the individual. Thus, the new individual represents a cloud model containing a-priori "worse" components than the source individual (e.g. slower CPUs, smaller memories or a data-centre containing less physical machines). The changes are performed with a certain probability, where small modifications have greater chances to be performed than larger ones,

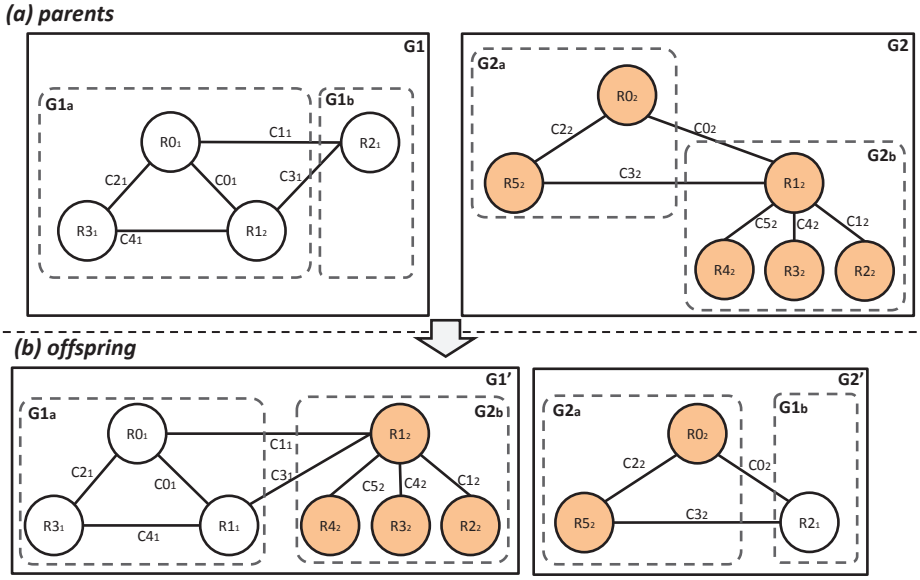


Figure 6: Example of full crossover.

and are seeded in a way that the new generated individuals satisfy the MRs. This is one of the salient features of this work, where the MRs conduct the search towards a reduced exploration space. We have designed the following set of 8 mutation operators:

- *Operator 1*: This operator mutates all the components of a rack. Initially, a rack is randomly selected from a given cloud (individual), where all the racks have the same probability to be selected. Then, the operator seeds changes on all the components of the rack (nodes, disk, CPUs, ...). The following example illustrates how this operator is applied to the cloud represented in Figure 4. $R2$ is the rack before the operator is applied, and $R2'$ after its application. The changes are remarked in boldface. In this case, the mutation is applied towards degenerating the source individual.
  - $R2 = (2, \{(2, \{((78000, 24), (1000, 564, 409), (32, 12000, 10000))\})\})$
  - $R2' = (2, \{(2, \{((\mathbf{57200}, \mathbf{13}), (1000, \mathbf{364}, \mathbf{209}), (32, \mathbf{4212}, \mathbf{3210}))\})\})$
- *Operator 2*: This operator decreases the bandwidth of a network link of the cloud that is randomly selected from the communication network.
- *Operator 3*: Similarly to the previous operator, this one increases the latency of a randomly selected network link.
- *Operator 4*: This operator removes a link of the network, and is only applied if the resulting graph is connected. Figure 7 (a) illustrates the result of applying this mutation operator to the model of Figure 4, where link $C3$ is removed.
- *Operator 5*: This operator creates a new link connecting two (randomly selected) racks that are not directly connected. The bandwidth and latency values of the new network link are set to the average of the existing links in the network. Figure 7 (b) shows the result of applying this mutation operator to the initial model, where the link C5 is added.

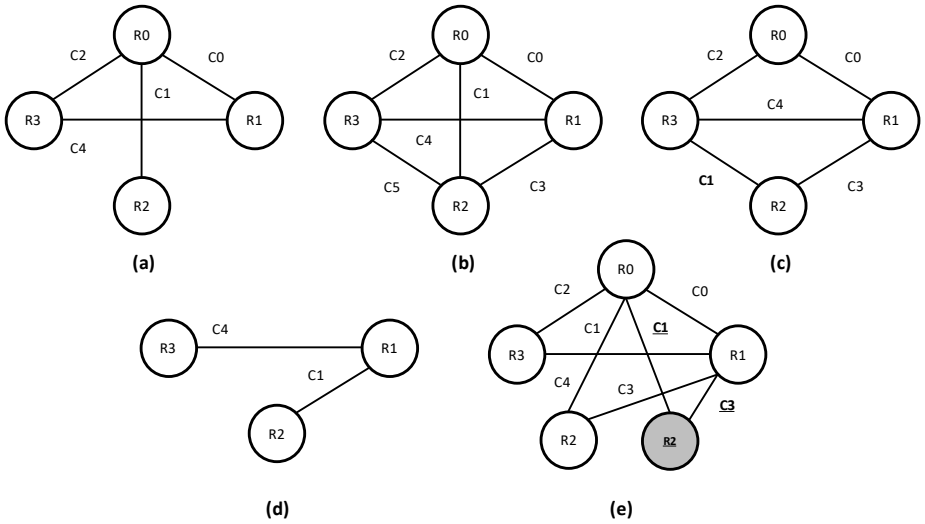


Figure 7: Example of mutation operator applications: (a) Operator 4 deletes link C3. (b) Operator 5 creates link C5. (c) Operator 6 replaces the source of $C1$ by $R3$. (d) Operator 7 deletes $R0$. (e) Operator 8 splits $R2$.

- *Operator 6*: This operator replaces the source/destination of a randomly selected link. An example application of this operator is depicted in Figure 7 (c), where the source of the link $C1$ ($R0$) is replaced by $R3$.
- *Operator 7*: This operator removes a randomly selected rack and all its connected links. If the resulting graph is unconnected, new links are created until the graph becomes connected. New connections are created based on the bandwidth and latency values of the removed connections. Destination nodes are randomly selected following the guidance of Operator 6. For this, it modifies the connection links to avoid unconnected graphs. Figure 7 (d) shows an example application, where R0 and all its connections are removed.
- *Operator 8*: This operator splits a rack in two, both having half of the capacities of the original one, and where the links are duplicated. An example application is shown in Figure 7 (e), where rack R2 and its links have been duplicated.

Decoding. A decoding process is required to evaluate the fitness of new and modified individuals. Although the crossover and mutation operators have been designed to avoid incorrect models, it is possible that some of the individuals do not satisfy some of the MRs provided as input, due to the multiple changes seeded in each model. In this case, the individual is replaced by another one that satisfies the MRs.

Please note that, as Figure 3 shows, after decoding, the individuals are evaluated again and the next population generated using those individuals with the lowest energy consumption.

Finally, let us analyse the computational cost of our approach. Our genetic algorithm performs a fixed number of iterations, denoted by *nIter*. This way, in each iteration the algorithm:

- Generates a new population by applying first a crossover mutation and then a mutation operator. The total population of clouds is multiplied by 3. We denote the size of the population by $nClouds$.

- Performs the simulations on all the elements of the population. We write $Sim(S)$ to denote the time needed to simulate one individual (cloud) in the simulator S.

All the mutation operators require just constant time except the full crossover algorithm depicted in Figure 6. The authors do not give its complexity, so we assume that it requires $\Omega(fCross(C))$, where C is the size of the individual (cloud). Hence, putting all together, our approach has a computational cost of $\Omega(nIter \cdot (nClouds \cdot (fCross(C) + Sim(S))))$.

7. Tool support

We have developed tool support for `MT-EA4Cloud`, which implements the different modules depicted in Figure 1 using Java. Its scheme is depicted in Figure 8, and shows how MT, EAs and simulation tools are combined to optimise the energy consumption in cloud systems. For the sake of clarity, only the most relevant parts are shown.

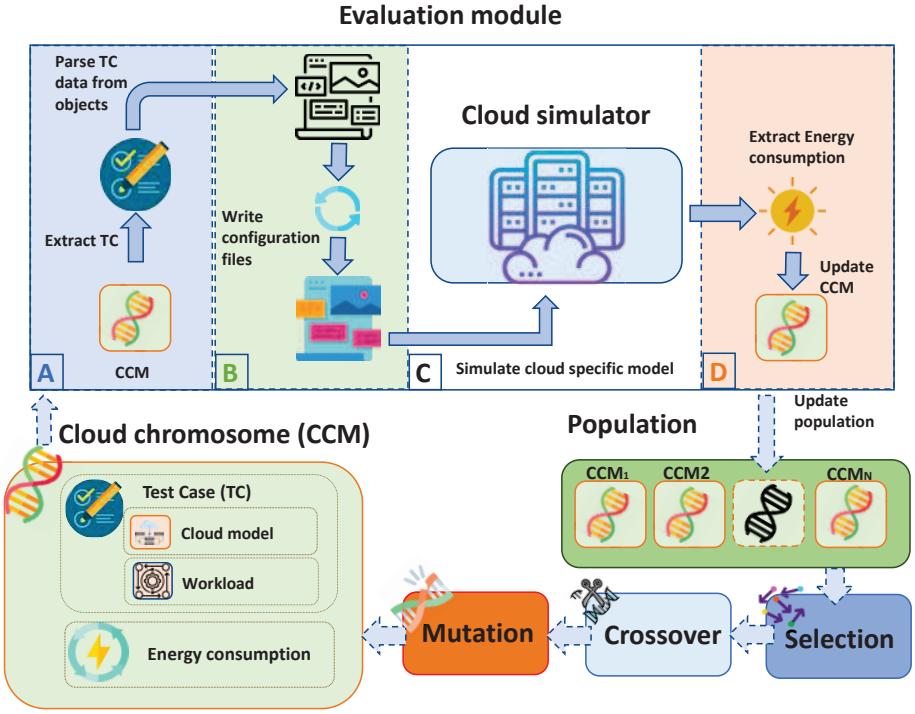


Figure 8: Tool support for `MT-EA4Cloud`: integration of the simulation tools within the EA scheme.

The Evaluation Module (EM) consists of 4 main submodules: the test case extraction module (labelled as A), the template transformation module (labelled as B), the cloud simulation module (labelled as C) and the energy consumption module (labelled as D).

Initially, the *cloud chromosome* (in short, CCM) is extracted from the population. A CCM consists of three main elements: a *cloud model* (in short, CM), a *test case* (in short, TC) and *energy consumption* (in short, EC). Test cases are automatically generated using metamorphic

testing techniques. Each TC contains a CM and a workload, where CM represents all the components used to model a cloud system, such as computational resources, network topology, and the workload refers to the operations to be processed by CM. Each object that represents a TC contains a path indicating the location of the test input data and a path where the results of the simulation are stored. The last element of the CMM, labelled as *Energy consumption*, refers to the amount of energy required by CM to execute the workload.

The information allocated in TC is parsed from the CCM to create a generic data structure. It is *generic* since it allocates the data required to configure a test case, but is not specific to a single simulator. The idea is to manage a common structure that can be applied to the different simulators deployed in the system, allowing an easy translation of this data into the different formats of the cloud simulators that are currently deployed (module A).

Next, the configuration files required to execute TC using a cloud simulator are generated (module B). The cloud simulator deployed in module C executes the simulation of the CM contained in TC. Once the simulation has finished, a results file is generated (see module D). The energy consumption is extracted from the results, and the CCM is updated and inserted into the population accordingly. The optimisation of the clouds is performed using the EA explained in Section 6, and depicted in Figure 3.

This framework has been developed using a modular and flexible design, which allows including easily new simulators and other approaches inspired by EAs to optimise the energy consumption in clouds. For this, the main submodules of the EM have been represented as Java interfaces in such a way that the integration has been reduced to the following three points:

- Implementing the interface *TestCaseTransformations* (see module B), to transform a cloud model to the specific format required by the new simulator.
- Implementing the interface *SimExecution* (see module C), to provide the specific format to execute the simulations in the new integrated simulator.
- Implementing the interface *EnergyExtraction* (see module C), to parse the results file generated by the execution of the new simulator.

8. Empirical study

This section reports on an empirical study, where `MT-EA4Cloud` is applied to check the correctness of different cloud systems. First, in Section 8.1, we formulate several research questions we aim at answering with the experiments. Second, we present a detailed description of the experimental setting in Section 8.2. Next, in Section 8.3, we evaluate the adequacy of each MR in our catalogue and the suitability of each simulator used in this study. Section 8.4 presents a sensitivity analysis, and in Section 8.5, we perform a testing process to evaluate the correctness of different clouds using our approach. Finally, we discuss the obtained results and answer the research questions in Section 8.6.

8.1. Research questions

The experiments described in this section seek to answer the following questions:

RQ1 *Is it feasible to analyse the correctness of energy-aware cloud systems using simulation?*

In general, the main difficulty of choosing a tool that properly represents the underlying behaviour of the cloud lies in the uncertainty of the provided results, that is, how the researcher

can be sure that the provided results properly represent the expected behaviour of the cloud? Hence, we are interested in analysing and comparing the suitability of different simulation tools to represent the features established by the user and, thus, to decide which simulation tool is most adequate to model and simulate these features.

RQ2 *How adequate are the MRs for analysing energy-aware clouds?*

In MT, the MRs model de underlying behaviour of the system under test. In this work, we provide a catalogue of MRs exclusively dealing with the energy consumption of cloud systems. Hence, we are interested in investigating the adequacy – from an energy-aware point of view – of the proposed MRs for studying cloud systems.

RQ3 *Is it possible to automatically detect drawbacks in the energy consumption of cloud systems and provide convenient solutions?*

We are interested in evaluating whether our methodology is capable of optimising the energy consumption of the clouds under test by locating drawbacks in their underlying architectures. A drawback is discovered in a cloud model when `MT-EA4Cloud` locates an alternative cloud model that provides better energy consumption without enhancing its underlying components, like increasing the CPU speed, using more physical machines in the data-centre or using a faster network. Additionally, we are interested in comparing the quality of the test cases generated using our proposed EA against the quality of randomly generated test cases.

8.2. Experimental setting

The main objective of our methodology is to check the correctness – from an energy consumption point of view – of cloud systems. Hence, the first step consists in analysing the cloud features having a direct impact on energy consumption, which are modelled in the form of MRs. In this study, the catalogue of MRs presented in Section 5.2 has been used to model the underlying behaviour of cloud systems formally.

In order to execute the experiments of this study, we have chosen different well-known simulators designed to model and simulate the energy consumption of cloud systems. First, we have carefully investigated, in the research papers found in the current literature, the main features and drawbacks of each simulator. Second, we have analysed the documentation provided by each simulator to decide whether a simulator is appropriate, or not, for this study. As a result, we provide the set $\mathcal{S}$ consisting of 7 simulators (step 2). Table 1 shows the MRs of the catalogue that can be modelled and simulated using each simulator $S \in \mathcal{S}$.

Id	MR_1	MR_2	MR_3	MR_4	MR_5	MR_6	MR_7	MR_8
CloudSim	✓	✓	✓	✗	✓	✗	✓	✓
CloudSimStorage	✓	✓	✓	✓	✓	✗	✓	✓
DCSIM	✓	✓	✓	✓	✓	✗	✓	✓
GreenCloud	✓	✓	✓	✗	✓	✗	✓	✓
SimGrid	✓	✓	✓	✗	✓	✗	✓	✓
iCanCloud	✓	✓	✓	✓	✓	✓	✓	✓
DISSECT-CF	✓	✓	✓	✓	✓	✗	✓	✓

Table 1: Analysis of different cloud simulators to represent the provided MRs.

Parameter / Cloud	*cloudA*	*cloudB*	*cloudC*
#Hosts	512	512	512
RAM (MBytes)	1024	16384	8192
CPU speed (MIPS)	1k	90k	20k
CPU cores	2	8	4
HDD size (TBytes)	1	1	1
HDD speed (Mbps)	20	350	100
Net bw (Mbps)	500	10000	10000
Net lat (us)	10	10	10

Table 2: Source cloud configurations.

Next, we manually design three clouds – *cloudA*, *cloudB* and *cloudC* – providing different configurations, each one representing a specific cloud profile. Thus, *cloudA* represents a low-profile cloud, providing slow CPUs, small RAM memories and a slow network; *cloudB* models a high-profile cloud, with large RAM memories, fast CPUs with 8 cores and a fast communication network; and *cloudC* represents a mid-profile cloud, with a fast communication network and a fair CPU and memory system. The configurations of these clouds are depicted in Table 2.

Additionally, we have created different workloads, which are inspired by operations performed in big data analysis. In particular, we use traces that represent the infrastructure of PlanetLab [81], to be executed by cloudSim, and a Map-Reduce based application [82], to be executed by simGrid. It must be noted that each simulator requires a specific type of application to be executed and, therefore, the same application cannot be executed in both simulators. Hence, the idea is to apply different workloads over different cloud systems to analyse the energy consumption in these systems. In the following, a trace is denoted by ω_{sim}^{size}, where *sim* is the simulator used to execute the trace and *size* represents the trace length. The size of a small trace is denoted by the sub-index *s*, the size of a medium trace – larger than the small trace – is denoted by *m*, and the size of the largest trace is denoted by *l*.

The set of source test cases, denoted by $\mathcal{T}$, is generated by combining the clouds presented in Table 2 and the three generated workloads. In step 4, we automatically generate a set of follow-up test cases using C and $\mathcal{T}$, as input. In this case, we generate the set $\mathcal{F}$ containing a total of 4000 follow-up test cases.

8.3. Assessing the effectiveness of the MRs and the suitability of the simulators

In this section, the catalogue of MRs C (see Section 5) is evaluated using Equation 1 to calculate the adequacy. This requires an MR, a simulator $S \in \mathcal{S}$ and a set of source test cases $\mathcal{T}$. We have calculated the adequacy for each pair $(MR, S) \in C \times \mathcal{S}$. The results are presented in Table 3, where each column refers to an MR, and each row represents a simulator. In essence, these results show the percentage of tests that fulfil the different MRs for each simulator.

In general, all simulators used in this study provide acceptable results to simulate cloud systems. However, there are some simulators that cannot model the features formulated in the MRs. For instance, GreenCloud does not provide capabilities to model MR_4 and MR_6. In other cases, the obtained results show that some simulators do not properly represent the expected behaviour of a cloud. For example, in the case of iCanCloud, it achieves low effectiveness in MR_2 and MR_7, while the rest of the simulators provide better results for these MRs. It must be noted that the catalogue of MRs is designed to represent a general view of the cloud and, thus, there are some specific situations where the MRs cannot reflect the real behaviour of the

Id	MR_1	MR_2	MR_3	MR_4	MR_5	MR_6	MR_7	MR_8
CloudSimStorage	100	40	100	100	99	-	100	100
GreenCloud	0	20	100	-	45	-	100	100
SimGrid	100	63	100	-	100	-	100	100
iCanCloud	100	39	100	100	89	84	73	100
$DISSECT-CF$	99	51	100	99	95	-	100	100

Table 3: Adequacy (in %) of each MR using different simulators.

cloud. These situations are represented when some follow-up test cases do not satisfy a given MR. For instance, when cloudSim-storage is used to simulate the follow-up test cases generated to evaluate MR_5, which obtains a 99% of adequacy score. In this case, only 1% of the cases cannot be represented by this MR.

As the table shows, MR_2 provides a low adequacy score, that is, a high number of test cases do not fulfil this MR. This means that none of the simulators used in this study provides significant results supporting the claim that MR_2 accurately represents the behaviour of a cloud system. Consequently, since this MR cannot be used in the testing process, it has been discarded.

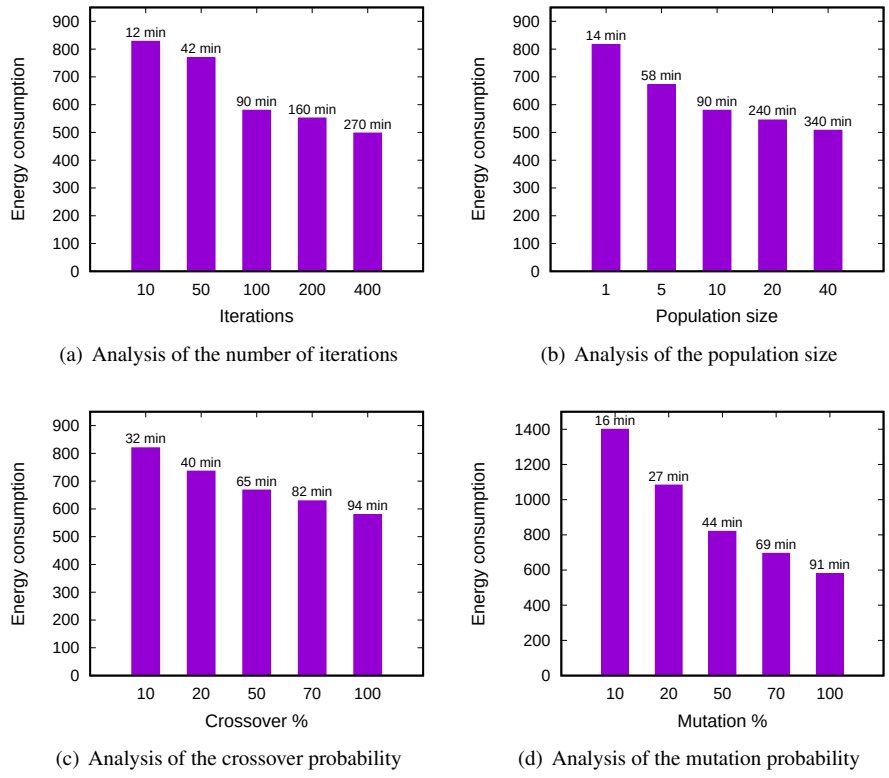
Table 3 also shows that MR_6 can only be modelled using iCanCloud. Usually, cloud simulators do not provide proper models for calculating the energy consumption in the memory system. Although iCanCloud implements this feature, the obtained results are not as promising as the ones obtained for evaluating other systems, like the computing or the storage systems.

Another feature that must be taken into account is the performance provided by the simulator. Since our methodology requires executing a large number of simulations, this aspect must be carefully taken into account. For example, iCanCloud requires almost 1 hour to simulate a small scenario, while simGrid provides the results in a few minutes.

Once all the simulators have been checked (step 5), we analyse the results presented in Table 3 to select the most appropriate simulators for the testing process. If we discard MR_2 and MR_6, we notice that cloudSimStorage and simGrid provide results achieving, at least, 99% of adequacy in the rest of MRs. Hence, we chose cloudSimStorage and simGrid as the most appropriate simulators, for two reasons. First, they provide high performance for executing the simulations. Second, the obtained results show that these simulators are suitable to model and simulate the required features of cloud models, represented in each $MR \in C$.

8.4. Sensitivity analysis

The efficiency and effectiveness of a GA strongly rely on the choice of its input parameter values. Hence, in order to carry out the experimental study using an appropriate setup, first we have conducted a sensitivity analysis to investigate how the parameters of the algorithm impact on the overall performance [83]. Our algorithm depends on four main parameters: the number of generations (iterations), the size of the population, and the application probability of the mutation and crossover operators. For this analysis, we use the cloud configuration *cloudA* described in Table 2 and the cloudSim simulator to perform the simulations. The basic idea is to run different experiments for optimising the processing of the workload ω^s in *cloudA* using different values for setting-up the input parameters and, therefore, to analyse the trade-off between the final result and the performance provided. The results of this study are depicted in Figure 9. The x-axis of each chart shows the value of the analysed parameter, while the y-axis depicts the obtained energy consumption of the cloud. The execution time is shown at the top of each bar.



(a) Analysis of the number of iterations

(b) Analysis of the population size

(c) Analysis of the crossover probability

(d) Analysis of the mutation probability

Figure 9: Results of the sensitivity analysis.

Firstly, we start the analysis with the *iterations* parameter. In general, the number of created generations is expected to have considerable impact on the obtained results. Hence, increasing the number of iterations may lead to a significant improvement of the obtained results. This way, we started the analysis with a low number of iterations and increased this value in order to find a good balance between the results obtained and the execution time. Figure 9 shows that using 100 iterations provides a good trade-off between performance and efficiency.

Next, we analyse the *population size*, that is, the number of generated individuals (clouds) in each generation. In this case, we observe that increasing the number of individuals has only a slight impact on the obtained results, and a larger impact on the execution time. For instance, executing the algorithm using 10 individuals requires 90 minutes, while using 40 individuals requires 340 minutes, providing slightly better results. For this purpose, we choose a population size of 10 individuals for configuring the algorithm.

Similarly, the *probability of using a crossover operator* has a low impact on the provided results. However, the performance of the algorithm is proportional to the probability of applying a crossover operator. Consequently, we use the medium value (50%) for configuring the algorithm.

Finally, we analyse the *probability of using a mutation operator*. This is clearly the parameter that has the greatest impact on the quality of the provided results. In order to analyse this

parameter, we use values in the range [10-100]. Let us remark that this value represents the probability of applying a mutation operator over the current individual. Hence, using a value of 100, the algorithm mandatorily applies a mutation operation in each iteration of the algorithm for every processed individual. Figure 9.d shows that using a 10% of probability for applying a mutation operator only requires 16 minutes of execution time, providing poor results. On the contrary, using a probability of 100% significantly increases the quality of the results, achieving an effectiveness greater than 200%. Hence, we use a value of 100% for this parameter in the experiments, which corresponds with the configuration labelled as *high* in Table 4.

8.5. Checking the correctness of energy-aware cloud systems

In this section, we perform a set of experiments to analyse the correctness of the clouds designed in step 3. We test a cloud by measuring the required amount of energy to execute a workload. The testing process is conducted by using two different techniques, our proposed combination of MT and EAs, and a random approach. Both techniques focus on automatically generating test cases, where a test case consists of a workload and a cloud configuration. Their main difference lies in how the cloud configuration is generated. While our approach performs a guided search that uses the "best" clouds of each generation to create a new offspring of individuals, the second approach generates new clouds by apply mutations randomly over the original cloud. It is important to remark that both approaches generate clouds that satisfy the input constraints of the MRs. Hence, since the former approach performs a guided search, we expect to outperform the random approach both in performance and efficiency.

Our EA evolves different generations of individuals using three different crossovers and eight mutation operators. The crossover to be applied to each individual is randomly selected, while we have tried three different probability levels (three configurations) for applying the mutation operators. Although the sensitivity analysis carried out in Section 8.4 shows that using a high probability for applying a mutation operator provides the best results, we use three different configurations to validate this assumption. As a remark, since each simulator requires a specific configuration to model the cloud, there are some mutation operators that cannot be applied in certain simulators. In order to alleviate this issue, we provide a specific configuration for each simulator.

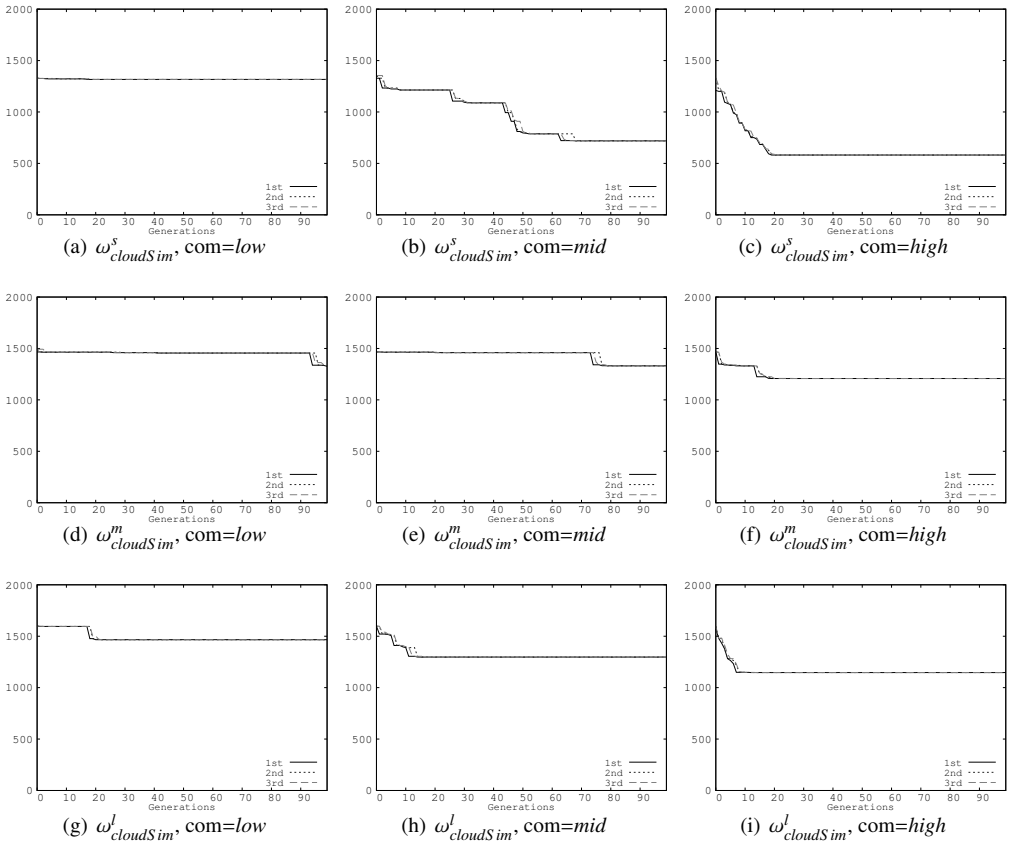
Table 4 shows the configurations – *low*, *mid* and *high* – used in the testing process when cloudSim executes the simulations. Each column represents the probability of each mutation operator to be applied, where the sum of the probabilities must be less or equal than 100. In the cases where the sum is less than 100, we use the difference to represent the null operator, that is, none of the mutation operators is applied. On the contrary, when the sum of the probabilities is 100, one of the operators is mandatorily applied. For instance, using the *mid* configuration of Table 4, there is a probability of 100-(15+10+5+1)=69% to apply the null operator.

Config	$mutOp_1$	$mutOp_2$	$mutOp_3$	$mutOp_4$	$mutOp_5$	$mutOp_6$	$mutOp_7$	$mutOp_8$
low	1.5%	1.5%	1%	–	–	–	0.5%	–
mid	15%	10%	5%	–	–	–	1%	–
high	25%	25%	25%	–	–	–	25%	–

Table 4: Configuration to apply the mutation operators in cloudSim.

Figures 10, 11 and 12 show the results obtained in the testing process using cloudSim to execute the simulations and the individuals *cloudA*, *cloudB* and *cloudC* as a seed to generate the

initial population, respectively. Each figure contains 9 charts, corresponding to the combination of executing three different workloads using three different configurations. Thus, the charts of the same row represent the simulations that use the same workload, while the charts of the same column represent the simulations that use the same configuration for applying the mutation operators (in short, *com*). The x-axis of each chart represents the generations of individuals (clouds) and the y-axis represents the energy consumption of each cloud, measured in kW. Each chart shows the three best individuals from each generation. The idea is to investigate how the best individuals of each generation evolve and, thus, to analyse how their genetic information – transferred to further generations – affects the energy consumption. The solid line represents the best individual of each generation (*1st*), the dotted line represents the second (*2nd*) and the dashed line represents the third (*3rd*).



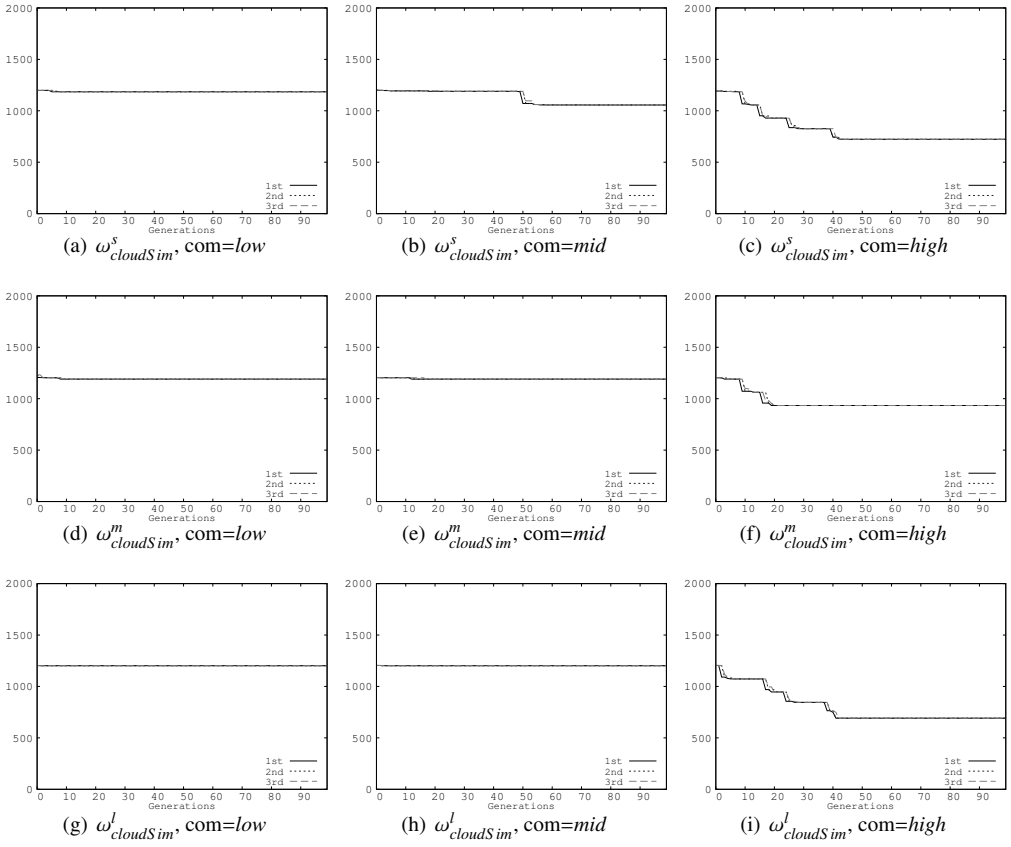
(a) $\omega^{s}_{cloudSim}$, com=*low* (b) $\omega^{s}_{cloudSim}$, com=*mid* (c) $\omega^{s}_{cloudSim}$, com=*high*

(d) $\omega^{m}_{cloudSim}$, com=*low* (e) $\omega^{m}_{cloudSim}$, com=*mid* (f) $\omega^{m}_{cloudSim}$, com=*high*

(g) $\omega^{l}_{cloudSim}$, com=*low* (h) $\omega^{l}_{cloudSim}$, com=*mid* (i) $\omega^{l}_{cloudSim}$, com=*high*

Figure 10: Testing process using cloudSim and *cloudA* as a seed to generate the initial population.

Confirming the results of the sensitivity analysis, Figure 10 shows that the configuration used for mutating the individuals has a significant impact on the quality of the further generations. That is, using a configuration with a low mutation probability provides a slight improvement in the offsprings, while using a configuration with a high mutation probability provides substantial



(a) $\omega^{s}_{cloudSim}$, com=*low* (b) $\omega^{s}_{cloudSim}$, com=*mid* (c) $\omega^{s}_{cloudSim}$, com=*high*

(d) $\omega^{m}_{cloudSim}$, com=*low* (e) $\omega^{m}_{cloudSim}$, com=*mid* (f) $\omega^{m}_{cloudSim}$, com=*high*

(g) $\omega^{l}_{cloudSim}$, com=*low* (h) $\omega^{l}_{cloudSim}$, com=*mid* (i) $\omega^{l}_{cloudSim}$, com=*high*

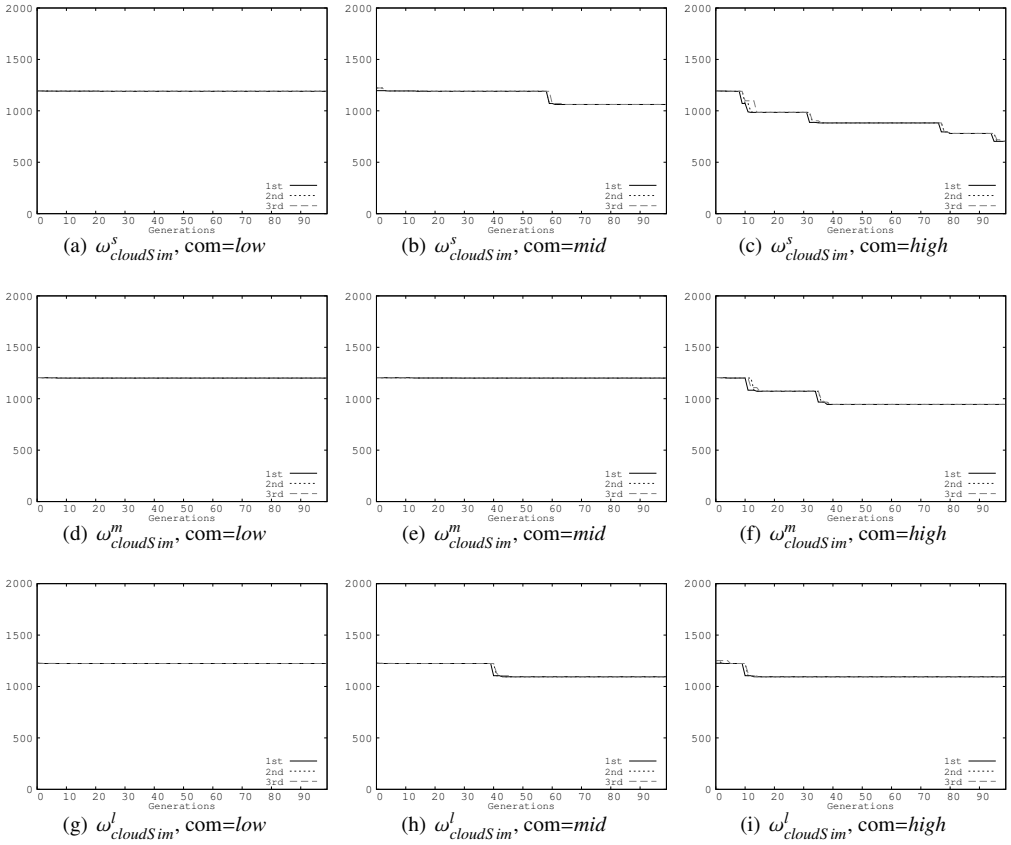
Figure 11: Testing process using cloudSim and *cloudB* as a seed to generate the initial population.

improvement. This fact is appreciated in the charts of the same row, which use the same workload. We can also notice that the workload also has an impact on energy consumption. In this case, the larger workload $\omega^{l}_{cloudSim}$ requires more energy to be processed than the other workloads. These results also show that the tendency of each generation, in the sense of improving the energy consumed by the cloud, is practically the same. This can be appreciated in how the best three individuals of each generation are similarly improved along with the generations.

Figure 11 shows that the *low* configuration provides similar results for all the workloads. The *mid* configuration enhances *cloudB* only for the smallest workload, that is, $\omega^{s}_{cloudSim}$ (see Figure 11.b). On the contrary, the *high* configuration provides notable improvements along the generations. In this case, the EA is not able to find a proper optimisation of the cloud using workload $\omega^{m}_{cloudSim}$ (see Figure 11.f). This effect can be observed by looking at Figure 11.i, which shows that the EA provides an enhanced cloud for executing $\omega^{l}_{cloudSim}$, which theoretically must require more energy than the execution of $\omega^{m}_{cloudSim}$.

Similarly, Figure 12 depicts that the *high* configuration is more suitable to enhance the initial cloud than the rest of the configurations (see Table 4). In this case, when using the *mid* con-

figuration, the EA slightly enhances the initial cloud only for processing $\omega^{s}_{cloudSim}$ and $\omega^{l}_{cloudSim}$. On the contrary, when the *high* configuration is used, the obtained improvement is inverse in proportion to the size of the processed workload (see Figure 12.c, 12.f and 12.i).



(a) $\omega^{s}_{cloudSim}$, com=*low* (b) $\omega^{s}_{cloudSim}$, com=*mid* (c) $\omega^{s}_{cloudSim}$, com=*high*

(d) $\omega^{m}_{cloudSim}$, com=*low* (e) $\omega^{m}_{cloudSim}$, com=*mid* (f) $\omega^{m}_{cloudSim}$, com=*high*

(g) $\omega^{l}_{cloudSim}$, com=*low* (h) $\omega^{l}_{cloudSim}$, com=*mid* (i) $\omega^{l}_{cloudSim}$, com=*high*

Figure 12: Testing process using cloudSim and *cloudC* as a seed to generate the initial population.

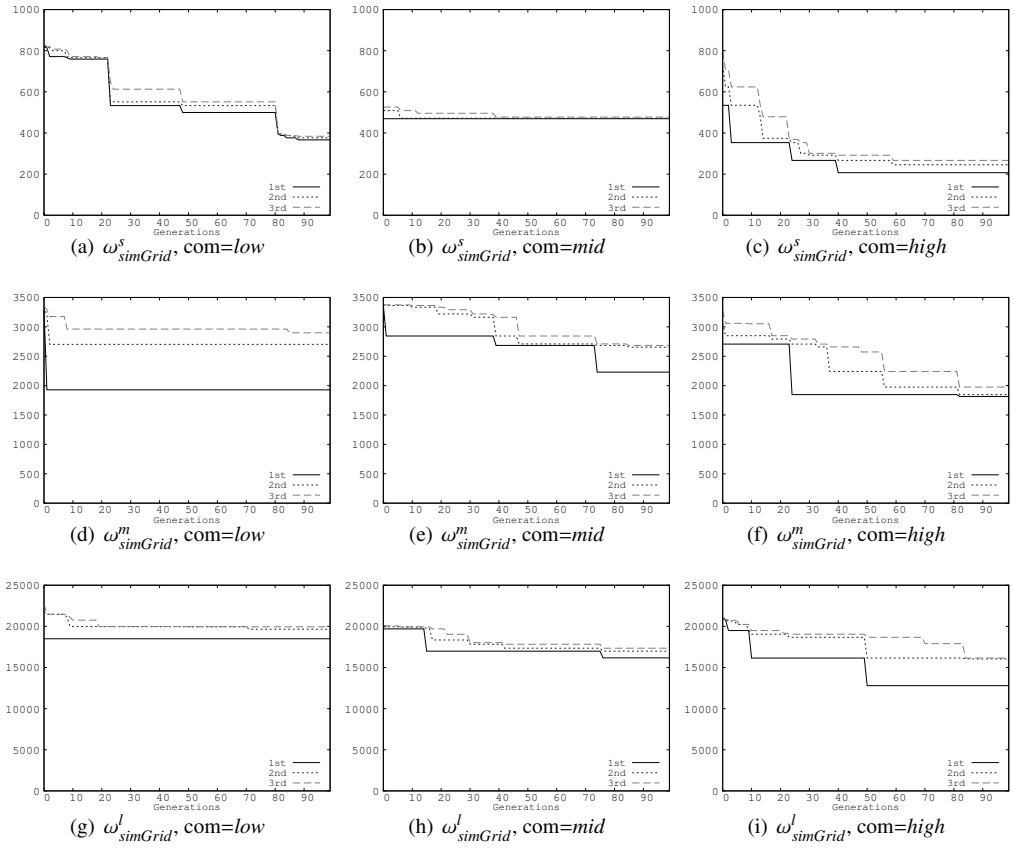
Figures 13, 14 and 15 show the results of applying our EA to *cloudA*, *cloudB* and *cloudC* using the simGrid simulator. Table 5 shows the configurations used in this experiment.

Config	$mutOp_1$	$mutOp_2$	$mutOp_3$	$mutOp_4$	$mutOp_5$	$mutOp_6$	$mutOp_7$	$mutOp_8$
low	1%	1%	1%	0.25%	0.25%	0.25%	1%	0.25%
mid	10%	10%	5%	1%	1%	1%	5%	1%
high	20%	20%	20%	5%	5%	5%	20%	5%

Table 5: Configuration to apply the mutation operators in simGrid.

When the EA is applied to *cloudA* (see Figure 13), we observe that the energy consumption of the cloud is proportional to the size of the executed workload. Although this fact can also be appreciated when the simulations are executed using the cloudSim simulator, in this case,

the difference between the energy consumptions is more significant. For instance, executing $\omega^{m}_{simGrid}$ requires 5 times more energy than the execution of $\omega^{s}_{simGrid}$ (see Figures 13.a and 13.d). Moreover, increasing the mutation probability provides a greater improvement in the offsprings than using a low probability. Nevertheless, there is an exception in Figure 13.d, where using the *low* configuration provides almost the same results – for the best individual – than using a higher probability to perform the mutations (see Figure 13.f). This is caused by the stochastic nature of the algorithm.



(a) $\omega^{s}_{simGrid}$, com=*low* (b) $\omega^{s}_{simGrid}$, com=*mid* (c) $\omega^{s}_{simGrid}$, com=*high*

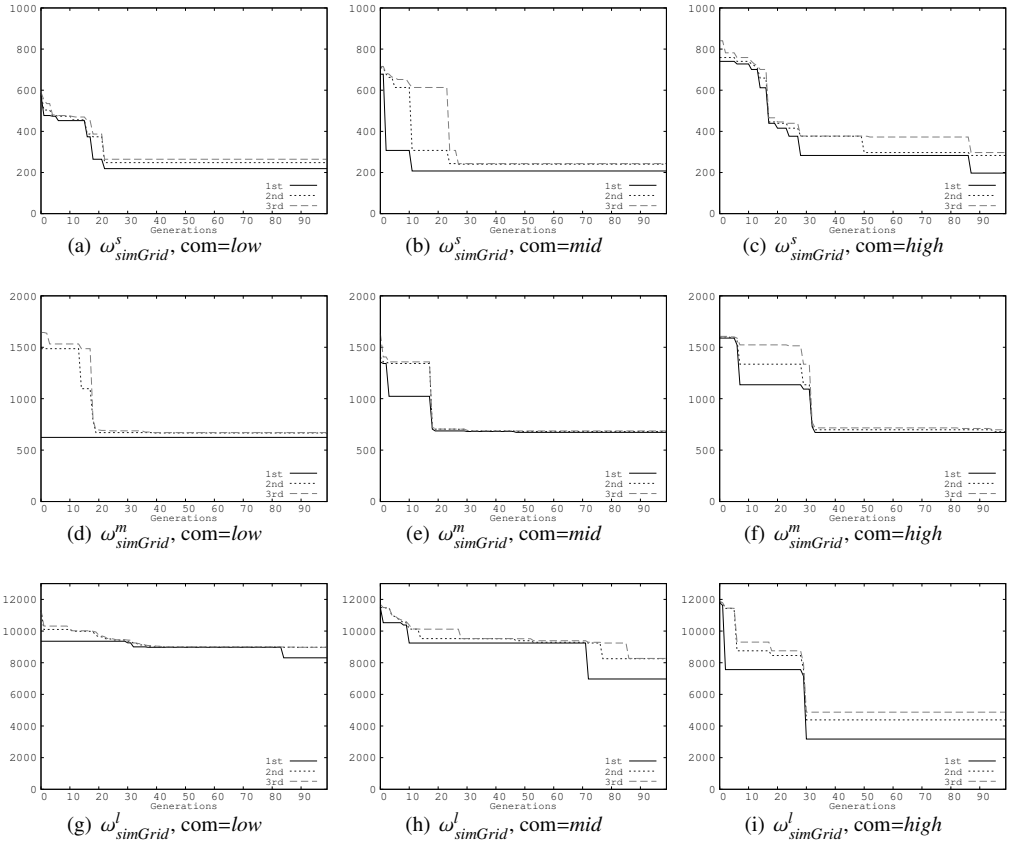
(d) $\omega^{m}_{simGrid}$, com=*low* (e) $\omega^{m}_{simGrid}$, com=*mid* (f) $\omega^{m}_{simGrid}$, com=*high*

(g) $\omega^{l}_{simGrid}$, com=*low* (h) $\omega^{l}_{simGrid}$, com=*mid* (i) $\omega^{l}_{simGrid}$, com=*high*

Figure 13: Testing process using simGrid and *cloudA* as a seed to generate the initial population.

Figure 14 shows that the EA provides a similar tendency in the energetic consumption, using all the configurations for applying mutation operators, when $\omega^{s}_{simGrid}$ and $\omega^{m}_{simGrid}$ are processed. However, when the cloud processes $\omega^{l}_{simGrid}$ using the *high* configuration, the obtained cloud configuration significantly improves the energy consumption (see Figure 14.i).

The charts depicted in Figure 15 show similar results than the ones obtained in the previous experiments. However, in this case, we identify a significant difference in the energy consumption between the first three individuals of each generation (see Figure 15.i). On the contrary, the rest of the cases show a relatively similar improvement for the best three individuals.



(a) $\omega^{s}_{simGrid}$, com=*low* (b) $\omega^{s}_{simGrid}$, com=*mid* (c) $\omega^{s}_{simGrid}$, com=*high*

(d) $\omega^{m}_{simGrid}$, com=*low* (e) $\omega^{m}_{simGrid}$, com=*mid* (f) $\omega^{m}_{simGrid}$, com=*high*

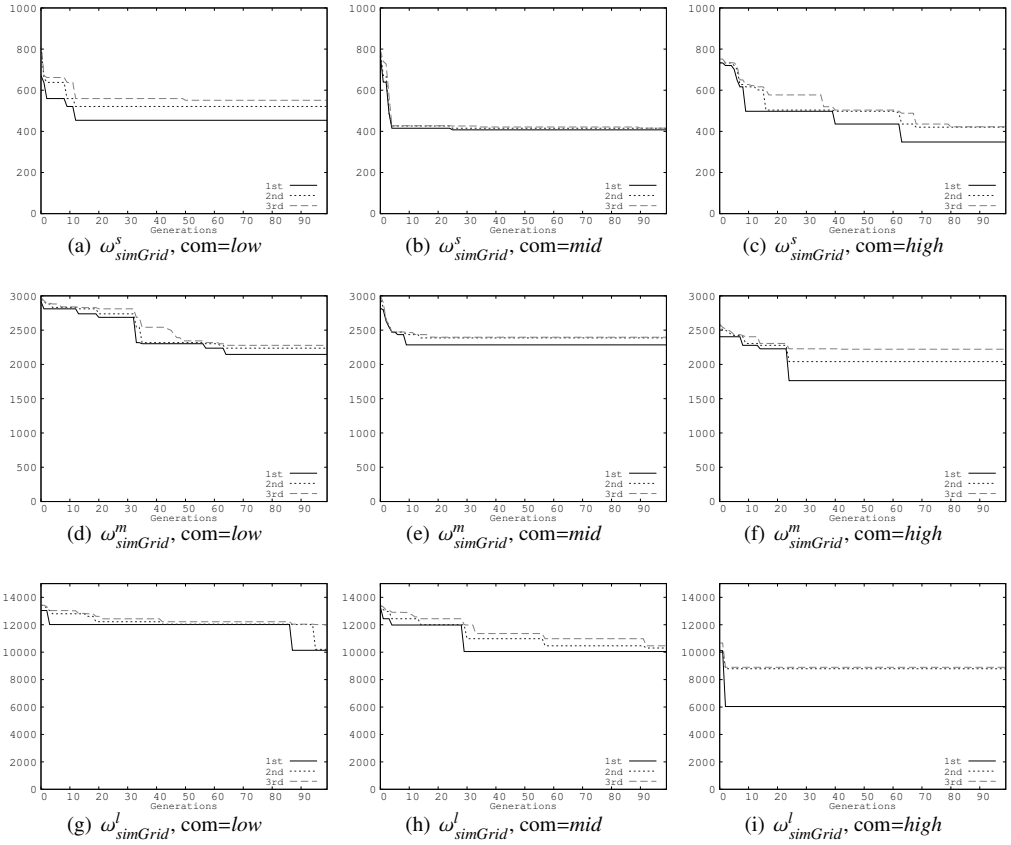
(g) $\omega^{l}_{simGrid}$, com=*low* (h) $\omega^{l}_{simGrid}$, com=*mid* (i) $\omega^{l}_{simGrid}$, com=*high*

Figure 14: Testing process using simGrid and *cloudB* as a seed to generate the initial population.

Overall, the results obtained in these experiments show that the *high* configuration provides the best cloud optimisations. Also, this configuration is the fastest to find a proper optimisation of the cloud, that is, the final result is reached by creating fewer generations of individuals than the other configurations. In general, the reached solution outperforms, in the sense of energy consumption, the initial cloud configuration.

In order to check the effectiveness of our EA, we present an experiment comparing the best cloud optimisation obtained by the EA against the results obtained by an approach that randomly applies mutation operators to generate new cloud configurations. From now on, we refer to the second approach as *random approach*. In both approaches, the MRs are used to check the new generated clouds, which must fulfil the constraints reflected in the input part of each MR.

Figure 16 shows the results obtained for simulating, using cloudSim, the executions of the workloads $\omega^{s}_{simGrid}$, $\omega^{m}_{simGrid}$ and $\omega^{l}_{simGrid}$ over *cloudA*, *cloudB* and *cloudC*. In this experiment, for each cloud, 150 different configurations are generated. Each new configuration is generated by applying one mutation operator, which is randomly selected, over the initial cloud. The x-axis of each chart represents the generated cloud configurations (individuals), and the y-axis represents



(a) $\omega^{s}_{simGrid}$, com=*low* (b) $\omega^{s}_{simGrid}$, com=*mid* (c) $\omega^{s}_{simGrid}$, com=*high*

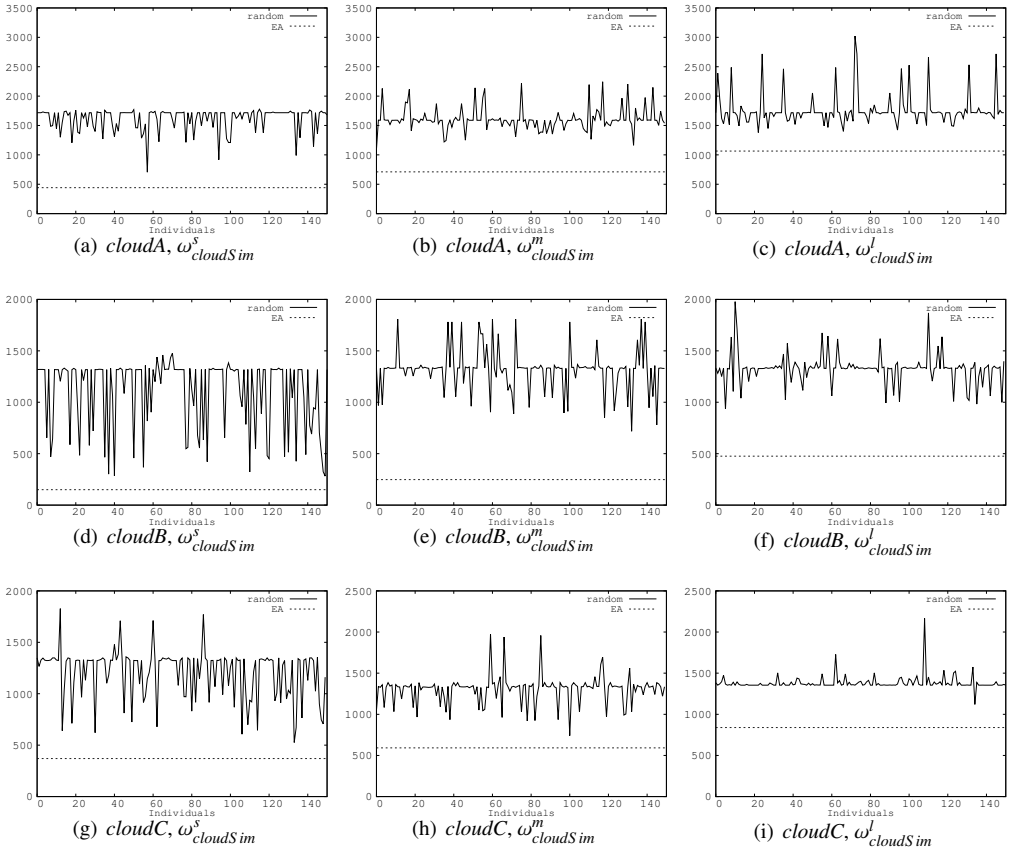
(d) $\omega^{m}_{simGrid}$, com=*low* (e) $\omega^{m}_{simGrid}$, com=*mid* (f) $\omega^{m}_{simGrid}$, com=*high*

(g) $\omega^{l}_{simGrid}$, com=*low* (h) $\omega^{l}_{simGrid}$, com=*mid* (i) $\omega^{l}_{simGrid}$, com=*high*

Figure 15: Testing process using simGrid and *cloudC* as a seed to generate the initial population.

the energy consumption of each cloud. The solid line shows the energy consumption of the new generated clouds, while the dashed line represents the best cloud configuration reached by the EA. In this figure, each row of charts represents the same cloud configuration, while each column of charts represents the same workload. The EA clearly provides the best result in all cases. The random approach provides similar results for each cloud using all the workloads. However, the charts show some noticeable peaks due to the stochastic nature of the generated clouds. In general, the difference between the EA and the random approach is greater when the small workload is processed. On the contrary, the difference between these approaches is smaller when the large workload is executed in the clouds.

Figure 17 shows the results when the simGrid simulator is used to execute the workloads over the clouds. In this case, both approaches provide similar results for processing the small and medium workloads. Similarly to the previous experiment, the EA is better than the random approach in all the cases.



(a) *cloudA*, $\omega^{s}_{cloudSim}$ (b) *cloudA*, $\omega^{m}_{cloudSim}$ (c) *cloudA*, $\omega^{l}_{cloudSim}$

(d) *cloudB*, $\omega^{s}_{cloudSim}$ (e) *cloudB*, $\omega^{m}_{cloudSim}$ (f) *cloudB*, $\omega^{l}_{cloudSim}$

(g) *cloudC*, $\omega^{s}_{cloudSim}$ (h) *cloudC*, $\omega^{m}_{cloudSim}$ (i) *cloudC*, $\omega^{l}_{cloudSim}$

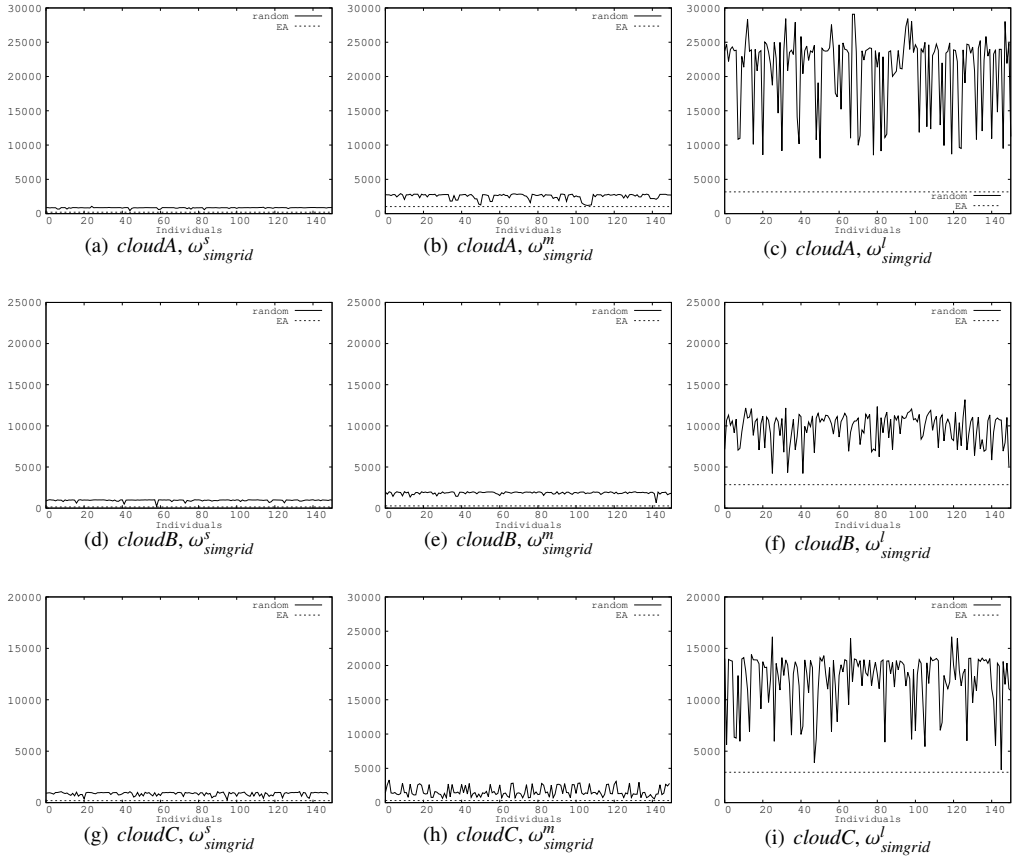
Figure 16: Testing process using cloudSim and randomly generated cloud configurations.

8.6. Discussion of the results and answers to the research questions

In this subsection, we discuss the obtained results and answer the research questions presented in Section 8.1.

*8.6.1. **RQ1**: Is it feasible to analyse the correctness of energy-aware cloud systems using simulation?*

In order to answer this question, we have designed `MT-EA4Cloud`, a methodology to check the correctness – from an energy consumption point of view – of different cloud systems. We have carried out an empirical study, which has been supported by our proposed methodology, to check the suitability of seven simulators for modelling and simulating the energy consumption of three different cloud systems (see Section 8.2). The analysed features of the cloud have been modelled in the form of MRs (see Section 5.2). We identify those features that can be modelled for each simulator in Table 1 and, thus, we are able to decide if a given simulator appropriately simulates these features, which represent the underlying behaviour of the cloud. In this case, we



(a) *cloudA*, $\omega^{s}_{simgrid}$ (b) *cloudA*, $\omega^{m}_{simgrid}$ (c) *cloudA*, $\omega^{l}_{simgrid}$

(d) *cloudB*, $\omega^{s}_{simgrid}$ (e) *cloudB*, $\omega^{m}_{simgrid}$ (f) *cloudB*, $\omega^{l}_{simgrid}$

(g) *cloudC*, $\omega^{s}_{simgrid}$ (h) *cloudC*, $\omega^{m}_{simgrid}$ (i) *cloudC*, $\omega^{l}_{simgrid}$

Figure 17: Testing process using simGrid and randomly generated cloud configurations.

have discarded two simulators from the initial list $\mathcal{S}$ and, therefore, the remaining simulators are used to carry out the testing process in the following experiments.

We can conclude that the answer to **RQ1** is *yes, it is feasible to analyse the correctness of energy-aware cloud systems using simulation*. However, there are some steps that must be manually performed by the expert, like designing an appropriate catalogue of MRs. Although the major part of the work can be automated – generating a large number of follow-up test cases and simulating a wide spectrum of cloud models – it is key that the expert takes the right decisions in those steps requiring the user intervention.

*8.6.2. **RQ2**: How adequate are the MRs for analysing energy-aware clouds?*

In order to answer this question, we provide a complete catalogue of MRs that model the underlying behaviour of cloud systems regarding energy consumption (see Section 5.2). The adequacy of each MR is calculated using Equation 1 and shown in Table 3.

After a careful analysis of the results depicted in Table 3, we can conclude that *the catalogue of MRs is adequate to analyse energy-aware clouds*. First, the adequacy of each MR is calculated

to discard those that do not properly model the underlying behaviour of a cloud environment. Second, the major part of the accepted MRs provides promising results reaching, at least, a 99% of adequacy score. Hence, designing accurate MRs is crucial for successfully applying our proposed methodology and, consequently, the quality of the results heavily depends on the accuracy of the MRs to model the underlying behaviour of the cloud.

8.6.3. ***RQ3****: Is it possible to automatically detect drawbacks in the energy consumption of cloud systems and provide convenient solutions?*

To answer this question, we have conducted an empirical study (see Section 8.5) where three different cloud systems have been analysed using two well-known simulators, simGrid and cloudSim. In essence, we use two different approaches to test the clouds. The first approach (labelled as EA) uses our EA to find a proper optimisation of the cloud under test, while the second approach (labelled as Random) randomly applies a mutation operator to the cloud under test for generating new cloud configurations. We analyse the quality of the results by calculating a 95% confidence interval using the results obtained from executing 30 times each experiment. When the EA approach is used, we focused on analysing the clouds using the *high* configuration for applying the mutation operators (com=high). We summarise the results obtained from this study in Table 6, 7, 8 and 9.

In the tables, the first column represents the cloud under test and the executed workload. The subsequent columns show statistics of the obtained results. Each value represents the energy required by the cloud to execute the workload. Thus, *min* refers to the "best" cloud configuration generated, in the sense of energy consumption, using as a basis the cloud under test, *max* refers to the "worst" generated cloud, *avg* refers to the energy consumption average of all the generated cloud configurations – using the cloud under test as basis – to execute a specific workload, *median* represent the result located at the midpoint of all the generated data from the experiments, and *interval* represents the interval of all the obtained results with a 95% of confidence.

It is important to remind that each simulator executes a specific type of workload (see Section 8.2) and, therefore, the results obtained from cloudSim cannot be compared with those obtained from simGrid.

Configuration		EA				
Cloud	Workload	min	max	avg	median	interval
cloudA	$\omega^{s}_{cloudSim}$	441.78	1189.99	627.16	600.30	572.95-681.38
	$\omega^{m}_{cloudSim}$	711.76	1225.62	873.82	818.27	815.09-932.54
	$\omega^{l}_{cloudSim}$	1065.37	1255.38	1147.33	1109.72	1123.05-1171.61
cloudB	$\omega^{s}_{cloudSim}$	150.35	1184.74	376.00	270.76	312.27-439.74
	$\omega^{m}_{cloudSim}$	248.15	1190.40	390.92	301.77	338.27-443.56
	$\omega^{l}_{cloudSim}$	476.57	1201.55	569.09	549.86	514.06-624.12
cloudC	$\omega^{s}_{cloudSim}$	370.04	1316.57	549.69	531.47	516.42-582.95
	$\omega^{m}_{cloudSim}$	592.49	1455.49	640.26	570.21	590.01-690.51
	$\omega^{l}_{cloudSim}$	839.90	1466.18	1026.25	1087.43	1002.11-1050.39

Table 6: Summary of the results obtained with the EA approach using cloudSim.

The provided results show that our EA (Tables 6 and 8) clearly provides better results than the random approach (Tables 7 and 9). Also, it is worth to mention that the intervals provided using the EA are significantly smaller than the ones provided using the random approach.

Configuration		Random				
Cloud	Workload	min	max	avg	median	interval
cloudA	$\omega^{s}_{cloudSim}$	705.67	1759.13	1221.7	1447.95	1092.42-1349.00
	$\omega^{m}_{cloudSim}$	1163.35	2244.31	1435.51	1588.53	1313.59-1557.41
	$\omega^{l}_{cloudSim}$	1305.64	3202.00	1597.73	1717.37	1495.20-1700.24
cloudB	$\omega^{s}_{cloudSim}$	250.46	1459.59	1084.47	1317.25	954.74-1214.20
	$\omega^{m}_{cloudSim}$	851.47	1809.05	1172.55	1330.54	1049.54-1295.55
	$\omega^{l}_{cloudSim}$	934.69	1978.51	1229.87	1329.90	1141.19-1318.55
cloudC	$\omega^{s}_{cloudSim}$	525.83	1827.10	1102.25	1322.55	977.11-1227.39
	$\omega^{m}_{cloudSim}$	944.55	1927.97	1203.84	1333.60	1100.52-1307.15
	$\omega^{l}_{cloudSim}$	1120.24	2165.27	1386.71	1354.38	1359.65-1413.76

Table 7: Summary of the results obtained with the random approach using cloudSim.

Configuration		EA				
Cloud	Workload	min	max	avg	median	interval
cloudA	$\omega^{s}_{simGrid}$	209.99	903.71	548.07	619.42	483.71-612.42
	$\omega^{m}_{simGrid}$	1030.92	3500.75	1496.69	1390.14	1309.51-1683.87
	$\omega^{l}_{simGrid}$	2951.03	23863.45	10927.01	8852.95	8898.12-12955.89
cloudB	$\omega^{s}_{simGrid}$	139.82	806.33	260.19	265.94	246.75-273.63
	$\omega^{m}_{simGrid}$	270.63	1882.89	680.81	640.40	650.47-840.15
	$\omega^{l}_{simGrid}$	2853.73	11926.45	4871.01	4317.24	4104.33-5637.69
cloudC	$\omega^{s}_{simGrid}$	194.70	922.13	371.54	332.83	307.10-435.99
	$\omega^{m}_{simGrid}$	302.70	2724.51	974.61	888.25	815.38-1133.83
	$\omega^{l}_{simGrid}$	3176.00	15257.92	5397.87	5181.84	5041.27-5754.47

Table 8: Summary of the results obtained with the EA approach using simGrid.

Configuration		Random				
Cloud	Workload	min	max	avg	median	interval
cloudA	$\omega^{s}_{simGrid}$	483.20	1065.15	606.74	629.11	575.10-638.38
	$\omega^{m}_{simGrid}$	1192.39	3895.14	1883.06	1438.18	1622.71-2143.41
	$\omega^{l}_{simGrid}$	8676.82	29083.71	13372.97	11987.82	12360.62-15098.83
cloudB	$\omega^{s}_{simGrid}$	208.14	1022.51	314.80	325.91	277.69-351.90
	$\omega^{m}_{simGrid}$	656.42	1997.83	1356.16	1323.98	1302.68-1409.64
	$\omega^{l}_{simGrid}$	4326.31	13180.23	6816.81	6428.38	6013.58-7620.03
cloudC	$\omega^{s}_{simGrid}$	195.95	1022.81	553.77	636.06	494.26-615.28
	$\omega^{m}_{simGrid}$	702.22	2921.28	1938.03	2073.34	1724.71-2151.34
	$\omega^{l}_{simGrid}$	5962.29	16121.01	9781.57	10531.91	9245.66-10317.49

Table 9: Summary of the results obtained with the random approach using simGrid.

These results also show that cloudSim provides similar results to execute the three workloads. However, the results provided by the simulations executed using simGrid show otherwise, where the energy consumption of the cloud is proportional to the size of the processed workload. This is also observed in the size of the interval, which is greater when the cloudSim simulator is used to simulate the cloud under study. We think that this is mainly caused by the accuracy of these simulators. That is, while cloudSim provides similar results to execute workloads of different sizes, simGrid clearly reflects a significant difference in the energy required to execute each

workload.

The execution time – measured in minutes – for optimising the analysed clouds is summarised in Table 10. The two first rows represent the configuration of the experiment, that is, the execution of a workload over the analysed cloud. The next two columns represent the average execution time of our proposed algorithm using cloudSim and gridSim. Similarly, the next two columns show the average execution time when the random approach is applied. These results show that our proposed algorithm requires more computational time to be executed than the random approach. Moreover, the execution time is directly proportional to the workload to be processed, being ω^l the workload that requires more time to be executed.

Configuration		EA		Random	
Cloud	Workload	cloudSim	simGrid	cloudSim	simGrid
cloudA	ω^s	92	27	6	2
	ω^m	172	36	10	3
	ω^l	236	72	14	5
cloudB	ω^s	64	27	4	2
	ω^m	200	41	11	3
	ω^l	310	71	18	5
cloudC	ω^s	92	28	5	2
	ω^m	172	39	12	4
	ω^l	236	77	21	6

Table 10: Average execution times for optimising *cloudA*, *cloudB* and *cloudC* (in minutes).

Our proposed EA scales well when the size of the executed workload grows, as the optimisation reached is more substantial when the largest workload is processed. Since the EA is able to dynamically adapt the size of the cloud – in number of physical machines – to execute a given workload, the generated cloud should be more optimised – in the sense of energy consumption – for executing this specific workload. On the contrary, the random approach only performs one modification to the cloud under test, which in some cases is not enough to reach a proper optimisation of the cloud.

After a careful analysis of the results, we can conclude that the answer to **RQ3** is *yes, our methodology is able to provide different alternatives to improve the energy consumption and automatically detect flaws in the cloud designs created by the user*.

9. Threats to validity

In this section, we discuss the threats to validity of our empirical study.

9.1. Internal threats

Internal validity concerns whether our findings, which are based on the obtained results from the empirical study, truly represent a cause-and-effect relationship. Thus, the internal validity of our study relies in the implementation of our experiments.

The design of the provided catalogue of MRs is based on the experience of two experts. We are aware that the ability of MT to detect errors in the system highly depends on the selection of metamorphic properties and, therefore, the results may have varied if different MRs were used instead. Moreover, the use of domain-specific properties, like the ones used to design our proposed catalogue of MRs, should reveal a high failure percentage.

We have implemented both the EA and the MRs in Java. Also, we use different simulators, which have been widely adopted by the research community, to analyse a wide spectrum of scenarios. We have conducted code inspection and run different tests by hand to ensure the correctness of these implementations. The source code has been checked by different individuals. The results obtained during these analyses are used to check if the MRs are fulfilled or not. Our evaluation of the MRs is based on the tests manually created by the user, that is, the source test cases. The follow-up test cases have been generated using random values and the corresponding constraints to assure the MR is fulfilled.

Other issues might arise due to the simulators used. These might have errors that can affect our findings. We have conducted experiments using different well-known simulators, which represent the behaviour of different scenarios of cloud systems to execute the tests. We mitigated this threat during the experimental phase described in Section 8, where 20000 test cases were executed and checked over our proposed MRs.

9.2. External threats

External validity concerns the extent to which the results of a study can be generalised.

We have used three cloud configurations and three different workloads, inspired by big data analysis. Although we believe that these models are representative, there is no guarantee that the obtained results and the achieved improvements in the effectiveness of the MRs are the same for other scenarios.

9.3. Constructs threats

Construct validity concerns whether the used measures are representative or not.

We measured the testing effectiveness of MRs based on the number of test cases that satisfy each MR, which is also widely used in the community. Defects in the simulators or in our proposed MRs could be a threat to construct validity, but we controlled this threat by executing a wide spectrum of test cases, using five cloud simulators to conduct our empirical study. After this experiment, we discarded three simulators because we detected some limitations, which do not properly represent the properties reflected in some MRs. Hence, we checked that the MRs were properly designed and that our implementation worked correctly.

10. Conclusions and future work

In this paper, we have proposed `MT-EA4Cloud`, a methodology that combines EAs and MT to check the correctness of energy-aware cloud systems. In essence, this methodology is based on checking the satisfiability of MRs while testing cloud systems. To that end, we have proposed a catalogue of MRs that formally model the underlying infrastructure of cloud systems focusing on energy consumption.

In order to show the applicability of `MT-EA4Cloud`, we have performed an extensive experimental study, where three cloud models are analysed using seven well-known simulators and the provided catalogue of MRs. The experimental results obtained from this study are promising, demonstrating that it is feasible to combine EAs and MT to formally test cloud computing systems. Our proposed catalogue of MRs was used to check the correctness of different well-known simulators. Since each simulator provides specific capabilities to model the different parts of the cloud, our methodology can be applied to focus on those simulators that satisfy the user requirements. We also have observed that this approach can not only be used to analyse the correctness

of simulation tools, but to discover flaws in cloud designs and to provide feasible solutions that improve these designs.

We can conclude that, during the testing process, the role of the expert is of vital importance. First, the expert is in charge of designing the MRs, providing source test cases and choosing an initial set of cloud simulators. Although there are steps in our methodology that can be automated, like the generation of a large number of follow-up test cases and the calculation of the MR effectiveness, her decisions have a direct impact of the final obtained results.

As future work, we plan to include heterogeneous cloud infrastructures, which provide machines exclusively dedicated to executing VMs (computing nodes) and machines dealing with the data accessed by the VMs (storage nodes). We are also interested in investigating the trade-off between cost and energy consumption. For this, a new EA should be designed dealing with the monetary cost of each component (e.g. CPUs, memories, networks) and to provide relevant information to the user about how the investment in better hardware impacts on the energy efficiency. Finally, we plan to study how dynamic workloads, which are generated in run-time, could be integrated into our framework. The main difficulty of this task lies in how these workloads are compared in the MRs.

Acknowledgments

This work was supported by the Spanish MINECO/FEDER projects DArDOS, FAME and MASSIVE under Grants TIN2015-65845-C3-1-R, RTI2018-093608-B-C31 and RTI2018-095255-B-I00, and the *Comunidad de Madrid* project FORTE-CM under grant S2018/TCS-4314. The first author is also supported by the Universidad Complutense de Madrid Santander Universidades grant (CT17/17-CT18/17).

References

[1] I. Flouris, N. Giatrakos, A. Deligiannakis, M. Garofalakis, M. Kamp, M. Mock, Issues in complex event processing: Status and prospects in the Big Data era, Journal of Systems and Software 127 (2017) 217 – 236.

[2] C.-F. Tsai, W.-C. Lin, S.-W. Ke, Big data mining with parallel computing: A comparison of distributed and MapReduce methodologies, Journal of Systems and Software 122 (2016) 83 – 92.

[3] K. Le, R. Bianchini, J. Zhang, Y. Jaluria, J. Meng, T. D. Nguyen, Reducing Electricity Cost Through Virtual Machine Placement in High Performance Computing Clouds, in: 24th Int. Conf. for High Performance Computing, Networking, Storage and Analysis, SC'11, ACM Press, 2011, pp. 22:1–22:12.

[4] Top500 Supercomputer sites, http://www.top500.org (April 2018) (2018).

[5] D.-M. Bui, Y. Yoon, E.-N. Huh, S. Jun, S. Lee, Energy efficiency for cloud computing system based on predictive optimization, Journal of Parallel and Distributed Computing 102 (2017) 103–114.

[6] Y. Sharma, B. Javadi, W. Si, D. Sun, Reliability and energy efficiency in cloud computing systems: Survey and taxonomy, Journal of Network and Computer Applications 74 (2016) 66–85.

[7] T. Chen, F. Kuo, H. Liu, P. Poon, D. Towey, T. Tse, Z. Zhou, Metamorphic testing: A review of challenges and opportunities, ACM Computing Surveys 51 (1) (2018) 4.

[8] S. Segura, G. Fraser, A. B. Sánchez, A. Ruiz-Cortés, A survey on metamorphic testing, IEEE Transactions on Software Engineering (in-press) PP (99) (2016) 1–1.

[9] K. A. D. Jong, Evolutionary Computation: A Unified Approach, march 25, 2016 Edition, A Bradford Book, 2016.

[10] M. Harman, Y. Jia, Y. Zhang, Achievements, open problems and challenges for search based software testing, in: IEEE 8th International Conference on Software Testing, Verification and Validation (ICST'15), 2015, pp. 1–12.

[11] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. D. Rose, R. Buyya, Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Software - Practice and Experience 41 (1) (2011) 23–50.

[12] G. Castañé, A. Núñez, P. Llopis, J. Carretero, E-mc^2: A formal framework for energy modelling in cloud computing, Simulation Modelling Practice and Theory 39 (2013) 56–75.

[13] H. Casanova, A. Legrand, M. Quinson, SimGrid: A generic framework for large-scale distributed experiments, in: 10th Int. Conf. on Computer Modeling and Simulation, UKSIM' 08, 2008, pp. 126–131.
[14] E. J. Weyuker, On testing non-testable programs, The Computer Journal 25 (4) (1982) 465–470.
[15] E. Gelenbe, Y. Caseau, The impact of information technology on energy consumption and carbon emissions, Ubiquity 2015 (June) (2015) 1:1–1:15.
[16] G. Pinto, F. Castor, Energy efficiency: a new concern for application software developers, Communications of the ACM 60 (12) (2017) 68–75.
[17] I. Manotas, C. Bird, R. Zhang, D. Shepherd, C. Jaspan, C. Sadowski, L. Pollock, J. Clause, An empirical study of practitioners' perspectives on green software engineering, in: 38th International Conference on Software Engineering (ICSE'16), ACM, 2016, pp. 237–248.
[18] G. S. Akula, A. Potluri, Heuristics for migration with consolidation of ensembles of virtual machines, in: Sixth International Conference on Communication Systems and Networks (COMSNETS'14), 2014, pp. 1–4.
[19] S. F. Smith, Is scheduling a solved problem?, in: Multidisciplinary Scheduling: Theory and Applications (MISTA'03), 2005, pp. 3–17.
[20] S. Dillon, A. Quentin, M. Ivković, R. Furbank, E. Pinkard, Photosynthetic variation and responsiveness to co2 in a widespread riparian tree, PloS one 13 (1) (2018) e0189635.
[21] A. Khosravi, R. Buyya, Energy and carbon footprint-aware management of geo-distributed cloud data centers, Advancing cloud database systems and capacity planning with dynamic applications (2017) 27.
[22] Á. L. García, E. F. del Castillo, P. O. Fernández, I. C. Plasencia, J. Marco de Lucas, Resource provisioning in Science Clouds: Requirements and challenges, Software: Practice and Experience 48 (3) (2018) 486–498.
[23] J. Wang, K. Rao, H. Ye, Application-Specific, Performance-Aware Energy OptimizationUS Patent App. 15/224,834.
[24] P. Kurp, Green computing, Commun. ACM 51 (10) (2008) 11–13.
[25] The Green Grid, http://www.thegreengrid.org/ (April 2018) (2018).
[26] The Green 500 List, http://www.green500.org (2018).
[27] T. Y. Chen, S. C. Cheung, S. M. Yiu, Metamorphic testing: a new approach for generating next test cases, Tech. Rep. HKUST-CS98-01, Department of Computer Science, Hong Kong University of Science and Technology (1998).
[28] J. Ding, D. Zhang, X. Hu, An application of metamorphic testing for testing scientific software, in: 1st International Workshop on Metamorphic Testing, ACM, 2016, pp. 37–43.
[29] H. Liu, F.-C. Kuo, D. Towey, T. Y. Chen, How effectively does metamorphic testing alleviate the oracle problem?, IEEE Transactions on Software Engineering 40 (1) (2014) 4–22.
[30] 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 (5) (2012) 755–768.
[31] I. Raïs, A. Orgerie, M. Quinson, L. Lefèvre, Quantifying the impact of shutdown techniques for energy-efficient data centers, Concurrency and Computation: Practice and Experience 30 (17).
[32] H. R. Faragardi, S. Dehnavi, T. Nolte, M. Kargahi, T. Fahringer, An energy-aware resource provisioning scheme for real-time applications in a cloud data center, Software: Practice and Experience 48 (10) (2018) 1734–1757.
[33] M. Sayadnavard, A. Haghighat, A. Rahmani, A reliable energy-aware approach for dynamic virtual machine consolidation in cloud data centers, The Journal of Supercomputing 75 (4) (2019) 2126–2147.
[34] M. Mohammadhosseini, A. Haghighat, E. Mahdipour, An efficient energy-aware method for virtual machine placement in cloud data centers using the cultural algorithm, The Journal of Supercomputing (2019) 1–30.
[35] M. Haghighi, M. Maeen, M. Haghparast, An Energy-Efficient Dynamic Resource Management Approach Based on Clustering and Meta-Heuristic Algorithms in Cloud Computing IaaS Platforms, Wireless Personal Communications 104 (4) (2019) 1367–1391.
[36] A. Merlo, M. Migliardi, L. Caviglione, A survey on energy-aware security mechanisms, Pervasive and Mobile Computing 24 (2015) 77–90.
[37] K. Sammy, R. Shengbing, C. Wilson, Energy efficient security preserving vm live migration in data centers for cloud computing, IJCSI International Journal of Computer Science Issues 9 (2) (2012) 1694–0814.
[38] V. Kharchenko, Y. Ponochovnyi, A. Boyarchuk, A. Gorbenko, Secure hybrid clouds: Analysis of configurations energy efficiency, in: International Conference on Dependability and Complex Systems, Springer, 2015, pp. 195–209.
[39] S. Segura, J. Parejo, J. Troya, A. Ruiz-Corts, Metamorphic Testing of RESTful Web APIs, IEEE Transactions on Software Engineering.
[40] M. Jiang, T. Y. Chen, F. Kuo, Z. Ding, Testing central processing unit scheduling algorithms using metamorphic testing, in: 4th IEEE International Conference on Software Engineering and Service Science, ICSESS'13, 2013, pp. 530–536.
[41] P. Rao, Z. Zheng, T. Y. Chen, N. Wang, K. Cai, Impacts of test suite's class imbalance on spectrum-based fault localization techniques, in: 13th International Conference on Quality Software (QSIC'13), 2013, pp. 260–267.
[42] X. Xie, W. E. Wong, T. Y. Chen, B. Xu, Metamorphic slice: An application in spectrum-based fault localization,

Information and Software Technology 55 (5) (2013) 866–879.
[43] M. Hutchins, H. Foster, T. Goradia, T. Ostrand, Experiments of the Effectiveness of Dataflow- and Controlflow-based Test Adequacy Criteria, in: 16th International Conference on Software Engineering (ICSE'94), 1994, pp. 191–200.
[44] V. Le, M. Afshari, Z. Su, Compiler validation via equivalence modulo inputs, in: 35th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI'14, ACM, 2014, pp. 216–226.
[45] W. K. Chan, T. Y. Chen, S. C. Cheung, T. H. Tse, Z. Zhang, Towards the testing of power-aware software applications for wireless sensor networks, in: Reliable Software Technologies – Ada Europe 2007, 2007, pp. 84–99.
[46] A. Núñez, R. M. Hierons, A methodology for validating cloud models using metamorphic testing, Annales of Telecommunications 70 (3-4) (2015) 127–135.
[47] C. Murphy, M. S. Raunak, A. King, S. Chen, C. Imbriano, G. Kaiser, I. Lee, O. Sokolsky, L. Clarke, L. Osterweil, On effective testing of health care simulation software, in: 3rd Workshop on Software Engineering in Health Care, SEHC '11, 2011, pp. 40–47.
[48] J. Ding, T. Wu, D. Wu, J. Q. Lu, X. Hu, Metamorphic testing of a monte carlo modeling program, in: 6th International Workshop on Automation of Software Test, AST '11, 2011, pp. 1–7.
[49] T. Y. Chen, F. Kuo, H. Liu, S. Wang, Conformance testing of network simulators based on metamorphic testing technique, in: 11th IFIP WG 6.1 International Conference FMOODS '09, 2009, pp. 243–248.
[50] T. Chen, F. C. Kuo, R. Merkel, W. K. Tam, Testing an open source suite for open queuing network modelling using metamorphic testing technique, in: 14th IEEE International Conference on Engineering of Complex Computer Systems, 2009, pp. 23–29.
[51] I. Zelinka, A survey on evolutionary algorithms dynamics and its complexity mutual relations, past, present and future, Swarm and Evolutionary Computation 25 (2015) 2 – 14.
[52] K. Maryam, M. Sardaraz, M. Tahir, Evolutionary algorithms in cloud computing from the perspective of energy consumption: A review, in: 14th International Conference on Emerging Technologies, ICET'18, IEEE, 2018, pp. 1–6.
[53] B. Keshanchi, A. Souri, N. J. Navimipour, An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: formal verification, simulation, and statistical testing, Journal of Systems and Software 124 (2017) 1–21.
[54] M. Vasudevan, Y.-C. Tian, M. Tang, E. Kozan, X. Zhang, Energy-efficient application assignment in profile-based data center management through a repairing genetic algorithm, Applied Soft Computing 67 (2018) 399–408.
[55] Z. Xiao, J. Jiang, Y. Zhu, Z. Ming, S. Zhong, S. Cai, A solution of dynamic vms placement problem for energy consumption optimization based on evolutionary game theory, Journal of Systems and Software 101 (C) (2015) 260–272.
[56] H. Ibrahim, R. Aburukba, K. El-Fakih, An integer linear programming model and adaptive genetic algorithm approach to minimize energy consumption of cloud computing data centers, Computers & Electrical Engineering 67 (2018) 551–565.
[57] E. Gabaldon, J. Lerida, F. Guirado, J. Planes, Blacklist muti-objective genetic algorithm for energy saving in heterogeneous environments, The Journal of Supercomputing 73 (1) (2017) 354–369.
[58] L. Zhang, Y. Wang, L. Zhu, W. Ji, Towards energy efficient cloud: An optimized ant colony model for virtual machine placement, Journal of Communications and Information Networks 1 (4) (2016) 116–132.
[59] M. Usman, A. Samad, H. Chizari, A. Aliyu, et al., Energy-Efficient virtual machine allocation technique using interior search algorithm for cloud datacenter, in: 6th ICT International Student Project Conference (ICT-ISPC), IEEE, 2017, pp. 1–4.
[60] F. Abdessamia, Y. Tai, W. Zhang, M. Shafiq, An improved particle swarm optimization for energy-efficiency virtual machine placement, in: International Conference on Cloud Computing Research and Innovation, ICCCRI'17, IEEE, 2017, pp. 7–13.
[61] M. Soltanshahi, R. Asemi, N. Shafiei, Energy-aware virtual machines allocation by krill herd algorithm in cloud data centers, Heliyon 5 (7) (2019) e02066.
[62] S. Segura, J. Troya, A. Durán, A. Ruiz-Cortés, Performance metamorphic testing: A proof of concept, Information and Software Technology 98 (2018) 1–4.
[63] J. Rounds, U. Kanewala, Systematic testing of genetic algorithms: A metamorphic testing based approach, arXiv preprint arXiv:1808.01033.
[64] D. Arora, V. G. Bassi, Generating test cases using metamorphic testing and genetic algorithm for integer bugs detection, Ph.D. thesis (2015).
[65] E. Elbeltagi, T. Hegazy, D. Grierson, Comparison among five evolutionary-based optimization algorithms, Advanced engineering informatics 19 (1) (2005) 43–53.
[66] V. Kachitvichyanukul, Comparison of three evolutionary algorithms: GA, PSO, and DE, Industrial Engineering and Management Systems 11 (3) (2012) 215–223.
[67] J. Wegener, K. Grimm, M. Grochtmann, H. Sthamer, B. Jones, Systematic testing of real-time systems, in: 4th

International Conference on Software Testing Analysis and Review (EuroSTAR 96), 1996.
[68] K. Wloch, P. Bentley, Optimising the performance of a formula one car using a genetic algorithm, in: International Conference on Parallel Problem Solving from Nature, Springer, 2004, pp. 702–711.
[69] J. Byrne, S. Svorobej, K. Giannoutakis, D. Tzovaras, P. Byrne, P. Ostberg, A. Gourinovitch, T. Lynn, A review of cloud computing simulation platforms and related environments, in: 7th International Conference on Cloud Computing and Services Science, 2017, pp. 679–691.
[70] M. Tighe, G. Keller, M. Bauer, H. Lutfiyya, DCSim: A data centre simulation tool for evaluating dynamic virtualized resource management, in: 8th International conference on network and service management, 2012, pp. 385–392.
[71] S. U. K. Dzmitry Kliazovich, Pascal Bouvry, GreenCloud: A packet-level simulator of energy-aware cloud computing data centers, The Journal of Supercomputing 62 (3) (2012) 1263–1283.
[72] A. Núñez, J. L. Vázquez-Poletti, A. C. Caminero, G. G. Castañé, J. Carretero, I. M. Llorente, iCanCloud: A flexible and scalable cloud infrastructure simulator, Journal of Grid Computing 10 (1) (2012) 185–209.
[73] G. Kecskemeti, DISSECT-CF: A simulator to foster energy-aware scheduling in infrastructure clouds, Simulation Modelling Practice and Theory 58 (2015) 188–218, special issue on Cloud Simulation.
[74] H. Ouarnoughi, J. Boukhobza, F. Singhoff, S. Rubini, Integrating i/os in cloudsim for performance and energy estimation, ACM SIGOPS Operating Systems Review 50 (1) (2017) 27–36.
[75] P. Cristian, P. Eugen, A. Marcel, C. Pop, T. Cioara, I. Anghel, I. Salomie, Coolcloudsim: Integrating cooling system models in cloudsim, in: IEEE 14th International Conference on Intelligent Computer Communication and Processing (ICCP'18), 2018, pp. 387–394.
[76] A. S. M. Rizvi, T. R. Toha, M. M. R. Lunar, M. A. Adnan, A. B. M. A. A. Islam, Cooling energy integration in simgrid, in: 2017 International Conference on Networking, Systems and Security (NSysS), 2017, pp. 132–137.
[77] N. Hassan, M. Khan, M. Rasul, Temperature monitoring and cfd analysis of data centre, Procedia Engineering 56 (2013) 551 – 559, 5th BSME International Conference on Thermal Engineering.
[78] K. Kurowski and A. Oleksiak and W. Piatek and T. Piontek and A. Przybyszewski and J. Weglarz, Dcworms - a tool for simulation of energy efficiency in distributed computing infrastructures, Simulation Modelling Practice and Theory 39 (2013) 135–151.
[79] D. Ortiz-Boyer, C. Hervás-Martínez, N. García-Pedrajas, Cixl2: a crossover operator for evolutionary algorithms based on population features, Journal of Artificial Intelligence Research 24 (2005) 1–48.
[80] A. Globus, S. Atsatt, J. Lawton, T. Wipke, Javagenes: Evolving graphs with crossover, Tech. Rep. NAS-00-006, NASA Advanced Supercomputing Division (2000).
[81] K. Park, V. S. Pai, CoMon: a mostly-scalable monitoring system for PlanetLab, ACM SIGOPS Operating Systems Review 40 (1) (2006) 65–74.
[82] W. Kolberg, P. B. Marcos, J. C. Anjos, A. Miyazaki, C. Geyer, L. Arantes, Mrsg–a mapreduce simulator over simGrid, Parallel Computing 39 (4-5) (2013) 233–244.
[83] H. R. Faragardi, M. Vahabi, H. Fotouhi, T. Nolte, T. Fahringer, An efficient placement of sinks and sdn controller nodes for optimizing the design cost of industrial iot systems, Software: Practice and Experience 48 (10) (2018) 1893–1919.