Variable Neighborhood Search-Based Symbiotic Organisms Search Algorithm for Energy-Efficient Scheduling of Virtual Machine in Cloud Data Center

Mohammed Abdullahi, Shafi'i Muhammad Abdulhamid, Salihu Idi Dishing and Mohammed Joda Usman

Abstract The quest for energy-efficient virtual machine placement algorithms has attracted significant attention of researchers in the cloud computing platform. This paper applied a novel symbiotic organisms search (SOS) algorithm to minimize the number of active server by consolidation VMs on few servers for energy savings. SOS algorithm was inspired by symbiotic relationship exhibit by organisms in an ecosystem to boost their chances of survival. Essentially, SOS mimics mutualism, commensalism, and parasitism forms of relationship for traversing the search space. Hybridized with variable neighborhood search, the hybrid algorithm is termed SOS-VNS. SOS-VNS algorithm is efficient in minimizing energy consumption and improving resource utilization. The SOS-VNS algorithm is applied to various workload instances with varying number of VMs in a simulated IaaS cloud. The results obtained showed that SOS-VNS outperforms the heuristics and achieved reasonable energy savings while improving resource utilization.

Keywords Energy efficiency · Cloud computing · Virtual machine placement · Symbiotic organisms search

M. Abdullahi · S. I. Dishing
Department of Computer Science, Ahmadu Bello University, Zaria, Nigeria
e-mail: abdullahilwafu@abu.edu.ng

S. I. Dishing
e-mail: sidishing@abu.edu.ng

S. M. Abdulhamid (✉)
Department of Cyber Security Science, Federal University of Technology Minna, Minna, Niger, Nigeria
e-mail: shafii.abdulhamid@futminna.edu.ng

M. J. Usman
Department of Mathematics, Bauchi State University Gadau, Gadau PMB 068, Bauchi, Nigeria
e-mail: umjoda@gmail.com

T. Herawan et al. (eds.), *Advances on Computational Intelligence in Energy*,
Green Energy and Technology, https://doi.org/10.1007/978-3-319-69889-2_5

1 Introduction

Cloud computing is a large-scale distributed computing system which presents virtualized computing resources that are dynamically controlled, supervised, sustained, and administered using market principles [1, 2]. It is a subscription-based computing facility, which makes available an appropriate platform for user applications because of its features like application scalability, heterogeneous resources, dynamic resource provisioning, and pay-as-you-go cost model.

Cloud computing technology is rapidly emerging as the de facto paradigm of cybercomputing, storage space and host virtual infrastructures, platforms, and software both in the industries and academia. The vast scalability prospects presented by cloud infrastructures can be simply exploited not just for services and applications hosting but also as an on-demand computing paradigm [3, 4]. Modern cloud data centers consume a huge sum of energy, and the expended energy largely comes from the conventional energy that is produced using fossil fuels. The consequence of this is the high cost of electricity that is being accumulated by the day, and in addition, it leads to high carbon radiations and the large quantity of pollution [5]. With the speedy growth in cloud computing technology, minimizing energy consumption in addition to sustaining high-level computation capacity has become a timely and vital issue in such an environment. Current virtual machines (VMs) scheduling systems have primarily concentrated on improving the resource utilization and minimizing energy usage by improving some of the classical optimization schemes. Nonetheless, many large resource demanding cloud systems are executing VMs in a very realistic situation to have substantial effects on the organization performance and energy consumption. Likewise, sudden peak loads could result in serious scheduling error that can considerably hinder the energy efficiency of scheduling schemes [6].

The approaches that shut down VMs that are not in use can have an undesirable influence in relation to performance of the entire scheme. The idle VMs would not be able to perform using large workloads in a short period. Thus, the optimum energy-aware schemes are required to assure a proper level of minimization of energy consumption. To assess these approaches and to measure the level of impact of performance and energy consumption, a very reliable computational intelligence (CI) technique needed. Furthermore, the selected CI technique has to be able to work with the current conditions of cloud data centers. CI algorithms are inspired by natural process, animals' behavior, or sports to optimize complex real-world problems or systems. CIs as used in cloud scheduling for energy minimization in cloud data centers can be classified into population-based like genetic algorithms [7], ant colony algorithm [8], symbiotic organisms search [9], and particle swarm optimization [10]; and trajectory-based like the simulated annealing [11].

Researchers have employed different metaheuristic algorithms [5, 12–14] (such as PSO, GA, ABC, and ACO) to solve energy-efficient optimization problems in cloud. The results obtained by these algorithms are superior to other heuristic algorithms. However, these metaheuristic algorithms still suffer from the shortcoming of being time-consuming or inefficient in optimizing energy consumption and resource

utilization. Therefore, more effective alternative metaheuristic algorithms need to be developed to improve the optimization of energy-efficient parameters. SOS algorithm is a metaheuristic optimization algorithm based on symbiotic association of organisms in an ecosystem, it was first introduced in [15], and being widely applied to optimization problems in various domains such as economic dispatch [16, 17], power optimization [18, 19], construction project scheduling [20], task scheduling [9, 21, 22], design optimization of engineering structures [23, 24], wireless communication [25], and machine learning [26, 27]. SOS is influenced by random oscillation effect in later evolution, which makes it to be easily trapped in local minima, thereby making the convergence rate very slow. The VNS is a robust local search procedure that systematically uses the concept of neighborhood change for search space exploitation and avoidance of likely entrapment in local minima [28, 29]. SOS-VNS combines SOS and VNS to enhance the ability of the proposed algorithm to jump out of likely local maxima, and speed up convergence rate.

In this chapter, we present a variable neighborhood search (VNS)-based symbiotic organisms search (SOS) algorithm for energy-efficient scheduling of VM in cloud data center. The main contributions of this chapter are to:

1. Formulate an energy-efficient VM scheduling optimization technique model for minimizing energy in a cloud data center.
2. Design a variable neighborhood search-based symbiotic organisms search algorithm for energy-aware scheduling in the cloud data center.
3. Evaluate of the proposed technique using standard energy efficiency performance metrics for VM scheduling.

The remaining parts of the chapter are organized as follows. In Sect. 2, literature related to cloud scheduling and energy awareness of VM management in cloud data centers is presented. Section 3 chronicles the formulation and design of energy-efficient VM scheduling optimization using VNS and SOS algorithms. In Sect. 4, the performance metrics were used to comparatively evaluate the proposed technique, and Sect. 5 presents conclusion and future works.

2 Related Works

Computing and VM management in data centers with different features to achieve a particular goal, using computations intelligence algorithms. These include genetic algorithm [30, 31], ant colony optimization [32], particle swamp optimization [33], SOS [9, 21, 22], and BAT [34]. Scheduling of VMs to different virtual resources has a substantial effect on both energy consumption and resource utilization in the cloud data center [35, 36]. Conversely, it is vital to develop an energy-aware VM scheduling technique that retains a balance between energy efficiency and VM utilization in this environment. Ibrahim et al. [12] present an integer linear programming (ILP) algorithm, which reduces the energy used in a cloud computing data centers. In addition, an adaptive genetic algorithm (AGA) was put forward for the reflection of the

dynamicity of the data centers and to deliver an optimal scheduling result that reduces the energy usage in the cloud network. Experimental results show that the ILP-AGA method used performs comparatively well in terms of response time and energy minimization. However, the ILP-AGA technique did not consider of task preemption in the process of dynamic scheduling decisions. Similarly, a repairing genetic algorithm (RGA) was developed to address the problem of the large-scale optimization problem in cloud data centers for energy efficiency. It was designed to improve the penalty-based GA scheme by integrating it with Longest Cloudlet Fastest Processor (LCFP) algorithm, through which a preliminary population size was produced with an infeasible solution repairing procedure (ISRP). The VM scheduling with RGA was incorporated into a three-stage energy-aware methodology for cloud data centers. The simulation results show that the proposed framework produces minimization of 23% of energy usage and 43% increase in resource utilization as compared with the steady-state GA in the simulated case study [37]. However, the proposed framework did not consider multi-objectivity and instances in a heterogeneous distributed cloud data centers.

A Euclidean distance founded multi-objective resource scheduling algorithms for VMs and a migration strategy in the cloud data center for energy efficiency was proposed. In addition, the sharing of VMs to physical machines (PMs) was done using the proposed hybrid method of GA and PSO called the HGAPSO [14]. The method, HGAPSO, uses VM scheduling and migration to minimize energy usage and consumption of cloud resources but also it avoids SLA violation in the cloud environment. In order to evaluate the performance of the HGAPSO technique and VM migration policy in relation to energy usage, VM utilization, and SLA violation, an experiment was conducted in both heterogeneous and homogeneous cloud computing systems. The investigation outcomes show the supremacy of HGAPSO and the VM migration strategy over legacy algorithm in terms of energy minimization and best resources utilization strategy. However, the HGAPSO did not consider the effect of computational complexity on the makespan time. Duan et al. [6] present a novel scheduling method termed PreAntPolicy, which comprises of a prediction model using fractal mathematics with scheduling algorithm by improving the ACO algorithm. It uses the proposed model to generate the implementation of the scheduling algorithm through the load tendency estimate. Also, the scheduler is accountable for VM scheduling while reducing the energy usage to guarantee the quality of service (QoS). Experimental results show that using real workload traces gathered from the compute clusters of Google, the PreAntPolicy method demonstrations good energy minimization and resource usage. Furthermore, the method provides an effective dynamic capability scheduling technique for resource-intensive tasks in a distributed environment and may minimize the usage of VMs and energy when scheduling is started by prompt peak loads. However, the computation intelligence algorithm used for optimization is inherently deficient in local entrapment avoidance during scheduling.

Fernandez-Caro et al. [38] developed a tool called SCORE, which was design to simulate energy-aware huge and parallel scheduling techniques and for the implementation of varied and artificial workloads. The experimental outcome shows that

the SCORE performance well in terms of energy awareness, security, and scheduling policies in cloud data centers. However, the SCORE simulations need to be compared with real-world data and emulators' results. Similarly, another study by Luo et al. [39] concentrates on the IaaS cloud model, where convention VMs were ran in applicable servers accessible in cloud environment. The research work presented a cloud data center resource scheduling algorithm. The system was designed to deliver QoS (via SLAs) and also reduces energy usage and green computing objectives. Taking into account that the cloud data center host is regularly in thousands of magnitude and that using an exact procedure to address the resource scheduling problem is difficult. The modified shuffled frog leaping algorithm (MSFLA) and enhanced extremal optimization are deployed to resolve the dynamicity of scheduling problem of cloud resources. Investigative outcomes show that the MSFLA system shows good performance in the cloud computing environment. However, the MSFLA scheduling method is relatively weak in energy management in heterogeneous cloud data centers. A computational intelligence algorithm is used for VM placement to reduce the amount of active physical servers running, in order to allocate underutilized servers to minimize energy lost. Encouraged by the performance of the ACO algorithm for undeterministic social problems, an ACO-based method is advanced to accomplish the VM placement objective. In addition to order exchange and migration (OEM) local search procedure, the hybridized system is called an OEMACS. It minimizes the number of active servers used for the scheduling of VMs from exploration viewpoint through an innovative approach for pheromone deposition that guides the artificial ants to encouraging results that cluster contestant VMs in the same place. The proposed hybrid scheme is used in a diverse VM placement with divergent sizes in cloud data centers of different cloud servers. Experimental outcome indicates that the OEMACS largely overtakes some computation intelligence algorithms and other evolutionary algorithms, particularly on VM placement with logjam resource features, and presents substantial savings of energy and more efficient usage of diverse VMs [13]. However, the OEMACS algorithm performs better in a homogeneous cloud data center environment than in a heterogeneous cloud data center environment. Likewise, another similar research presented a multi-objective energy-aware VM scheduling problem for cloud data centers within the concept of green computing. This is partly driven by renewable energy computing strategy, where the computing VMs of the cloud data centers are based on DVFS. Therefore, an improved multi-objective computational intelligence algorithm called OL-PICEA-g was put forward to address the issue. The PICEA-g system with the general opposition-based learning was used for searching the appropriate computing VM in the cloud data center. Experimental results of the OL-PICEA-g system were evaluated with the PICEA-g method. The outcome shows the dominance and efficiency of the OL-PICEA-g algorithm over PICEA-g scheme [5]. However, the OL-PICEA-g algorithm cannot handle the energy-aware scheduling of tasks and VMs on a data center on batteries. Therefore, our proposed variable neighborhood search-based symbiotic organisms search algorithm for energy-aware scheduling of VMs in the cloud data center environment is put forward to address this problem.

3 Energy-Efficient Virtual Machine Scheduling Optimization

In cloud computing, users' applications requirements are stated in the form of VMs features (CPU, memory, and storage) and operating system for executing user application. Then, the VMs that meet the user application requirements are assigned to a server based on a placement strategy. The choice of a suitable server for VMs assignment while minimizing energy consumption is a challenging problem. This paper studies the VM placement problem for optimizing the number of running servers where VMs are servers are balanced and consolidated based on CPU, memory, and storage constraints.

3.1 Problem Definition

Suppose there are m VMs and n servers, with $V = \{V_1, V_2, V_3, \ldots, V_m\}$ and $H = \{H_1, H_2, H_3, \ldots, H_n\}$ as sets of VMs and servers, respectively. Also, let cV_j, mV_j, and sV_j be the CPU, memory, and storage requirements of a VM $V_j \in V$, respectively. In a similar manner, let cH_j, mH_j, and sH_j be the CPU, memory, and storage capacities of server $H_j \in H$, respectively. The objective is to obtain a VM assignment schedule that will reduce the energy consumption and improve the resource utilization. It is assumed that a server has the capacity to meet the resource needs of a VM. That is, in a VM assignment schedule, a VM can only be mapped to one and only one server which is a kind of zero-one adjacency matrix P, where its element p_{ij} indicates the VM V_j is mapped to a server H_i. If V_j is mapped to H_i then $p_{ij} = 1$, otherwise $p_{ij} = 0$. Each server H_i must be able to meet the resource requirements of all the VMs assigned to it.

The VM placement problem for reducing the number of active servers can be formally written as:

$$\text{minimize } f(p) = \sum_{i=1}^{n} q_i \tag{1}$$

$$\text{subject to } p_{ij} \begin{cases} 1; \text{ if } V_j \text{ is assigned to } H_j \\ 0; \text{ otherwise} \end{cases} \tag{2}$$

$$p_{ij} \begin{cases} 1; \text{ if } \sum_{j=1}^{m} p_{ij} \geq V_j \text{ for all } V_j \in V \text{ is assigned to } H_j \\ 0; \text{ otherwise} \end{cases} \tag{3}$$

$$\sum_{i=1}^{n} p_{ij} = 1; \text{ for } V_j \in V \tag{4}$$

$$\sum_{j=1}^{m} cV_j p_{ij} \leq cH_i q_i \tag{5}$$

$$\sum_{j=1}^{m} mV_j p_{ij} \leq mH_i q_i \tag{6}$$

$$\sum_{j=1}^{m} sV_j p_{ij} \leq sH_i q_i \tag{7}$$

The constraint 3 indicates that a server H_i is selected when $q_i = 1$ while constraint 4 ensures that a VM is assigned to one and only one server. Constraints 5, 6, and 7 make sure that a server assigned to a VM must be able to satisfy its requirements. The power consumption model of a server is linearly dependent on its CPU utilization [40]. An active server in an idle state consumes about 50–70% of its power consumption when active at full load [41]. The power model is defined in Eq. 8.

$$P(\alpha) = \beta P_{\max} + \alpha(1 - \beta) P_{\max} \tag{8}$$

where $P_{\max}$ is the power consumption at full load, $\alpha \in [0,1]$ is the CPU utilization, β is the fraction of power consumed in an idle state. Since a reasonable amount of energy is wasted when a server is in an idle state, minimization of the number of active servers will amount to significant energy savings. During the experiment, the power consumption of a serve will be estimated using Eq. 8.

3.2 Basic Concepts of Symbiotic Organisms Search

SOS is a new and promising metaheuristic algorithm inspired by forms of interaction adopted by organisms for their survival in the ecosystem. The algorithm simulates mutualism, commensalism, and parasitism forms of interaction to evolve candidate solutions. Mutualism interaction involves two organisms which cohabit together for the benefit of each other, none of the organisms loose in the interaction. In commensalism interaction, one of the pair of the organisms involved in the interaction benefits while another organism neither loses nor gained from the relationship. In the parasitism relationship, one organism benefits from the relationship while the other is harmed.

In SOS algorithm, potential solutions are represented by a population of organisms which evolved through successive iterations. An initial ecosystem (population) of organisms (candidate solutions) with *ecosize* (number of organisms in the ecosystem) is generated as $E = \{X_1, X_2, X_3, \ldots, X_{\text{ecosize}}\}$. The position of organism i is denoted as $X_i = [x_{i1}, x_{i2}, x_{i3}, \ldots, x_{id}]^T$ for optimizing a d–dimensional problem. The positions of each organism in the ecosystem are updated using mutualism, commensalism,

and parasitism phases, respectively. The following sections describe the mutualism, commensalism, and parasitism phases of SOS algorithms.

Mutualism Phase In this phase, an organism X_i randomly selects an organism X_j ($i \neq j$) for mutual interaction to improve the survival of both X_i and X_j. The new organisms are obtained according to Eqs. 9 and 10. MV is the mutual relationship vector between X_i and X_j as defined in Eq. 11. X_{best} represents the organism with best fitness value. B_1 and B_2 represent the benefit factors between organism X_i and X_j. B_1 and B_2 stochastically determined are either 1 or 2. The values 1 and 2 denote light and heavy benefits, respectively. The fitness value of the new organisms $f\left(x_i^{\text{new}}\right)$ and $f\left(x_j^{\text{new}}\right)$ are evaluated, then X_i and X_j are updated to x_i^{new} and x_j^{new}, respectively, if the fitness of the new organisms are better as represented in Eqs. 12 and 13, respectively.

$$X_{i,\text{new}} = X_i + R_1(0, 1-\beta) * (X_{\text{best}} + \text{MV} * \beta_1) \tag{9}$$

$$X_{j,\text{new}} = X_j + R_2(0, 1-\beta) * (X_{\text{best}} + \text{MV} * \beta_2) \tag{10}$$

$$\text{MV} = \frac{1}{2}(x_i + x_j) \tag{11}$$

$$X = \begin{cases} X_i^{\text{new}} & \text{if } f\left(X_i^{\text{new}}\right) > f(X_i) \\ X_i & \text{if } f\left(X_i^{\text{new}}\right) \leq f(X_i) \end{cases} \tag{12}$$

$$X = \begin{cases} X_j^{\text{new}} & \text{if } f\left(X_j^{\text{new}}\right) > f(X_j) \\ X_j & \text{if } f\left(X_j^{\text{new}}\right) \leq f(X_j) \end{cases} \tag{13}$$

where $R_1(0,1)$ and $R_2(0,1)$ are vectors of random numbers in the range 0–1; $f(.)$ is the fitness function.

Commensalism Phase In commensalism phase, an organism X_i interacts with a random organism x_j ($j \neq i$) for the improvement of survival of X_i. The new organism is obtained using Eq. 14; X_{best} is the fittest organism. X_i is updated to X_i^{new}, if $f\left(X_i^{\text{new}}\right)$ is better than that of $f(X_i)$ according to the relation in Eq. 15.

$$X_i^{\text{new}} = X_i + R(-1, 1) * (X_{\text{best}} + X_j) \tag{14}$$

$$X_i = \begin{cases} X_i^{\text{new}} & \text{if } f\left(X_i^{\text{new}}\right) > f(X_i) \\ X_i & \text{if } f\left(X_i^{\text{new}}\right) \leq f(X_i) \end{cases} \tag{15}$$

where $R(-1,1)$ is a vector of random numbers between -1 and 1.

Parasitism Phase In parasitism phase, an organism X_i is utilized to create an artificial parasite called parasite vector by mutating X_i using uniformly generated random number. The parasite vector is evaluated against a randomly selected organism X_j,

and the parasite vector replaces X_j if the parasite vector is fitter. The relationship in Eq. 16

$$X_j = \begin{cases} \text{PV if } f(\text{PV}) > f(X_j) \\ X_j \text{ if } f(\text{PV}) \leq f(X_j) \end{cases} \tag{16}$$

where PV is the parasite vector.

Algorithm 1 *Symbiotic Organisms Search Algorithm [11]*

Input: Set $eco-size$, initialize $X = \{X_i | i = 1, 2, 3, ..., ecosize\}$, termination condition,

Output: Optimal schedule

Identify X_{best}
while termination not met **do**
  for $i = 1$ to $eco - size$ **do**
    Mutualism Phase
    $MV = \frac{X_i + X_j}{2}$ ▷ $(j \neq i)$
    $X_i^{new} = X_i + R_1(0,1) * (X_{best} - MV * B_1)$ ▷ (B_1, B_2) : benefit factors
    $X_j^{new} = X_j + R_2(0,1) * (X_{best} - MV * B_2)$
    if $F(X_i^{new}) < F(X_i)$ **then**
      $X_i = X_i^{new}$
    end if
    if $F(X_j^{new}) < F(X_j)$ **then**
      $X_j = X_j^{new}$
    end if
    Commensalism Phase
    $X_i^{new} = X_i + R(-1,1) * (X_{best} + X_j)$
    if $F(X_i^{new}) < F(X_i)$ **then**
      $X_i = X_i^{new}$
    end if
    Parasitism Phase
    Create $parasite\_vector$
    if $F(parasite\_vector) < F(X_j)$ **then**
      $X_j = parasite_vector$
    end if
    Identify X_{best}
  end for
end while

Solution Encoding The meaning and dimension of an organism determine the encoding for the problem at hand. In the proposed algorithm, each organism is an individual in the ecosystem that represents a part of the solution search space. To define the solution representation for the problem, each organism represents a complete VM allocation; thus, the dimension of an organism is the same as the number of VMs. The real values are used to represent the servers to be selected. The coordinate system for determining the position of an organism in the solution search space is dependent on the dimension of the organism. The organism is a five-dimensional one and its position on the search space

is defined by coordinates 1 through 5. For instance, an organism which encodes an allocation with five VMs and three servers can have a solution $S = \{(1), (1,5), (2,3), (3,4), (2,3)\}$; the first element of the ordered pair represents the index of server and the second element of the ordered pair represents the index of the VM. For instance, the ordered (1) indicates that VM 1 is allocated to server 1, the ordered pair (2,3) indicates that VM 2 is allocated to server 3.

Initialization Uniformly generated random numbers are used to initialize the ecosystem (population) which also serves as the source of the randomness for updating the positions of the organisms during the searching procedure.

Algorithm 2 *Individual Organism Encoding*

Output: An individual $X = (x, s)$ ▷ $x = \{x_1, x_2, x_3, ..., x_n\}$; n is the number of tasks to be scheduled.

1: Initialize a vector y_0 ▷ $y = \{y_1, y_2, y_3, ..., y_n\}; y_i \in (0, 1)$

2: Generate uniformly distributed random sequence y.

3: Transform the random sequence into the range of parameters of VM allocation model according to: $x_i = x_{max} + (x_{max} - x_{min}) \times y_i$; $i = 1, 2, 3, ..., n$

SOS Operators for Position Update The candidate solutions are represented by ecosystem (population) of organisms, while mutualism, commensalism, and parasitism operators to direct the search process by candidate solutions. Each organism is represented by a coordinate system in the search space, and organisms keep an update of global best position X_{best} which is determined based on the fitness function of the problem at hand. The fitter organisms are allowed to proceed to the next generation of potential solution, while the unfitted organisms are discarded. The fitter organisms are those with good solution, while the unfitted organisms hold bad solution. The positions of the organisms are then updated toward the X_{best} locations using mutualism, commensalism, and parasitism phases, respectively. The rate of movement of organisms toward the X_{best} locations is moderated by chaotic random sequence to improve global search ability of the organisms. The SOS operators are continuously applied to the population of organisms which represents candidate solutions until the stopping criterion are reached.

Local Search Using Variable Neighborhood Search Variable neighborhood search (VNS) is a robust local search procedure that systematically uses the concept of neighborhood change for search space exploitation and avoidance of likely entrapment in local minima [28, 29]. VNS incrementally explores the distant neighborhoods of the present solution and move to the next neighborhood if there is an improvement in the current solution which makes VNS more robust than is in contrast to other local search approaches that traverse the trajectory of the solutions [42]. With this, the current solution is used to identify the potential neighborhood solutions. VNS has been successfully hybridized with metaheuristic algorithms in improving their search efficiency. Considering the fact that VNS has a strong

local search ability, a novel hybrid algorithm (SOS-VNS) is presented in order to further improve the local search ability of SOS algorithm. The proposed SOS-VNS algorithm utilizes the exploration ability of SOS and exploitation ability of VNS to improve the efficiency of cloud data centers. The framework of VNS algorithm is given as Algorithm 3.

Algorithm 3 *Steps of Variable Neighbourhood Search Algorithm [19]*

Input: Select the neighbourhood structure: $N_k; k = 1, 2, 3, ..., k_{max}$.
Output: Optimal schedule
1: Let the initial solution be s
2: Set $k = 1$
3: **while** $k \leq k_{max}$ **do**
4: Randomly select a point s' in the k^{th} neighbourhood of s
5: Apply some local search procedures on s' to obtain s''
6: **if then**s'' is better than current solution
7: $s = s''$
8: Continue search in the current neighbourhood structure
9: **else**
10: $k = k + 1$
11: **end if**
12: **end while**

The neighborhood structures must ensure that the imposed optimization constraints are not violated. This study employs four different kinds of neighborhood structures.

Neighborhood Structure N_1: The neighborhood structure N_1 intends to change the server that a VM belongs to, seeking to improve resource utilization, thereby improving energy efficiency. Suppose that s is a candidate solution and i is the index of the randomly selected VM. Let $s[i] = h_i$ be the server that host VM i. The value of $s[i] = h_i$ is changed as follows: First, a check is done if VM with index i can be moved to a new server without violating the imposed constraints, in this case, the processing, memory, and storage constraints. The choice of the server to which VM i will be moved to is determined as follows: for each VM_j $(i \neq j)$, if $s[i] \neq s[j]$ then $[j]$ is a candidate server to accept the VM_i. If the candidate servers are more than one, then VM i is moved to the server with lesser utilization. As a result, a solution s^0 is better than a solution x if and only if it has a better resource utilization with respect to s.

Neighborhood Structure N_2: While N_1 changes the server that a VM belongs to, seeking to improve resource utilization, neighborhood structure N_2 migrates a VM i to another server to improve the degree of load balance. For a candidate solution s, the value of $s[i] = c_i$ is changed by the neighborhood as follows: the first step is to ensure that the server constraints (CPU, memory, and storage capacity) are not violated which is similarly done like for the N_1 neighborhood. If the constraints are not violated, then N_2 is applied. The server to which VM i will be migrated to is determined as follows: For each VM VM_j $(i \neq j)$, if $s[i] \neq s[j]$, then $[j]$ is a candidate server to accept the VM i. If the candidate servers are more than one, then

VM i is moved to the server with lesser load. As a result, a solution s^0 is better than a solution x if and only if its degree of load balance is lower than that of s.

Neighborhood Structure N_3: Like N_1 neighborhood, neighborhood structure N_3 is to achieve a better resource utilization. This is achieved by exchanging the VM i and VM j belonging to different servers h_i and h_j. In particular, to exchange VMs between servers h_i and h_j, the following conditions must be satisfied.

- The removal of VM i from server h_i does not violate the imposed constraints.
- The removal of VM j from server h_j does violate the imposed constraints.
- If VM i shares the same server with at least one VM k ($k \neq j$), in server h_j
- If VM j shares the same server with at least one VM k ($k \neq i$) in server h_i.

If the above conditions are satisfied, then VM i is migrated to server h_j and VM j is migrated to server h_i. In this case, a solution s^0 is better that s if and only if it has a better resource utilization with regard to s.

Neighborhood Structure N_4: Like N_3 neighborhood, neighborhood structure N_4 exchanges VM i and VM j belonging to different servers h_i and h_j. Different from N_3. In contrast to N_3, the essence of N_4 is to achieve more compact servers. As for N_3, to exchange VMs between servers h_i and h_j, the following conditions must be satisfied.

- The removal of VM i from server h_i does not violate the imposed constraints.
- The removal of VM j from server h_j does violate the imposed constraints.
- If VM i shares the same server with at least one VM k ($k \neq j$) in server $h_{j.}$
- If VM j shares the same server with at least one VM k ($k \neq i$) in server h_i.

If the above conditions are satisfied, then VM i is migrated to server h_j and VM j is migrated to server h_i. In this case, a solution s^0 is better that s if and only if it has a better degree of load balance with regard to s.

The Proposed VM Scheduling Algorithm In this section, the VM scheduling algorithm is presented. The various components of the proposed algorithms have been given in the previous sections. Hence, the problem initialization, application of SOS algorithm to optimize the objective function, and the use of variable neighbourhood search. The different constituents of the technique address different problems as they affect existing VM scheduling techniques. More specifically:

- The initialization method produces initial solutions to speed up the convergence rate of the SOS algorithm.
- The SOS algorithm optimizes the objective function in a more efficient manner.
- The use of VNS further improves the quality of the obtained solution, hybridizing the ability of SOS in efficiently exploring the solution search space with the capability of VNS (local search) in exploiting a specific region of the solution search space.

The resulting VM scheduling technique is given as Algorithm 4, while the VNS algorithm is given as Algorithm 5.

Algorithm 4 *The SOS based VM Scheduling technique*

Input: Eco_size = number of severs (hosts)
$E = \{E_i | i = 1, 2, 3, ..., Eco\_size\}$: ecosystem
h = the number of hosts
v = the number of VMs
G_{max} = maximum number of generations
Output: Optimal VM schedule - the best organism in E at the end of the evolution.

Initialization
Create an empty ecosystem E
Initialize E using uniformly generated random numbers; check and correct the constraint violation
Evaluate the fitness of the organisms
Identify the best organism E_{best}
$g \leftarrow 1$
Main loop SOS
while $g \leq G_{max}$ **do**
  for $i \leftarrow 1$ to Eco_size **do**
    Mutualism Phase
    Randomly select organism $E_j (i \neq j)$
    Update E_i and E_j according to Equations 12 and 13 respectively.
    Commensalism Phase
    Randomly select organism $E_j (i \neq j)$
    Update E_i according to Equation 15.
    Parasitism Phase
    Create parasite vector
    Update E_i according to Equation 16.
    Identify the best organism X_{best}
  end for
  Apply variable neighbourhood search
  VNS(E)
  $g \leftarrow g + 1$
end while
return E_{best}

Algorithm 5 *Local Search with Variable Neighbourhood Search Algorithm*

Input: E = the ecosystem of organisms (candidate solutions)
Set the neighbourhood structure: $N_k; k = 1, 2, 3, 4$ as defined in Section 3.3
Let ρ be the probability of applying VNS to an organism.
Let E_i be the ith organism in E
Let T_{max} be the maximum number of iterations
for $i \leftarrow 1$ to Eco_size **do**
  $r \leftarrow rand[0, 1]$ ▷ $rand[0, 1]$ is a random number between 0 and 1
  if then $r \leq \rho$
    Set $k = 1$
    while $k \leq 4$ **do**
      for $j \leftarrow 1$ to T_{max} **do**
        Generate a solution E_i' from the kth neighbourhood of $E_i; E_i' \in N_k(E_i)$
        Apply Simulated Annealing using E_i' as the initial solution to obtain E_i''
        if then E_i'' is better than E_i
          $E_i \leftarrow E_i''$
          Continue search in N_k
        else
          $k = k + 1$
        end if
      end for
    end while
  end if
  Apply variable neighbourhood search
  VNS(E)
  $g \leftarrow g + 1$
end for
return E_{best}

4 Performance Evaluation

This section presents the performance evaluation of the proposed VM scheduling algorithm.

4.1 Experimental Setup

This study used CloudSim simulator toolkit [43] for performance evaluation of the proposed technique. The simulator toolkit is the most popular simulator used by cloud computing researchers because of its support modeling and simulation of cloud computing infrastructures [44]. It provides support for simulating resource management and scheduling algorithms through its cloud information service and DataCenterBroker components for the realization of resource discovery and information exchange.

Table 1 Workload traces characteristics in CPU utilization

Instance name	Date	Number of VMs	Mean (%)	STD (%)
t_1	03/03/2011	1052	12.31	17.09
t_2	06/03/2011	898	11.44	16.83
t_3	09/03/2011	1061	10.70	15.57
t_4	22/03/2011	1516	9.26	12.78
t_5	25/03/2011	1078	10.56	14.14
t_6	03/04/2011	1463	12.39	16.55
t_7	09/04/2011	1358	11.12	15.09
t_8	11/04/2011	1233	11.56	15.07
t_9	12/04/2011	1054	11.54	15.15
t_{10}	20/04/2011	1033	10.43	15.21

The development of new algorithms is carried in the DataCenterBroker which equally provides support for the development of energy-aware algorithms (Table 1).

The simulated IaaS cloud platform is configured with two data centers, 200 hosts of two types, each type consisting of 100 hosts. The configuration settings of the hosts are given as Table 2. Equation 8 is used to estimate the energy consumption of each host with $E_{\max}$ value of 259 W. The popular workload from monitoring facilities of PlanetLab was used to generate tasks for VMs which is available in CloudSim simulator. The workload primarily contains CPU utilization of VMs across 500 data centers for about 5 min. The workload traces were collected for 10 days, and the characteristics of the traces for each day are shown in Table 1. All the workload traces were used for the evaluation of the proposed algorithm. The study adopted a task scheduling algorithm of CloudSim for scheduling of tasks on VMs.

The performance of SOS and SOS-VNS algorithms was carried using similar conditions. Each algorithm is run independently for 30 times using each workload. For each run, 100 organisms (individuals) constitute the ecosystem (population) which evolves for 100 generations as the stopping criteria. The values of the benefit factors β_1 and β_2 are, respectively, determined stochastically as either 1 or 2. The probability of applying VNS to an organism is 0.2 and $T_{\max}$ of Algorithm 5. A comparison was conducted in terms of SLAV and energy consumption in the data center to evaluate the performance of the algorithm. Resource utilization and energy consumption are used as the performance metrics for evaluating the proposed technique.

4.2 Results and Discussion

This section reports the experimental results obtained from the simulation of the proposed technique. Energy consumption and resource utilization are used as performance metrics to evaluate the efficiency of the proposed SOS-VNS algorithm.

Table 2 Host configuration settings

Host type	Parameter	Value
ProLiant ML110 G3	Number	2
	Processor	3000 MHz * dual-core
	RAM	6 GB
	Storage	8 GB
	Bandwidth	1 GB
	Operating system	Linux
	VMM	Xen
ProLiant DL360 G4p	Number	2
	Processor	3400 MHz * dual-core
	RAM	8 GB
	Storage	6 GB
	Bandwidth	1 GB
	Operating system	Linux
	VMM	Xen

Figures 1, 2, and 3, respectively, show statistics (best, average, and standard deviation) of the energy consumption of the hosts for different workload instances (t_1,t_2,t_3,t_4,t_5,t_6,t_7,t_8,t_9,t_{10}). The SOS algorithm allocates VMs using the standard SOS concepts, while SOS-VNS algorithm dynamically allocates VMs to achieve lower energy consumption while improving the utilization of compute resources. Each workload instance was executed 30 times, and the best, average, and standard deviation of energy consumption of the hosts are recorded. As it can be observed from Figs. 1, 2, and 3, the proposed technique produces the server schedules with lower energy consumption. According to the results of the experiments, the SOS and SOS-VNS can the performance of the system in terms of energy consumption and resource utilization. This is due to the strong global search ability of SOS algorithm and robust local search ability of VNS algorithm which equips the SOS-VNS with both exploration and exploitation capability, thereby leading to the enhanced performance. Clearly, the proposed SOS-VNS algorithm produces lower energy consumption as compared to SOS and LR-MMT algorithm [44].

The average CPU, memory, and storage utilization for different number of VMs on workload instance t_4 obtained by SOS-VNS, SOS, and LR-MMt algorithms are shown in Figs. 4, 5, and 6, respectively. The SOS-VNS algorithm outperformed SOS and LR-MMT algorithm in terms CPU, memory, and storage utilization, the SOS-VNS algorithm obtained the highest CPU, memory, and storage utilization averaged 90.52, 86.71, and 86.33%, respectively. The average CPU, memory, and storage utilization obtained by SOS algorithm are 82.89, 81.85, and 80.41%, respectively, while those of LR-MMT algorithm are 73.83, 71.66, and 74.36%, respectively. The proposed algorithm obtained better resource (CPU, memory, and storage) utilization

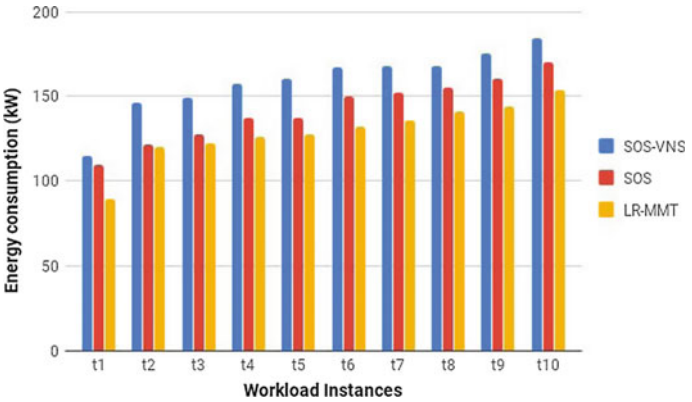

Fig. 1 Best energy consumption

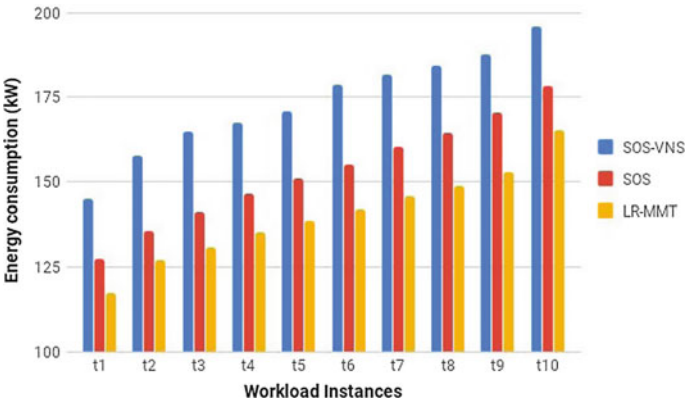

Fig. 2 Average energy consumption

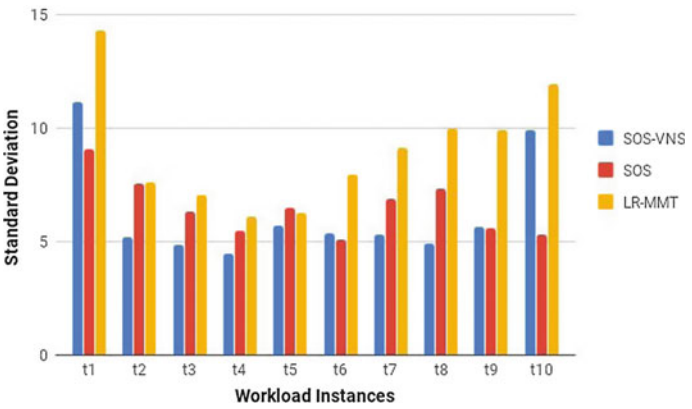

Fig. 3 Standard deviation of energy consumption

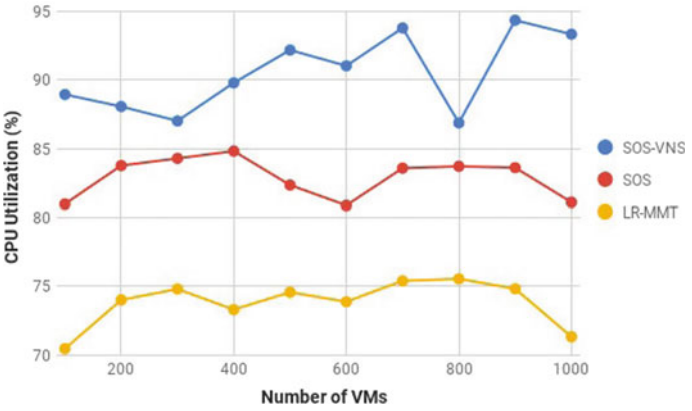

Fig. 4 Average CPU utilization

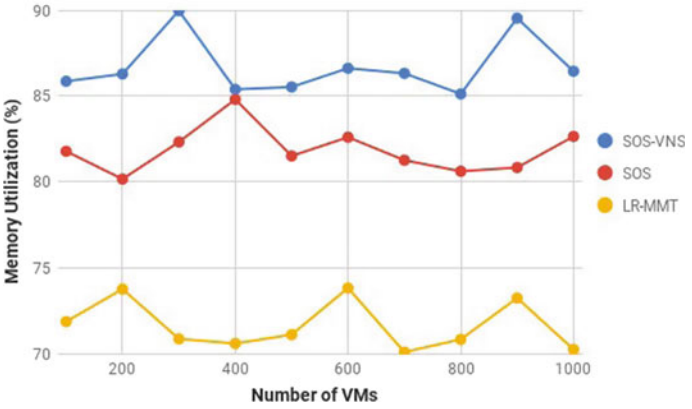

Fig. 5 Average memory utilization

compared to SOS and LR-MMT algorithm. This higher performance is likely due to the efficient local search capability of VNS algorithm integrated into SOS algorithm. Utilizing the SOS-VNS algorithm to optimize energy consumption parameters of the cloud model can improve the energy consumption optimization for cloud computing data centers.

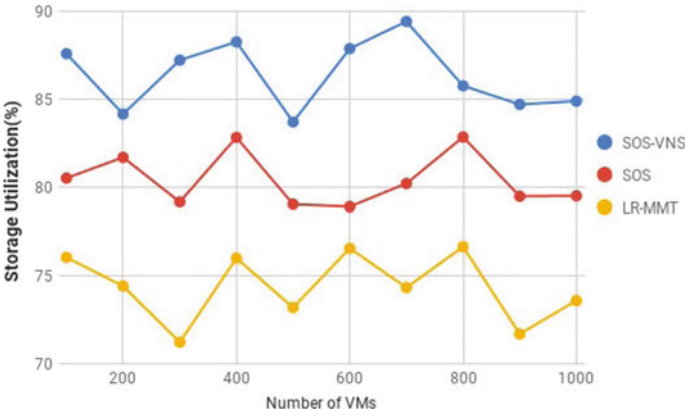

Fig. 6 Average storage utilization

5 Conclusion and Future Work

High energy consumption of data centers has been a major concern for cloud for researchers and cloud stakeholder, the cost of settling energy bills constitutes a significant cost of running cloud systems. This concern motivated the development of an efficient SOS-VNS algorithm for energy-efficient management in cloud computing. The optimal VM allocation has been achieved with minimal number of active hosts and turning off the idle servers for energy saving. VM scheduling problem is a NP-Complete problem, and a SOS-VNS algorithm has been proposed to solve this problem. The schedule of VMs is constructed using symbiotic organism search algorithm which is characterized by strong global convergence to optimal solution. The strong global convergence of SOS makes it suitable for optimizing large-scale problems. This is a great advantage over heuristic algorithms which fail to find optimal solutions for relative large-scale cloud computing scheduling problems. The SOS-VNS algorithm is applied IaaS cloud system of different sizes and characteristics. The results of the experiment prove that SOS-VNS algorithm is able to minimize number of active servers, resource balancing, improve resource utilization, and reduction in consumption. The enhanced performance is enabled by robust local search ability of variable neighborhood search. Conclusively, the SOS-VNS is an efficient approach to VM scheduling problem. As part of future work, rigorous experimentation of the proposed technique can be considered. Also, design and development of efficient VM allocation algorithms for large-scale heterogeneous and distributed data centers will be our focus of future.

References

1. Chandrashekar DP (2015) Robust and fault-tolerant scheduling for scientific workflows in cloud computing environments. Ph.D. thesis
2. Poola D, Ramamohanarao K, Buyya R (2014) Fault-tolerant workflow scheduling using spot instances on clouds. Proc Comput Sci 29:523–533
3. Vouk MA (2008) Cloud computing–issues, research and implementations. J Comput Inf Technol 16(4):235–246
4. Caron E, Desprez F, Loureiro D, Muresan A (2009) Cloud computing resource management through a grid middleware: a case study with diet and eucalyptus. In: IEEE international conference on cloud computing, 2009. CLOUD'09, IEEE, pp 151–154
5. Lei H, Wang R, Zhang T, Liu Y, Zha Y (2016) A multi-objective coevolutionary algorithm for energy-efficient scheduling on a green data center. Comput Oper Res 75:103–117
6. Duan H, Chen C, Min G, Wu Y (2017) Energy-aware scheduling of virtual machines in heterogeneous cloud computing systems. Future Gener Comput Syst 74:142–150
7. Kolodziej J, Khan SU, Xhafa F (2011) Genetic algorithms for energy-aware scheduling in computational grids. In: 2011 international conference on P2P, parallel, grid, cloud and internet computing, IEEE, pp 17–24
8. Achary R, Vityanathan V, Raj P, Nagarajan S (2015) Dynamic job scheduling using ant colony optimization for mobile cloud computing. In: Intelligent distributed computing, Springer, pp 71–82
9. Abdullahi M, Ngadi MA et al (2016) Symbiotic organism search optimization based task scheduling in cloud computing environment. Future Gener Comput Syst 56:640–650
10. Zhong SB, He ZS (2010) The scheduling algorithm of grid task based on pso and cloud model. Key Eng Mater 439:1487–1492
11. Geng J, Huang ML, Li MW, Hong WC (2015) Hybridization of seasonal chaotic cloud simulated annealing algorithm in a svr-based load forecasting model. Neurocomputing 151:1362–1373
12. Ibrahim H, Aburukba RO, El-Fakih K (2018) An integer linear programming model and adaptive genetic algorithm approach to minimize energy consumption of cloud computing data centers. Comput Electr Eng 67:551–565
13. Liu XF, Zhan ZH, Deng JD, Li Y, Gu T, Zhang J (2016) An energy efficient ant colony system for virtual machine placement in cloud computing. IEEE Trans Evolut Comput
14. Sharma N, Guddeti RM (2016) Multi-objective energy efficient virtual machines allocation at the cloud data center. IEEE Trans Serv Comput
15. Cheng MY, Prayogo D (2014) Symbiotic organisms search: a new metaheuristic optimization algorithm. Comput Struct 139:98–112
16. Dosoglu MK, Guvenc U, Duman S, Sonmez Y, Kahraman HT (2016) Symbiotic organisms search optimization algorithm for economic/emission dispatch problem in power systems. Neural Comput Appl pp 1–17
17. Secui DC (2016) A modified symbiotic organisms search algorithm for large scale economic dispatch problem with valve-point effects. Energy 113:366–384
18. Duman S (2016) Symbiotic organisms search algorithm for optimal power flow problem based on valve-point effect and prohibited zones. Neural Comput Appl pp 1–15
19. Zamani MKM, Musirin I, Suliman SI (2017) Symbiotic organisms search technique for svc installation in voltage control. Indones J Electr Eng Comput Sci 6(2):318–329
20. Tran DH, Cheng MY, Prayogo D (2016) A novel multiple objective symbiotic organisms search (mosos) for time-cost-labor utilization tradeoff problem. Knowl-Based Syst 94:132–145
21. Abdullahi M, Ngadi MA (2016) Hybrid symbiotic organisms search optimization algorithm for scheduling of tasks on cloud computing environment. PLoS ONE 11(6):e0158229
22. Abdullahi M, Ngadi MA, Dishing SI (2017) Chaotic symbiotic organisms search for task scheduling optimization on cloud computing environment. In: 6th ICT international student project conference (ICT-ISPC), IEEE, 2017, pp 1–4

23. Panda A, Pani S (2016) A symbiotic organisms search algorithm with adaptive penalty function to solve multi-objective constrained optimization problems. Appl Soft Comput 46:344–360
24. Prayogo D, Cheng MY, Prayogo H (2017) A novel implementation of nature-inspired optimization for civil engineering: a comparative study of symbiotic organisms search. Civ Eng Dimens 19(1):36–43
25. Dib NI (2016) Design of linear antenna arrays with low side lobes level using symbiotic organisms search. Prog Electromagn Res B 68:55–71
26. Nanda SJ, Jonwal N (2017): Robust nonlinear channel equalization using wnn trained by symbiotic organism search algorithm. Appl Soft Comput
27. Wu H, Zhou Y, Luo Q, Basset MA (2016) Training feedforward neural networks using symbiotic organisms search algorithm. Comput Intell Neurosci 2016
28. Hansen P, Mladenović N (2001) Variable neighborhood search: principles and applications. Eur J Oper Res **130**(3):449–467
29. Hansen P, Mladenović N, Urošević D (2006) Variable neighborhood search and local branching. Comput Oper Res **33**(10):3034–3045
30. Gasior J, Seredyński F (2013) Multi-objective parallel machines scheduling for fault-tolerant cloud systems. In: International conference on algorithms and architectures for parallel processing, Springer, pp 247–256
31. Jung D, Suh T, Yu H, Gil J (2014) A workflow scheduling technique using genetic algorithm in spot instance-based cloud. KSII Trans Internet Inf Syst 8(9)
32. Tawfeek MA, El-Sisi A, Keshk AE, Torkey FA (2013) Cloud task scheduling based on ant colony optimization. In: 2013 8th international conference on computer engineering & systems (ICCES), IEEE, pp 64–69
33. Ramezani F, Lu J, Hussain FK (2014) Task-based system load balancing in cloud computing using particle swarm optimization. Int J Parallel Prog 42(5):739–754
34. Raghavan S, Sarwesh P, Marimuthu C, Chandrasekaran K (2015) Bat algorithm for scheduling workflow applications in cloud. In: 2015 international conference on electronic design, computer networks & automated verification (EDCAV), IEEE, pp 139–144
35. Madni SHH, Latiff MSA, Abdullahi M, Usman MJ et al (2017) Performance comparison of heuristic algorithms for task scheduling in Iaas cloud computing environment. PLoS ONE 12(5):e0176321
36. Madni SHH, Latiff MSA, Coulibaly Y et al (2016) Resource scheduling for infrastructure as a service (iaas) in cloud computing: challenges and opportunities. J Netw Comput Appl 68:173–200
37. Vasudevan M, Tian YC, Tang M, Kozan E, Zhang X (2018) Energy-efficient application assignment in profile-based data center management through a repairing genetic algorithm. Appl Soft Comput 67:399–408
38. Fernandez-Caro D, Fernández-Montes A, Jakóbik A, Kołodziej J, Toro M (2018) Score: simulator for cloud optimization of resources and energy consumption. Simul Model Pract Theory 82:160–173
39. Luo J, Li X, Chen M (2014) Hybrid shuffled frog leaping algorithm for energy efficient dynamic consolidation of virtual machines in cloud data centers. Expert Syst Appl 41(13):5804–5816
40. Greenberg A, Hamilton J, Maltz DA, Patel P (2009) The cost of a cloud: research problems in data center networks. ACM SIGCOMM Comput Commun Rev
41. Liu X-F, Zhan Z-H, Deng JD, Li Y, Gu T Zhang, J (2018) An energy efficient ant colony system for virtual machine placement in cloud computing. IEEE Trans Evolut Comput
42. Vanneschi L, Henriques R, Castelli M (2017) Multi-objective genetic algorithm with variable neighbourhood search for the electoral redistricting problem. Swarm Evolut Comput
43. Calheiros RN, Ranjan R, Beloglazov A, De Rose CA, Buyya R (2011) Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw: Pract Exp 41(1):23–50
44. Beloglazov A, Abawajyb J, Buyya R (2012) Energy-aware resource allocation heuristics for efficient management of datacenters for cloud computing. Future Gener Comput Syst