Cluster Computing (2021) 24:919–934
https://doi.org/10.1007/s10586-020-03152-9

An efficient power-aware VM allocation mechanism in cloud data centers: a micro genetic-based approach

Mehran Tarahomi[1] · Mohammad Izadi[2] · Mostafa Ghobaei-Arani[3]

Received: 7 December 2017 / Revised: 5 July 2020 / Accepted: 8 July 2020 / Published online: 9 August 2020

Abstract
Efficiency in cloud servers' power consumption is of paramount importance. Power efficiency makes the reduction in greenhouse gases establishing the concept of green computing. One of the beneficial ways is to apply power-aware methods to decide where to allocate virtual machines (VMs) in data center physical resources. Virtualization is utilized as a promising technology for power-aware VM allocation methods. Since the VM allocation is an NP-complete problem, we use of evolutionary algorithms to solve it. This paper presents an effective micro-genetic algorithm in order to choose suitable destinations between physical hosts for VMs. Our evaluations in simulation environment show that micro-genetic approach provides invaluable improvements in terms of power consumption compared with other methods.

Keywords Cloud computing · Power consumption · Micro-genetic algorithm · VM allocation

1 Introduction

Cloud computing offers a wide range of pay-as-you-go and Internet-based services in different levels such as infrastructure, platform, and Software as a Service (i.e. IaaS, PaaS, and SaaS) [1–5]. Main highlights of the cloud-based services based on Berkeley's report: "Cloud computing, the long-held dream of computing as a utility, has the potential to transform a large part of the IT industry, making software even more attractive as a service" [6]. NIST states that "cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources" [6]. Service provisioning can be easily and swiftly handled with an effective proactive management solutions. IaaS providers offer computing resources to cloud customers. Customers pay based on the capacity and time of using VM-based services to IaaS providers [7, 8].

From the other side, cloud services are becoming more and more popular daily that attracts huge attention of big IT providers such as Microsoft, Google, and IBM to run up more and more centralized data centers in different spot of the world [9]. Current cloud data center are compose of thousands of physical hosts that consume a significant amount of power. We are making more and more data centers while we do not make use of the current available cloud resources we have now. For example, most of current initiated data centers have hundreds of available hosts that are not fully utilized or idle [2]. Furthermore, more power consumption by more physical hosts will also increase in carbon dioxide (CO_2) emission. Also, the operational costs of cloud data centers are dramatically growing, mostly because of increase in power consumption [10]. Thus, the rate of power consumption by cloud data centers will directly affect cloud providers' profit [11, 12].

In this research work, we focus on live VM migration i.e. provided by virtualization technology, between hosts in cloud data centers. A promising technique to prevent hosts from overutilization is VM live migration. Furthermore,

✉ Mehran Tarahomi
tarahomi@ce.sharif.edu

Mohammad Izadi
izadi@sharif.edu

Mostafa Ghobaei-Arani
m.ghobaei@qom-iau.ac.ir

[1] Department of Computer Engineering, Kish International Campus Sharif University of Technology, Tehran, Iran

[2] Department of Computer Engineering, Sharif University of Technology, Tehran, Iran

[3] Department of Computer Engineering, Qom Branch, Islamic Azad University, Qom, Iran

VMs from underutilized hosts can be relocated and then we can switch the under-loaded host to sleep mode. This will be quite promising for power-care management of VMs in data centers [13]. The main problems in online VM consolidation problem are detection physical servers' status in the data center, identifying overloaded and under-loaded ones, adjusting overloaded servers' load by selecting a subset of their VMs for migration, and thus adjusting an optimal mapping for allocation of the migrating VMs to other servers. To do so, the rate of power needed for computation and service level agreement violation (SLAV) evaluated based on quality of services (QoS) are two important evaluation criteria. Since idle physical servers generally consume more than two third of a fully utilized server, optimality in power consumption will be achieved by maximizing physical servers' utilization and minimizing number of available under-loaded servers. Thus, the main solution is to transfer under-loaded servers' load to other available servers and then switch the under-loaded server to sleep mode for power efficiency purposes. The main benefit in reducing power consumption is to minimize operational costs and thus it will bring more profit to cloud providers [14, 15].

The main contributions of the proposed VM consolidation approach are as follows:

- We proposed a micro-genetic VM allocation algorithm for cloud data centers.
- We demonstrated the suitability and performance of the proposed algorithm in several experiments.
- We provided and assessed the micro-genetic and baseline algorithms over different scenarios.

The rest of the paper is organized as follows: we survey recent studies and main concepts of the stated problem in Sect. 2; Sect. 3 states the problem and presents the proposed approach in details; extensive analysis excepted from our simulation studies are presented in Sect. 4; finally we conclude the overall performance of the proposed solution in the paper and the future plans to further research in line with this research work are presented in Sect. 5.

2 Related works

Recently, many researchers from academia and industry tried to provide optimal allocation of VMs in the cloud data centers. However, there is a long way to reach optimality in cloud resource management. Linear programming is among most common analytical solutions for this problem [16, 17]. As a case study, Chaisiri et al. [18] provided a multi payment plan for hosting VMs in multiple cloud providers. The presented approach is effective in terms of minimizing demand costs under uncertainty. The authors proposed a linear- and quadratic-based scheduling algorithm. Speitkamp et al. formulated cloud VM allocation problem with linear programming with specific constraints [19]. The authors decrease cost of VM allocation in cloud data centers with a linear programming-based heuristic. Wu et al. modeled the problem with genetic algorithm and used genetic population in placement of VMs in data centers to improve computation and communication power consumption [20]. The authors in [21] proposed a cost-efficient heuristic for solving VM allocation problem. They demonstrated that their approach improves power consumption by 25% in comparison with multi-start random searching.

Abdel-Basset et al. [22] have studied the VM placement problem with bandwidth allocation mechanism according to the best fit policy. They have proposed an energy-efficient approach using an enhanced version of whale optimization algorithm in cloud computing environment. Further, they validate their solution using Cloudsim toolkit on the 25 various data sets with different bandwidth randomly and proved that it reduces the number of active servers compared with other meta-heuristics techniques. Abdessamia et al. [23] have proposed a new solution using binary version of gravitational search algorithm for solving the VM placement problem in the heterogeneous cloud data center. Besides, they implement their solution using MATLAB and indicated that it outperforms in terms of energy consumption compared with worst-fit, best-fit, first-fit, and particle swarm optimization mechanisms.

Parvizi and Rezvani [24] have designed a multi-objective VM placement strategy for reducing the power consumption, the number of active servers, and the total resource wastage in cloud data center. They formulated their problem as a non-linear convex optimization form and used the non-dominated sorting genetic algorithm to determine the optimal placement solution PMs on the VMs. Finally, they evaluate their strategy using Cloudsim tool in terms of energy consumption, resource loss, and the number of active servers and confirmed that it superior to compared with exact mathematical and first-fit decreasing policies. Rasouli et al. [25] have proposed a learning automata-based approach to place VMs between the physical servers in a cloud data center. Their proposed approach does not need any knowledge about cloud-based applications running on cloud servers and utilized dynamic migration and forcing idle servers to shut down. Besides, they simulate their approach using Cloudsim on the PlanetLab data set and illustrated that it significantly reduces energy consumption while satisfying QoS compared with existing mechanisms.

Azizi and Li [26] have presented a new heuristic-based algorithm that considers both resource wastage and power consumption metrics to solve VM placement problem.

Their proposed algorithm minimizes resource loss by balancing resource utilization between physical cloud servers. Besides, they proposed resource usage factor policy to solve VM placement problem using reward and penalty mechanisms. Their simulation results on the Amazon EC2 VMs workloads indicated that their proposed algorithm reduces total energy consumption resource loss of a cloud data center compared with existing algorithms and it is an interesting solution to achieve the green cloud computing. In [27, 28], a review on the multi-objective VM placement mechanisms using nature-inspired meta-heuristic algorithms is studied. They classified VM placement mechanisms into three classes: single-based, population-based, and hybrid solutions. Then, they analyzed VM placement solutions in terms of utilized optimization technique, optimization resource, placement type, and environment and provided the future research works that can be explored in VM placement area.

Ghasemi and Haghighat [29] have designed a reinforcement learning-based mechanism to handle VM placement issue using load balancing policies. Their proposed mechanism selects an action from set of the acceptable actions and performs it on the cloud environment receives a reinforcement signal according to the suitability of the VM placement solution by utilizing that action in the cloud environment. Their simulation results using Cloudsim tool demonstrated that their proposed mechanism outperforms to balance the workload in a shorter time compared with other mechanisms. Qin et al. [30] have developed a Pareto-based reinforcement learning algorithm for minimizing energy consumption in a cloud environment. They utilized the Chebyshev function and considered the weight selection issue in their proposed solution to achieve a Pareto approximation to solve VM allocation problem. Further, they validate their proposed algorithm using MATLAB and demonstrated that it is scalable for VM requests with large scale.

Wei et al. [31] have developed energy-efficient exact algorithms for solving VM allocation in cloud-based systems. They formulated the VM allocation problem in form of three-dimension bin-packing as a mixed-integer linear program and it solved by best-fit, first-fit, and greedy heuristics strategies to minimize energy consumption. Besides, they implement their proposed algorithm using Gurobi solver on the real-world cloud data centers and indicated that it has a linear consuming time and reduces the numbers of physical servers.

Abohamama and Hamouda [32] have proposed a hybrid approach using improved genetic algorithm and best fit allocation policy to reduce the energy consumption in cloud data centers. Their proposed approach handles the balancing between the exploration and exploitation and achieves the trade-off the usage of CPU, RAM and bandwidth of cloud servers to reduce the resource wastage. Reddy and Ravindranath [33] have designed an efficient VM provisioning and placement strategy for minimizing the energy consumption of cloud data centers. They utilized JAYA optimization technique to find and optimal placement solution. Further, their obtained results indicated that their proposed strategy reduces the SLA violation, energy consumption, and VM migration compared with modified best fit decreasing and particle swarm optimization mechanisms.

Most of recent studies mainly focused on single parameter to decide where to allocate each VM in cloud data center. Although single parameter like CPU utilization and single objective like power consumption makes the problem quite straight-forward to be easily modeled and solved, the real-world VM allocation problem is way to more complex and dependent on different parameters. Gao et al. [34] proposed a multi-objective algorithm for deciding where to allocate VMs in data center in order to optimize power consumption and resource utilization maxim. The authors applied a variant of ant colony system (ACS) algorithm to find solution for large-scale VM allocation problem. Experimental evaluation of simulation studies on power consumption shows suitability and applicability of the solution provided in big data centers.

According to the reviewed and summarized the VM allocation mechanisms, a side-by-side comparison of them in terms of the utilized technique, evaluation tool, performance metrics, and data center configuration as well as workload of each mechanism are shown in Table 1.

3 Proposed approach

This section focuses on the stated problem and the micro-genetic solution is presented. We first describe the problem in Sect. 3.1. Section 3.2 covers the explanation for the applied methodology, which micro-genetic algorithm. The proposed micro-genetic VM allocation algorithm will be discussed in Sect. 3.3.

3.1 Problem definition and target model

Table 2 provides all the notations used to explain the VM allocation problem. We consider, we have a cloud data center including $m * r$ racks that each contains h hosts [35]. Based on the number of hosts, racks, and modules, number of all hosts is calculated according to Eq. 1 [3]:

$$n = m \times r \times h. \tag{1}$$

Let U_k is kth cloud user who buy cloud services such that $U = \{U_1, U_2, \ldots, U_{k-1}, U_k, U_{k+1}, \ldots, U_u\}$ where $1 \leq k \leq u$. We represent a set of vectors to demonstrate VMs of

Table 1 A side-by-side comparison of the VM allocation mechanisms

Reference	Utilized technique	Evaluation tool	Performance metric	Data center	Workload
Abdessamia [22]	GSA	Simulation (MATLAB)	Number of active server, energy consumption	Synthetic (cluster of heterogeneous servers)	Artificial
Abdel-Basset [23]	WOA + Lévy flight	Simulation (Cloudsim)	Utilization, number of physical server	Synthetic (cloud user-customized VMs)	PlanetLab
Parvizi and Rezvani [24]	NSGA-III	Simulation (Cloudsim)	Execution time, utilization, resource loss, and energy consumption	Synthetic (cloud user-customized VMs)	PlanetLab
Rasouli et al. [25]	LA	Simulation (Cloudsim)	Energy consumption, Number of VM migration, SLA violation	Synthetic (Cloud user-customized VMs)	PlanetLab
Azizi and Li [26]	Heuristic-based (priority-based)	Simulation (C++)	Number of active server, energy consumption, resource wastage, CPU utilization	Synthetic, real-world (cloud user-customized VMs, Amazon EC2 Instances)	Artificial
Ghasemi and Haghighat [29]	RL	Simulation (Cloudsim)	Execution time, number of server shutdown, VM migration cost	Synthetic, real-world (Gaussian distribution, Nottingham University)	PlanetLab
Qin et al. [30]	Pareto-based RL	Simulation (MATLAB)	Energy consumption, resource wastage, computation time	Synthetic (Cloud user-customized VMs)	Artificial
Wei et al. [31]	Exact-based method	Simulation (Gurobi solver + Python)	Energy consumption, resource utilization, number of active server, computation time	Real-world (Google Data Center)	Artificial
Abohamama and Hamouda [32]	Permutation-based GA	Simulation (MATLAB)	Power consumption, resource wastage, elapsed run time	Real-world (TSP dataset)	Artificial
Reddy and Ravindranath [33]	JAYA optimization	Simulation (Cloudsim)	Energy consumption, VM migration, SLA violation	Real-world HP ProLiant ML110)	PlanetLab
Our approach	Micro genetic	(Cloudsim)	Energy consumption, VM migration, SLA violation, number of server shutdown	Real workload (PlanetLab)	PlanetLab

GSA gravitational search algorithm, *LA* learning automata, *RL* reinforcement learning, *WOA* whale optimization algorithm, *NSGA-III*, non-dominated sorting genetic algorithm, *TSP* traveling salesman problem

Table 2 Notations used for problem definition

Notation	Definition	Notation	Definition
H_i	Host i	Vrt^j	The amount of time jth VM is running
n	Total number of physical servers	Vs^j	SLA violation of jth VM
U_k	User k	$_{Vstj}$	Initiated time of jth VM
r	Number of racks	$Vrt_\tau^{j,i}$	Execution time of jth VM on host i since time slot τ
UV_k	Total number of VMs of the kth cloud customer	u	Total number of cloud customers
V_j	VM j	ℓ	Length of each time slot in minutes
Vu_τ^j	CPU usage in percentage of jth VM in time slot τ	Eh_i	Total power consumption of ith Host
Va_τ^j	Million instructions allocated from jth VM to cloud servers in time slot τ	$E_i(Hu_\tau^i)$	Power consumption of ith host in time slot τ
Hu_τ^i	CPU usage in percentage of ith host in time slot τ	F_τ^i	Number of available VMs in ith host at time slot τ
Vc_j	Total CPU capacity of jth VM	Vmt_τ^j	Minutes that jth VM is in migration during time slot τ
Hc_i	Total CPU capacity of ith host	Vdt_τ^j	Total time that jth VM experiences violation during time slot τ

all cloud users denoted by V = {V_1,V_2,…,V_u} where V_k is a vector of user k's VMs. Hence, user k has a vector of available VM denoted by $U_{Vk} = \{V_k^1, V_k^2, \ldots, V_k^l\}$.

Similar to literature studies, we assume that CPU utilization of cloud physical servers is the main factor in power consumption [36–38]. To do so, we consider a direct correlation between power usage and CPU performance of the cloud host. Accumulative resource usage of allocated VMs to a specific host is called total CPU usage of the host in MIPS. It can be divided to total CPU capacity of the host to calculate the host CPU utilization, which is calculated as follows:

$$Hu_\tau^i = \frac{\sum_{j=1}^{F_\tau^i} Va_\tau^j}{Hc_i}. \tag{2}$$

Each VM may experience a performance degradation due to resource outage in part of its execution which will cause SLA violations and VM down time (VDT). It is mainly happening due to VM live migration or resource under provisioning. Thus, total VDT (i.e. total time a VM experiences performance degradation) is calculated similar to our previous research work [3], as follows:

$$\begin{aligned} Vdt_\tau^j = {} & VM\ migration\ time + host\ overutilization\ time \\ & when\ VM\ is\ not\ in\ migration \\ & \rightarrow Vmt_\tau^j + \left\lfloor Hu_\tau^{Vh_\tau^j} / Huut \right\rfloor \cdot Vrt_\tau^{j,i}, \end{aligned} \tag{3}$$

where Vmt_τ^j denotes how many minutes the jth VM is in migration at time slot τ; $Huut$ denotes specific threshold to detect overutilized hosts; and $Vrt_\tau^{j,i}$ denotes total execution time of the jth VM when it is allocated to ith host at current time slot.

Total violation experience of the jth VM until current time slot τ can be calculated according to Eq. 4:

$$Vs^j = \frac{\sum_{\tau'=\lfloor Vst_j/\ell \rfloor}^{\tau} Vdt_{\tau'}^j}{Vrt^j/\ell}. \tag{4}$$

Here, Eq. 5 denotes calculation of total power consumed by the ith host at time slot τ of the data center.

$$Eh_\tau^i = E_i(Hu_\tau^i) \cdot \ell/60, \tag{5}$$

where minimization of power consumption and SLAV are two objectives of the problem, the objective function will be defined as follows [3]:

Minimize ϑ and δ

$$\vartheta = \sum_{\tau=1}^{x} \sum_{m=1}^{Nm_\tau} \sum_{r=1}^{Nr_\tau(m)} \sum_{h=1}^{Nh_\tau(r)} Eh_\tau^h Hs_\tau^h, \tag{6}$$

$$\delta = \sum_{\tau=1}^{x} \sum_{m=1}^{Nm_\tau} \sum_{r=1}^{Nr_\tau(m)} \sum_{h=1}^{Nh_\tau(r)} \sum_{v=1}^{F_\tau^i} Vm\_SLAV_\tau^v \tag{7}$$

Subject to:

$$0 \le Hu_\tau^i \le 100 \quad i \in \{1, 2, \ldots, n\}, \quad \tau \in \{1, 2, \ldots, x\}, \tag{8}$$

$$j \in \{1, 2, \ldots, u\}, \tag{9}$$

$$Hs_\tau^h = \begin{cases} 1, Active \\ 0, Turnedoff/slept. \end{cases} \tag{10}$$

3.2 Methodology and concepts

In this section, a micro-genetic approach is presented, which is a promising population-based meta-heuristic algorithm to guarantee good solutions with acceptable but levels of computations [39]. In this section, we will provide necessary micro-genetic concepts to demonstrate how to make of it for the stated problem.

3.2.1 Basic concepts

Genetic algorithm is an effective search method in very broad and large spaces, which ultimately leads to orientation to find a solution. Genetic algorithm works with a series of coded variables. The advantage of working with coded variable is the codes are able to convert continuous space to a discrete space. Micro-genetic algorithm is the agile version of genetic algorithm has been attempted to remove some of the genetic algorithm's steps to make this algorithm more agile. In most cases, the initial population of micro-genetic algorithm considers much less and in most standard documents, the number of initial population for the micro-genetic algorithm is 15. Algorithm 1 displays the pseudo of micro-genetic algorithm. In the following, different components of genetic algorithms are presented:

(A) *Chromosome* a genetic algorithm includes a population of chromosomes that each denotes a solution in the search space. Each chromosome includes a string of genes that each is part of a solution and it is fixed to a specific number. Binary coding is the most common representation for chromosomes, which include bit strings.

(B) *Population* a number of chromosomes constitutes together is called the genetic population. Genetic operators are used to make changes in the chromosomes in order to form new solutions.

(C) *Fitness function* genetic algorithm needs an evaluation function called fitness function to calculate the goodness of any solution found by chromosomes. Fitness function calculates a non-negative score for

any chromosomes in the population, which shows the how close the chromosome is to the optimal solution.

(D) *Coding* commonly the binary code is used, but in many cases another coding is required based on the situation of the problem. Genetic algorithm deals with the coding face of the problem's parameters or variables. One of the coding ways is the binary coding, which aims to turn the problem's solution to the string of binary digits (phase 2). The numbers of bits used for coding the variables are depend on the considered accuracy of the solution, the change range of the parameters and the relationship between the variables.

Algorithm 2 presents dynamic host management with live VM migration with live VM migration in details. Algorithm 2, line 2 shows that a number of VMs regarding cloud customers' requests are initialized and allocated to physical hosts randomly. It is then the main loop of the problem which will be repeated till there are available VMs according to customers' requests (lines 3–9). In the start of each loop, overloaded hosts shall be detected (line 4) and a subset of their allocated VMs should be chosen for migration (line 5). Selected VMs for migration shall be migrated to other normal active hosts (line 6). Furthermore, hosts based on their low load will be detected as underutilized cloud servers (line 7) and finally VMs from these detected underutilized hosts should be migrated to other places, i.e. hosts with normal CPU utilization (line 8).

Algorithm 1: Pseudo code for micro-genetic algorithm

```
/* Micro-genetic optimization algorithm*/
1: Begin
2: Initialize population
2:   While termination criteria not satisfied
4:     If converge criteria satisfied
5:       Re-initialization of offspring population
6:     End If
7:     Selection of individuals from parent population for reproduction
8:     Gene permutation of individuals to produce offspring population
9:     Evaluation of offspring population
10:    Introduction of filter offspring into parent population
11: End While
12: Return best individual
10: End
```

3.2.2 The proposed micro-genetic VM allocation algorithm

In this section, the proposed micro-genetic approach for allocation of VMs is presented in details based on the explanation provided in Sect. 3.2.1.

Algorithm 2: Proposed online management

```
/* Cloud resource management */
1: Begin
2:   Initialize VMs for customer requests and allocate them to hosts /*VM initialization*/
3:   For each time slot do
4:     Detect over-utilized hosts
5:     Choose migrating VMs from overutilized hosts
6:     Migrate all migrating VMs to other active hosts /*Algorithm 3*/
7:     Detect underutilized hosts
8:     Migrate VMs from underutilized resources to other active cloud resources /*Algorithm 3*/
9:   End For Each
10:End
```

Now, we are going to explain the proposed micro-genetic VM allocation algorithm. Chromosome, which is a fundamental part of the algorithm, is the string or the sequence of bits as the coded face of one solution in considered problem. In fact the bits of a chromosome play the role of genes in nature. Working alternately on the coding and solution space is one of the main characteristics of the genetic algorithms. The genetic operators acts on the chromosome or coding spaces, while selecting and evaluating acts on the solution space. In nature is the same way, individuals (chromosomes) are in the real non-coded space in the phenotype mode. With coding in any mechanism, genotypes mode appears.

3.2.2.1 Binary coding This conversion is a standard conversion in genetic algorithms. The binary coding is the most simple and best conversion for genetics operators. But this type of conversion in not suitable for complex issues like the problem we have here, because usually causes the chromosomes to be very large for holding solution's information. In binary conversion, the members of the population convert to string of 1s and 0s. For example, suppose algorithm finds the maximum of the function F (x, y, z).

Consider the search should be in the range 0 to 255 and positive integers. Each solution may contain three X, Y, and Z. Each number's length in the range of considered problem should be maximum of 8-bits. If each chromosome is considered as XYZ, so to cover all possible solutions, the chromosomes' length needs to be strings of bits. For this chromosome C can be as follows:

C = 11010010 11100011 00110111.

In this case, if it is necessary, the negative numbers should be searched, a bit could be added to the beginning of each string. For example if a bit is 0, the number is positive and it is 1, the number is negative.

000000001 = 1,

100000001 = − 1.

Conversion of decimal numbers can also be done with the use of such measures.

3.2.2.2 Permutation coding In this way, the chromosomes are shown as a series of natural numbers and each of these numbers is related to the special parameter in the solving problem space. The arrangement of these numbers is important and the length of the string is exactly equal with the numbers of defined parameters in the problem. The use of this type of coding is to solve travelling salesman problem. The definition is shown below.

On many issues such as the problem of "traveling salesman", we are facing with the various permutations of the set of the solutions. In this problem we have some cities, the distance between them is cleared and with starting from one and end to the same city we must:

(1) We should pass all the cities only once.
(2) The lowest distance should be passed.

The point which is important here is causes the binary coding is not a suitable method for this problem, that the cutting between two parents must be somehow that there is no repetition element. Single-point method is modified as this form, all the previous sections before the cut-off point in the first parent is copied identically to the child and the rest of the first parent genes which do not occur in the child, accordance with the coming arrangement in the second parent are certainly copied in the child.

3.2.2.3 Encoded value In this coding method, the chromosomes can choose any kind of related data in the problem in their string. This data can be real numbers, rational expressions, navigation commands, and coded data as strings of characters. In this type of coding all the cutting operator mechanisms such as binary mode is used.

3.2.2.4 Modeling and chromosome representation of the problem with micro-genetic algorithm Here, we first model allocation of VMs to cloud servers using micro-genetic algorithm. To do so, each gene is shown with a one-dimensional array and number of elements in the array equals to number of VMs in VM migration list denoted by N is shown. In the following the modeling the resource allocation problem in form of array structure, as follows:

Resource_ID=Val	**2**	**5**	**4**	**2**	**3**	**0**
VM_ID =Index	**0**	**1**	**2**	**3**	**7**	**5**

As seen, the second row represents the array index, which demonstrates the migrating VM id. The first row shows the resource in which the VM will be allocated. In other words, the ith index represents VM[i], and its value shows the physical hosts number in which the VM will be allocated. As an example, if 9th index equals to 4, it means that 9th VM should be processed on the 4th available physical host.

3.2.2.5 Description of the steps and fitness function

Step I: t = 0;
Initialize population:
For producing the initial population, every chromosome or the solution is considered as a one-dimensional array and any index on the array represents one specific virtual machine.

Step II: Termination conditions:
Algorithm execution continues until a condition occurs as follows:
1. The number of iterations reaches the maximum limit.
2. Where fitness of the best solution is not changed after a certain number of repetitions.
3. Converging the fitness of chromosomes.
In this study the first type of conditions is selected.

Step III: Fitness Function.
To determine how much a solution can offer an appropriate response, its value should be measured by a fitness function. This function gives a value to the solution based on the quality parameters in the solution. In the optimization algorithm by applying this function on all solutions, their value is calculated. And the solutions with the best value which can be based on the policy-making parameters have the maximum value, could considered as the most suitable solution. In this problem, the following parameters are used, as shown in

Current CPU utilization of cloud host $P_j$is calculated as follows:

$$\mu_j = P_j^w cpu / P_j^{cpu}. \tag{11}$$

The following equation calculates power consumed by cloud host P_j with CPU usage of μ_j is:

$$E(p_j) = k_j \cdot e_j^{max} + (1 - k_j) \cdot e_j^{max} \cdot \mu_j, \tag{12}$$

where k_j is rate of power consumption while the host is idle and e_j^{max} shows power consumption rate of the host p_j in fully utilization mode [7].

The following equation calculates the fitness function for measuring the value of each solution:

$$fitness = 1/\left(\sum_{1}^{N} E(p_j)\right). \tag{13}$$

4 Performance evaluation

In this section, experimental setup for simulation is firstly presented. Performance metrics for experimental evaluation is provided in the following section. Finally, we discuss experimental results of the proposed algorithm in comparison to three baseline algorithms.

4.1 Experimental setup

This section explains setup information for our simulation experiments such as simulation environment and its setup. The Cloudsim tool [40] as a widely used simulation tool for cloud platforms. This simulation tool is used to assess the proposed and baseline VM allocation algorithms. We consider a data center including several modules that each includes several racks with a number hosts. The details of our simulation setup are illustrated in Table 4.

We consider two different types of physical hosts from HP in our simulation. Table 5 provides details and

Table 3 The used parameters

V: VMs list	P: a list of hosts
V_i: ith VM in the VMs list	V_i^{cpu}: required CPU for ith VM
$V_i mem$: required memory for ith VM	P_j: jth host in list P
P_j^{cpu}: CPU processing capacity of P_j	P_j^{mem}: memory capacity of of P_j
$P_j^w cpu$: current computing load allocated to P_j	$P_j^w mem$: current memory load allocated to P_j
VP_j: a list of VMs allocated to the host P_j	

Table 4 Data center setup [9]

Parameter	Value
Number of modules	4
Number of racks in each modules	5
Available racks in all modules	20
List of physical hosts in a rack	48
List of physical hosts in all racks and modules	960

specifications of these two host types. Table 6 provides power consumption of these hosts over different CPU utilization in watt. We also consider four different VM types provided in Table 7. Note that power consumption level of each cloud server at time slot τ is calculated according to its CPU usage level. Each kind of cloud server has different power consumption level. In this study, we consider two different standard server configurations (i.e., HP ProLiant G4 and G5), as shown in Table 5. Besides, the power consumption level of different types of cloud servers are in detail provided in Table 6. As an example, if a HP ProLiant G5 has a CPU utilization of 10% during a time slot, the power level of that cloud server during that time slot would be 97 W per hour, as shown in Table 6. It is worth noting that the power consumption rate of cloud servers has linear relationship to the data provided in Table 6. That is, if CPU utilization of a cloud server is in between two of those utilization levels, a simple proportion between those levels shall be calculated.

To better illustrate the effectiveness of our micro-genetic algorithm, we make use of a dataset generated from real resources in the form of VMs executed in hundreds of cloud servers around the globe. The applied dataset is gathered in 10 different days and is applied for evaluation of our proposed and other baseline algorithms. The used dataset is a result of gathered information from real physical hosts in CoMon project [42]. This data set includes percentage of CPU load of many VMs in for a full daytime. Details of the used dataset are provided in Table 8.

In this experiment, we do not use real implementation of the scenarios in order to evaluate effectiveness of the solutions provided because of technical and physical resource limitations. Even though we adjust simulation setup in line with real data center environments and also we make use of real dataset from log of physical cloud servers and VMs to better understand superiority of the micro-genetic algorithm.

4.2 Metrics for performance evaluation

This section presents effective metrics for evaluation of the micro-genetic algorithm including cumulative consumption of power by physical hosts, SLAV, rate of VM migration, and finally totals times physical hosts switched to sleep.

Power consumption cumulative consumption of power by physical hosts is the first important metric of this research problem. This metric should be minimized to reach more efficiency in allocation of VMs in data centers.

Table 5 Setup specification of physical hosts [3]

Server	CPU model	Cores	Frequency (MHz)	RAM (GB)
HP ProLiant G4	Intel Xeon 3040	2	1860	4
HP ProLiant G5	Intel Xeon 3075	2	2660	4

Table 6 Power profile of the considered physical hosts in our simulation in W [3]

Server	Idle (0%)	10%	20%	30%	40%	50%	60%	70%	80%	90%	Full (100%)
HP ProLiant G4	86	89.4	92.6	96	99.5	102	106	108	112	114	117
HP ProLiant G5	93.7	97	101	105	110	116	121	125	129	133	135

Table 7 VMs specification in our simulation [41]

VM type	High-CPU medium instance	Extra large instance	Small instance	Micro instance
CPU (MIPS)	2500	2000	1000	500
RAM (MB)	870	1740	1740	613

Table 8 Workload information in details [43]

Date (working days)	Number of VMs	Means (%)	SD (%)
03/03/2011	1052	12.31	17.09
06/03/2011	898	11.44	12.83
09/03/2011	1061	10.70	15.57
22/03/2011	1516	9.26	12.78
25/03/2011	1078	10.56	14.14
03/04/2011	1463	12.39	16.55
09/04/2011	1358	11.12	15.09
11/04/2011	1233	11.56	15.07
12/04/2011	1054	11.54	15.15
20/04/2011	1033	10.43	15.21

Power consumption profile and the calculation function is provided in Sect. 3.

SLA violation since decrease in power consumption with minimizing available physical servers may cause increase in SLA violation due to performance degradation, SLAV will be the second most important metric. SLA violation (i.e. SLAV) consists of host overload violation and violation because of migration. Hence, it is the multiplication of SLA violation of hosts during overutilization (SLATAH) and violation of VMs during migration (PDM) [3]:

$$SLAV = SLATAH \cdot PDM \rightarrow \frac{\sum_{host=1}^{Numberofhosts} DownTime(host)}{\sum_{host=1}^{n} TotalRuntime(host)} \cdot \frac{\sum_{vm=1}^{numberofVMs} MigrationTime(vm)}{\sum_{vm=1}^{numberofVMs} TotalExecutionTime(vm)}. \quad (14)$$

VM migration the more live VM migration, the more SLAV due to live migration. Thus, the number of VM migration is another effective metric because less VM migration may reduce total SLAV.

Host shutdowns periodically switching on/off hosts will cause increase in power consumption, and live VM migration. It goes without saying that once we decide to turn a host off; all its VMs shall be moved to other places for load balancing purposes. Then, fewer number of server shutdown may bring better QoS.

4.3 Experimental evaluation

In this section, we analyze the performance of the micro-genetic algorithm in details. This is done with comparison by recent baseline algorithms in the form of several scenarios under metrics provided in Sect. 4.2.

Ten different executions for any experiments are done and the average results of these executions are used for evaluation in this section. To demonstrate the superiority of micro-genetic VM allocation algorithm, we provide a concrete comparison with standard genetic VM allocation algorithm [1] and a power-care algorithm for allocation with bin-packing (PABFD) [35], which is a modified version of the allocation solution called MBFD [16].

We also provide different scenarios to do a fair analysis to proof that the proposed algorithm does not just promising for a specific scenario. Table 9 provides different VM consolidation scenarios used in our experiments. In each scenario, we used different VM selection algorithm, heuristics for detection of over-utilized and underutilized algorithms.

First of all, we tune number of interaction for proposed micro-genetic VM allocation algorithm. To do so, we run different experiments for all scenarios I, II and III with different number of iterations with micro-genetic VM allocation algorithm. The average results of the achieved fitness values of each experiment over different executions and scenario are provided in Fig. 1. Vertical axis demonstrate fitness value.

At it is seen, for the first iterations the algorithm has bit improvement over iterations since the higher fitness value the better the solution is. It is clear that there is a plateau after iteration 15 for all the scenarios since there a nuance change in fitness value that is not worth the algorithm overhead. Then, we can conclude that 30 can be considered as the tuned iteration number for the problem. Thus, we will provide all the details and experiment results of the proposed algorithms with 30 iterations in this manuscript.

As mentioned, we run proposed and baseline algorithms for 10 different working days we have in the dataset. In order to remove the results variance, we execute each experiment 10 times and the median of all results are used for evaluation in Figs. 2, 3, 4 and 5.

Figure 2 illustrates total number of live VM migration for the micro-genetic and comparing algorithms in all ten days of the dataset. Figure 2 illustrates number of VM migration over different working days in our dataset for the proposed and the baseline algorithms. It is worth mentioning that each working day in our dataset shows a different experiment in our simulation evaluation. That is how we analyze the performance of our approach over 10

Table 9 Scenario description

Scenario name	Overload algorithm	Under-load algorithm	VM selection algorithm
Scenario I	LR [16]	SM [43]	MMT [3]
Scenario II	LR [16]	SM [43]	MC [43]
Scenario III	Use window moving average (WMA) [13]	SM [43]	Maximum requested resources, MRR [13]

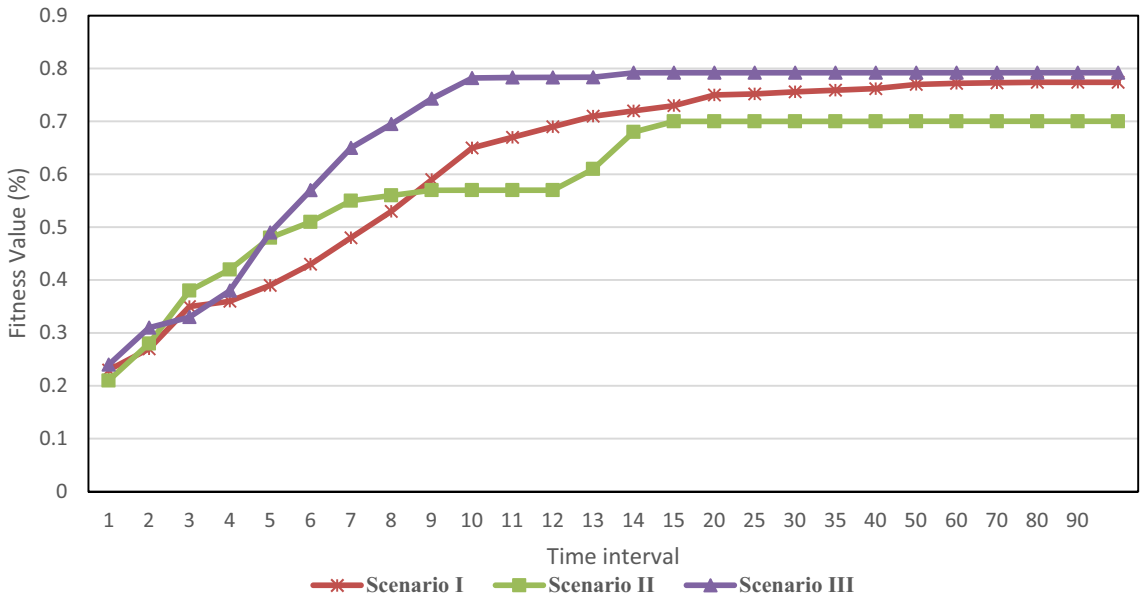

Fig. 1 Convergence speed of the proposed micro-genetic VM allocation algorithm regarding fitness function

different experiments (10 working days from 2011-03-03 to 2011-04-20). As seen the proposed micro-genetic VM allocation algorithm outperform other baseline algorithms in all working days. The main reason to this happen is that a better fitting algorithm is provided using micro-genetic that is more accurate than GA and heuristic PABFD algorithms. Better VM allocation brings better fitting and thus fewer hosts are needed to allocate VMs. Since we can fit migrating VMs to available hosts, there would not need to migrate VMs from overloaded hosts to other available VMs and thus number of VM migration is decreased.

Figure 3 illustrates the statistics regarding host shut-down in different working days and algorithms. As seen the proposed micro-genetic for allocation of VMs had better performance in comparison with other algorithms in all working days. This is because the micro-genetic VM allocation algorithm better allocate VMs to available hosts with considering SLAV as well as power consumption. Since the more allocation of the proposed algorithm will bring more accuracy, fewer hosts due to inaccurate VM allocation will be overloaded. On the other side, this algorithm may not allocate VMs on hosts that would be overloaded in near future. Thus, in such cases, the proposed algorithm will not turn off hosts, in contrast, allocate VMs to hosts with lower utilization rate. This policy will guaranty the solution from periodically switching hosts on/off. Thus, as it is seen number of host shutdown in the proposed algorithms is by far fewer than other two baseline algorithms.

Figure 4 depicts power profile of micro-genetic and comparing algorithms in different datasets. Number of VM migration will increase total computation requests of migrating VMs by 10%. Thus, increase in VM migration will indirectly increase total power consumption. Hence, there are two reasons that the micro-genetic allocation improves power consumption: (1) efficient host availability by switching unnecessary host to sleep mode; (2) the micro-genetic allocation reduced number of VM migration (which may cause increase in power consumption).

Figure 5 also depicts total SLAV of the proposed and baseline VM allocation algorithm for all working days. As it is shown, the proposed algorithm no only reduced power consumption, but it does positive effect on QoS and brings better SLAV (decrease in SLA violation).

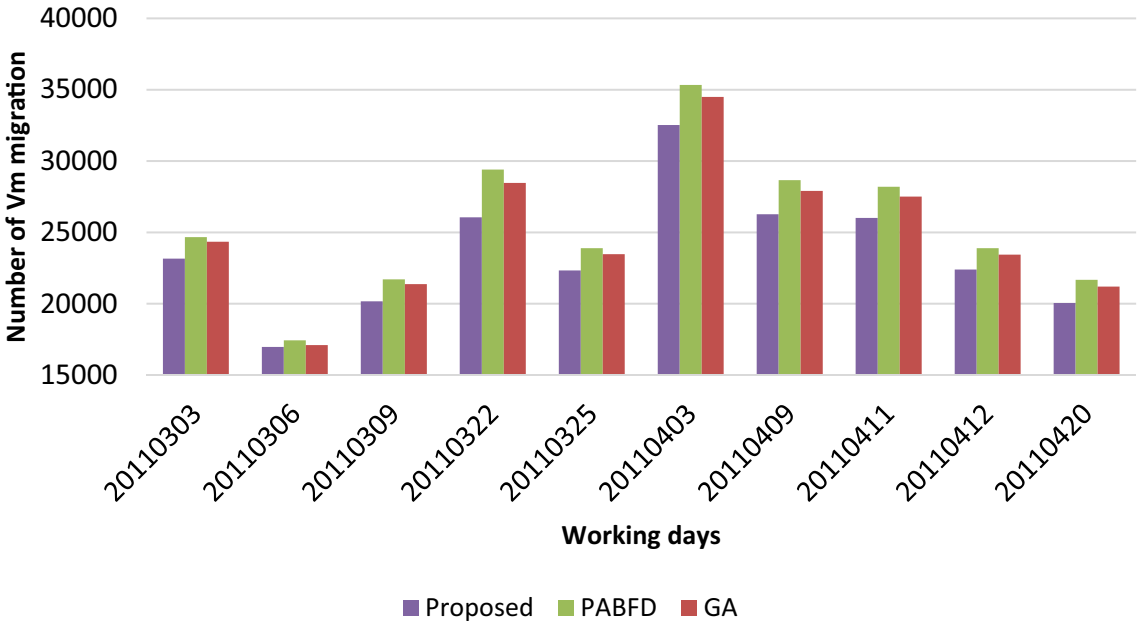

Fig. 2 Number of migrations over different algorithms and different working days in the first scenario

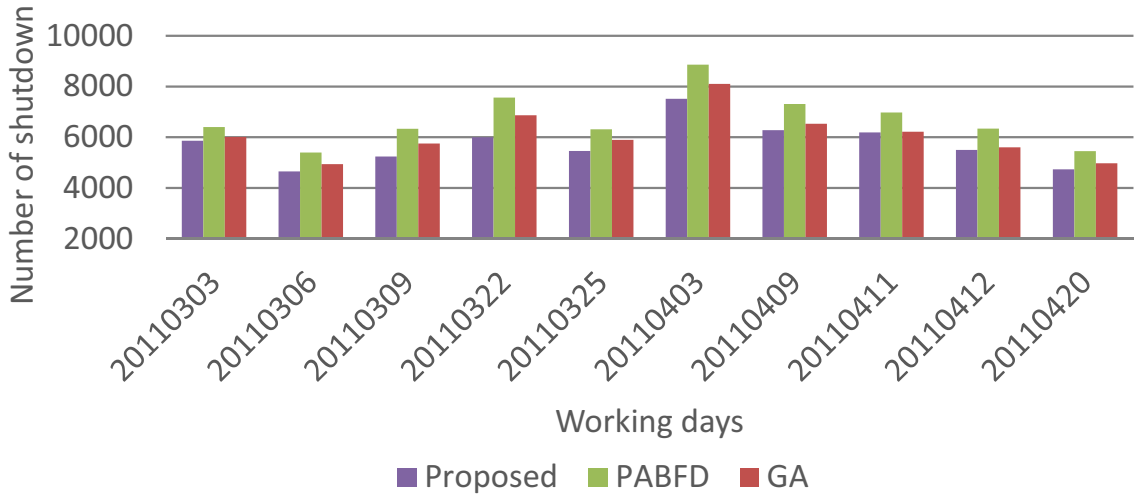

Fig. 3 Number of host shutdown of the proposed and baseline VM allocation algorithms for the first scenario

Figures 2, 3, 4 and 5 illustrate the average results of all scenarios to have a general understanding of the performance of the comparing algorithms. However, it failed to show the superiority of the micro-genetic algorithm over different scenarios. Hence, we provide different performance metrics for all scenarios in details in Table 10. As it is seen in Table 8, the proposed algorithm is by far outperform all comparing algorithms regarding all performance metrics in all scenarios. It is worth mentioning that some scenarios are easier for the proposed while the other scenarios. All in all, the micro-genetic allocation achieved the best results in all scenarios.

Figure 6 illustrates the performance differences of the proposed micro-genetic VM allocation algorithm over different scenarios. As it is shown, the first and second scenarios show way better performance specifically in power consumption and SLAV. Obviously, power consumption and SLAV rate in the third scenario are worse than the first and second scenarios and it has a linear correlation with the number of VM migration and number of shutdown.

5 Conclusion

This paper presented a power-efficient micro-genetic approach for allocation of VM in cloud data center. To do so, dynamic consolidation of cloud servers and live VM

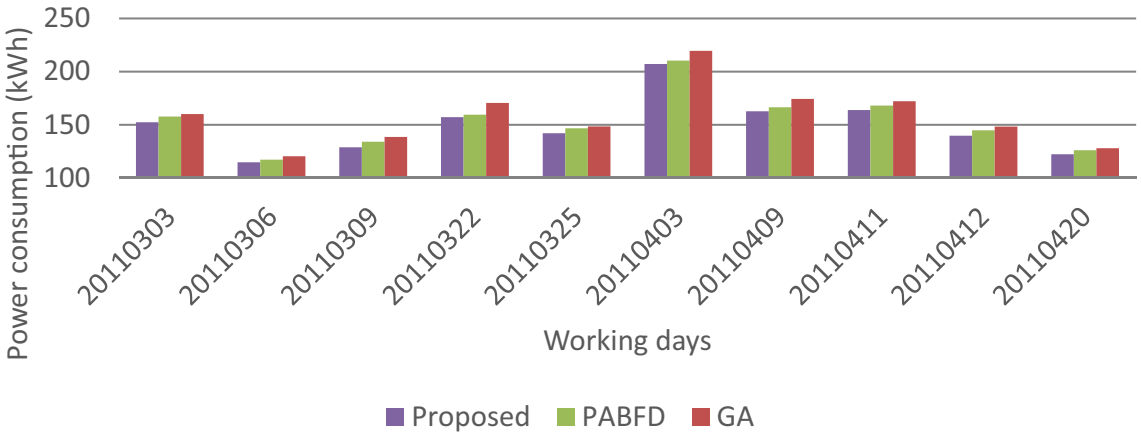

Fig. 4 Total power consumption of the proposed and baseline VM allocation algorithms for the first scenario

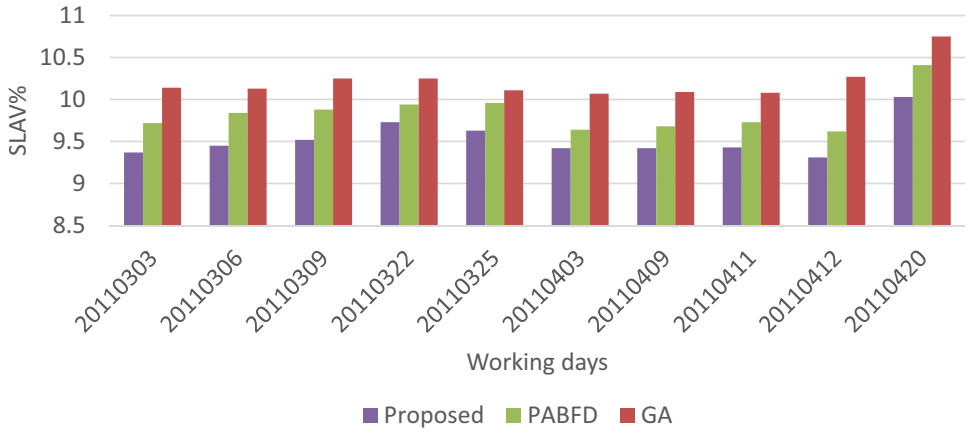

Fig. 5 SLAV rate of the proposed and baseline VM allocation algorithms for the first scenario

Table 10 Final experiment results

Scenario	Policy	Power consumption (kWh)	SLAV $\times 10^6$	VM migration	Shut down
Scenario	PABFD	158.3	9.888	29,848.7	5690
	GA	162.5	9.409	30,368.8	5601
	Proposed	150.02	9.179	28,174.7	5501
Scenario II	PABFD	165.7	9.725	26,722.8	4440
	GA	169.25	9.43	28,223.9	4310
	Proposed	158.68	9.192	26,102.3	4110
Scenario III	PABFD	171.5	11.01	23,594.1	6102
	GA	174.22	10.58	25,487.9	6001
	Proposed	163.05	10.32	24,931.5	5962

migration is applied. We discussed the importance of power consumption in cloud-based infrastructures. Unlike the reviewed studies in the paper, the proposed VM allocation algorithm could finely look into the solution space and explore and exploit for a proper mapping solution for the problem thanks to the advantages of micro-genetic convergence speed. We evaluated the proposed approach using Cloudsim and compared it with baseline algorithms (genetic and PABFD VM allocation algorithms) over different scenarios and dataset (10 working days). Since different scenarios for detection of hosts with over- or under-utilization load, and VM selection may cause different situations for the VM allocation algorithms, we have defined three different scenarios to better illustrate the superiority of the micro-genetic approach in different experiments. Experimental results using Cloudsim toolkit demonstrated that the micro-genetic algorithm improved power consumption. Our main future focus will be to provide a seamless framework for allocation of VMs in an OpenStack-based cloud data center to do unseen aspects of

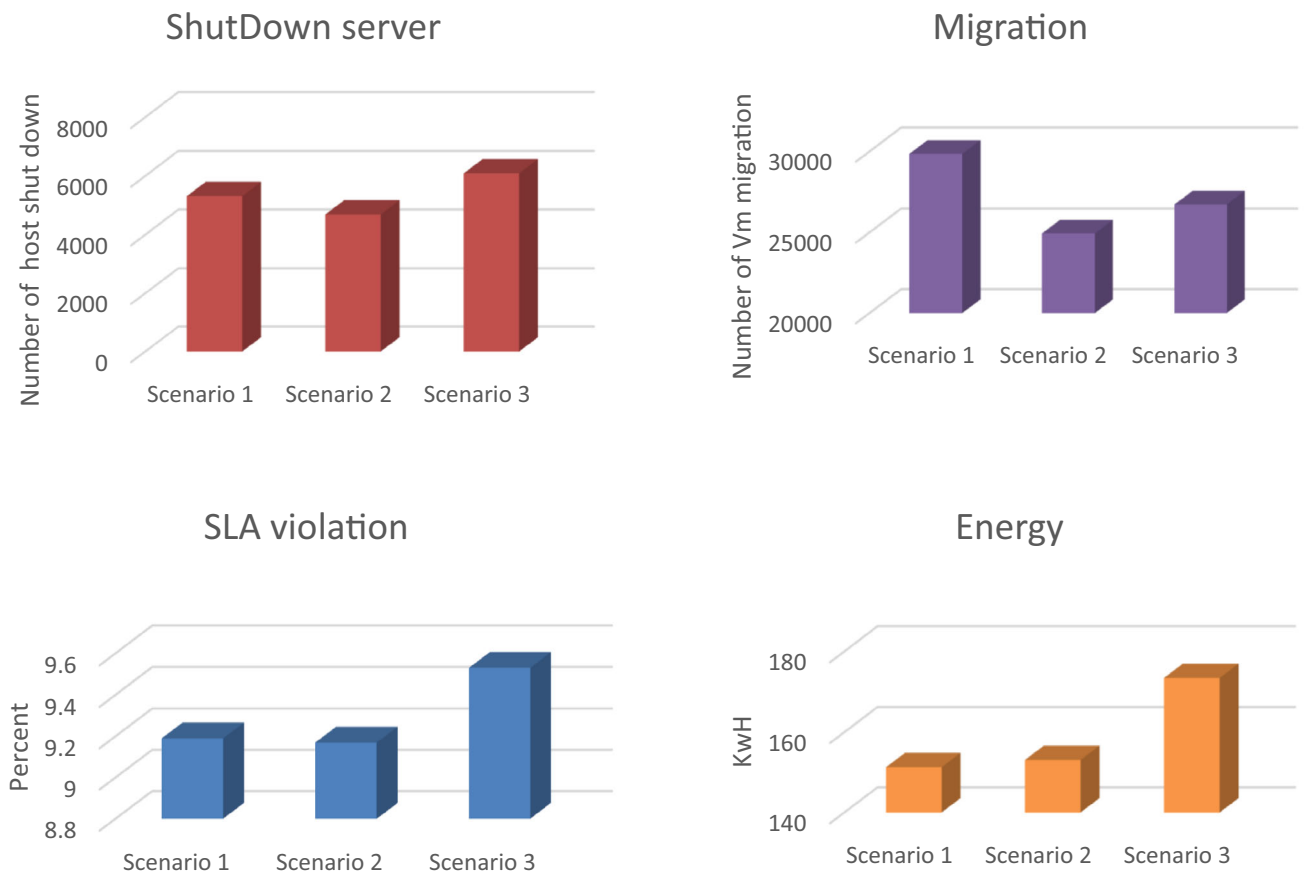

Fig. 6 Average results of all scenarios for the proposed algorithm

the real cloud data center to better understand other effective features of cloud data centers related to VM allocation problem. We also will work on data analytics to further improve resource management in cloud environments.

References

1. Kaaouache, M.A., Bouamama, S.: Solving bin packing problem with a hybrid genetic algorithm for VM placement in cloud. Procedia Comput. Sci. **60**, 1061–1069 (2015)
2. Tarahomi, M., Izadi, M.: New approach for reducing energy consumption and load balancing in data centers of cloud computing. J. Intell. Fuzzy Syst. **37**(5), 6443–6455 (2019)
3. Tarahomi, M., Izadi, M.: A prediction-based and power-aware virtual machine allocation algorithm in three-tier cloud data centers. Int. J. Commun. Syst. **32**(3), e3870 (2019)
4. Ghobaei-Arani, M., Souri, A.: LP-WSC: a linear programming approach for web service composition in geographically distributed cloud environments. J. Supercomput. **75**(5), 2603–2628 (2019)
5. Shahidinejad, A., Ghobaei-Arani, M., Esmaeili, L.: An elastic controller using Colored Petri Nets in cloud computing environment. Clust. Comput. **23**(2), 1045–1071 (2019). https://doi.org/10.1007/s10586-019-02972-8
6. Fox, A., et al.: Above the Clouds: A Berkeley View of Cloud Computing. Report UCB/EECS, vol. 28(13), p. 2009. Department of Electrical Engineering and Computer Science, University of California, Berkeley (2009)
7. Jeyarani, R., Nagaveni, N., Ram, R.V.: Design and implementation of adaptive power-aware virtual machine provisioner (APA-VMP) using swarm intelligence. Future Gener. Comput. Syst. **28**(5), 811–821 (2012)
8. Masdari, M., Nabavi, S.S., Ahmadi, V.: An overview of virtual machine placement schemes in cloud computing. J. Netw. Comput. Appl. **66**, 106–127 (2016)
9. Rahmanian, A.A., Dastghaibyfard, G.H., Tahayori, H.: Penalty-aware and cost-efficient resource management in cloud data centers. Int. J. Commun. Syst. **30**(8), e3179 (2017)
10. Zhu, X., et al.: 1000 Islands: an integrated approach to resource management for virtualized data centers. Clust. Comput. **12**(1), 45–57 (2009)
11. Rahmanian, A.A., Ghobaei-Arani, M., Tofighy, S.: A learning automata-based ensemble resource usage prediction algorithm for cloud computing environment. Future Gener. Comput. Syst. **79**, 54–71 (2018)
12. Horri, A., Rahmanian, A., Dastghaibyfard, G.H.: Energy and performance-aware virtual machine consolidation in cloud computing a two dimensional approach. Turk. J. Eng. **1**, 20–35 (2015)
13. Arianyan, E., Taheri, H., Sharifian, S.: Novel heuristics for consolidation of virtual machines in cloud data centers using multi-criteria resource management solutions. J. Supercomput. **72**(2), 688–717 (2016)

14. Dastjerdi, A.V., Buyya, R.: An autonomous time-dependent SLA negotiation strategy for cloud computing. Comput. J. **58**(11), 3202–3216 (2014)
15. Shahidinejad, A., Ghobaei-Arani, M., Masdari, M.: Resource provisioning using workload clustering in cloud computing environment: a hybrid approach. Clust. Comput (2020). https://doi.org/10.1007/s10586-020-03107-0
16. Beloglazov, A., Abawajy, J., Buyya, R.: Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Gener. Comput. Syst. **28**(5), 755–768 (2012)
17. Masdari, M., Zangakani, M.: Green cloud computing using proactive virtual machine placement: challenges and issues. J. Grid Comput. (2019). https://doi.org/10.1007/s10723-019-09489-9
18. Chaisiri, S., Lee, B.-S., Niyato, D.: Optimal virtual machine placement across multiple cloud providers. In: IEEE Asia–Pacific Services Computing Conference, 2009. APSCC 2009, pp 103–110 (2009)
19. Speitkamp, B., Bichler, M.: A mathematical programming approach for server consolidation problems in virtualized data centers. IEEE Trans. Serv. Comput. **3**(4), 266–278 (2010)
20. Wu, G., Tang, M., Tian, Y.-C., Li, W.: Energy-efficient virtual machine placement in data centers by genetic algorithm. In: Neural Information Processing, pp. 315–323 (2012)
21. Wu, Y., Tang, M., Fraser, W.: A simulated annealing algorithm for energy efficient virtual machine placement. In: 2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 1245–1250 (2012)
22. Abdel-Basset, M., Abdle-Fatah, L., Sangaiah, A.K.: An improved Lévy based whale optimization algorithm for bandwidth-efficient virtual machine placement in cloud computing environment. Clust. Comput. **22**(4), 8319–8334 (2019)
23. Abdessamia, F., Zhang, W.Z., Tian, Y.C.: Energy-efficiency virtual machine placement based on binary gravitational search algorithm. Clust. Comput. (2019). https://doi.org/10.1007/s10586-019-03021-0
24. Parvizi, E., Rezvani, M.H.: Utilization-aware energy-efficient virtual machine placement in cloud networks using NSGA-III meta-heuristic approach. Clust. Comput. (2020). https://doi.org/10.1007/s10586-020-03060-y
25. Rasouli, N., Razavi, R., Faragardi, H.R.: EPBLA: energy-efficient consolidation of virtual machines using learning automata in cloud data centers. Clust. Comput. (2020). https://doi.org/10.1007/s10586-020-03066-6
26. Azizi, S., Li, D.: An energy-efficient algorithm for virtual machine placement optimization in cloud data centers. Clust. Comput. (2020). https://doi.org/10.1007/s10586-020-03096-0
27. Masdari, M., Gharehpasha, S., Ghobaei-Arani, M., Ghasemi, V.: Bio-inspired virtual machine placement schemes in cloud computing environment: taxonomy, review, and future research directions. Clust. Comput. (2019). https://doi.org/10.1007/s10586-019-03026-9
28. Donyagard Vahed, N., Ghobaei-Arani, M., Souri, A.: Multiobjective virtual machine placement mechanisms using nature-inspired metaheuristic algorithms in cloud environments: a comprehensive review. Int. J. Commun. Syst. **32**(14), e4068 (2019)
29. Ghasemi, A., Haghighat, A.T.: A multi-objective load balancing algorithm for virtual machine placement in cloud data centers based on machine learning. Computing (2020). https://doi.org/10.1007/s00607-020-00813-w
30. Qin, Y., Wang, H., Yi, S., Li, X., Zhai, L.: Virtual machine placement based on multi-objective reinforcement learning. Appl. Intell. **50**, 1–14 (2020)
31. Wei, C., Hu, Z.H., Wang, Y.G.: Exact algorithms for energy-efficient virtual machine placement in data centers. Future Gener. Comput. Syst. **106**, 77–91 (2020)
32. Abohamama, A.S., Hamouda, E.: A hybrid energy-aware virtual machine placement algorithm for cloud environments. Expert Syst. Appl. **150**, 113306 (2020)
33. Reddy, M.A., Ravindranath, K.: Virtual machine placement using JAYA optimization algorithm. Appl. Artif. Intell. **34**(1), 31–46 (2020)
34. Gao, Y., Guan, H., Qi, Z., Hou, Y., Liu, L.: A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. J. Comput. Syst. Sci. **79**(8), 1230–1242 (2013)
35. Rahmanian, A.A., Horri, A., Dastghaibyfard, G.: Towards a hierarchical and architecture based virtual machine consolidation in cloud data centers. Int. J. Commun. Syst. **31**(4), e3490 (2017)
36. Ghobaei-Arani, M., Rahmanian, A.A., Shamsi, M., Rasouli-Kenari, A.: A learning-based approach for virtual machine placement in cloud data centers. Int. J. Commun. Syst. **31**(8), e3537 (2018)
37. Ghobaei-Arani, M., Souri, A., Baker, T., Hussien, A.: ControCity: an autonomous approach for controlling elasticity using buffer Management in Cloud Computing Environment. IEEE Access **7**, 106912–106924 (2019)
38. Ghobaei-Arani, M., Shamsi, M., Rahmanian, A.A.: An efficient approach for improving virtual machine placement in cloud computing environment. J. Exp. Theor. Artif. Intell. **29**(6), 1149–1171 (2017)
39. Ribas, P.C., Yamamoto, L., Polli, H.L., Arruda, L.V.R., Neves Jr., F.: A micro-genetic algorithm for multi-objective scheduling of a real world pipeline network. Eng. Appl. Artif. Intell. **26**(1), 302–313 (2013)
40. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. Pract. Exp. **41**(1), 23–50 (2011)
41. Agmon Ben-Yehuda, O., et al.: Deconstructing Amazon EC2 spot instance pricing. ACM Trans. Econ. Comput (TEAC) **1**(3), 1–20 (2013). https://doi.org/10.1145/2509413.2509416
42. Park, K., Pai, V.S.: CoMon: a mostly-scalable monitoring system for PlanetLab. ACM SIGOPS Oper. Syst. Rev. **40**(1), 65–74 (2006)
43. Beloglazov, A., Buyya, R.: Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers. Concurr. Comput. Pract. Exp. **24**(13), 1397–1420 (2012)

Mehran Tarahomi received accordingly his bachelor in Computer Engineering and Master of Software Engineering at Years of 2002 and 2005 from Central Branch of Azad University in Tehran and Shiraz University, respectively. He received the PhD degree in Computer Engineering at Kish International Campus, Sharif University of Technology, Tehran, Iran. His research interests include software engineering, distributed computing, computer network and energy-aware techniques specifically in the area of cloud computing.

Mohammad Izadi is an Assistant Professor and the Vice-Chairman of Educational Affairs in the Department of Computer Engineering at Sharif University of Technology, Tehran, Iran. He received a PhD Degree in Computer Science from Leiden University, the Netherlands and another PhD Degree in Software Engineering from Sharif University of Technology, respectively in 2011 and 2008. He also has a Master Degree in Philosophy of Science and another one in Computer Engineering and his BSc Degree all from Sharif University of Technology. His fields of research include distributed models and algorithms, game theory, verification of component based computing systems, computational complexity, and mathematical logic.

Mostafa Ghobaei-Arani received the PhD Degree in Software Engineering from Islamic Azad University, Science and Research Branch, Tehran, Iran. He is Assistant Professor of Computer Engineering Department, Qom Branch, Islamic Azad University, Qom, Iran. He has published more than 50 journal and conference papers in the area of distributed computing. His research interests include distributed computing, cloud computing, autonomic computing, edge/fog computing, exascale computing, soft computing, and the IoT.