Neural Computing and Applications
https://doi.org/10.1007/s00521-020-05559-2

ORIGINAL ARTICLE

A multi-objective Monarch Butterfly Algorithm for virtual machine placement in cloud computing

Mohamed Ghetas[1]

Received: 10 April 2020 / Accepted: 23 November 2020

Abstract
The growing demand for cloud computing adoption presents more challenges for researchers to make cloud computing more efficient and affordable for infrastructure providers and end users. The management of cloud computing involves investing in IT infrastructure in the first phase and investing in energy, maintenance and space costs in the second phase. However, energy costs account for a large portion of cloud management costs, and saving energy consumption can significantly reduce overall cloud management costs. Server consolidation is a strategy to improve data center energy efficiency and resource utilization. Virtual machine (VM) placement is considered one of the main problems with VM consolidation. The VM placement problem aims to reduce the number of active physical machines in data center to reduce data center power consumption and maintenance costs. However, the waste of data center resources has a significant impact on the energy efficiency of the data center, so it should be considered in the VM placement strategy. This paper proposes a new method based on the Monarch Butterfly Optimization algorithm (MBO) called MBO-VM for new virtual machine placement, designed to maximize packaging efficiency and reduce the number of active physical servers. CloudSim toolkit is used to test the efficiency of the proposed MBO-VM approach under real cloud workloads as well as synthetic workloads. Simulation results show that MBO-VM produces significantly better results compared with known VM placement techniques. The proposed MBO-VM can reduce the number of active servers more effectively and maximize the packaging efficiency.

Keywords Cloud computing · VM placement · Server consolidation · Optimization

1 Introduction

Cloud computing technology has been rapidly adopted by many companies. This adoption of cloud computing has some benefits, such as reliability, robustness and quality of service. Virtualization technology makes the potential of cloud computing possible through three different types of services on the Internet: The infrastructure is delivered as a service, such as the Amazon Elastic Compute cloud, the runtime environment, such as Google App Engine, is called platform as a service, and finally called Software as a service, such as SalesForce.com [1, 2].

Virtualization divides the hardware resources of one or more computers into a multiple execution environments called a virtual machine (VM). These virtual machines allow multiple applications to run in a performance-isolated environment. Therefore, cloud computing providers provide unlimited virtual machines through virtualization, thereby ensuring the quality of service for end users. A physical machine (PM) hosts multiple virtual machines, and each VM requires a certain amount of resources, such as CPU, memory and storage specified by the end user [3, 4]. VM migration is a compelling feature in cloud computing that points to a dynamic response to VM requests to ensure the quality of service of a running application. Therefore, once the VM needs resources that cannot be satisfied by the hosted physical machine, the VM is migrated to another physical machine to provide resource requests for the VM. One of the important stages of VM migration is finding the appropriate PM to host VM [5].

✉ Mohamed Ghetas
Mohamed.rizk@thebes.edu.eg

[1] Computer Science, Integrated Thebes Institutes, Maadi, Cairo, Egypt

Published online: 03 January 2021

Virtual machine placement is the process of mapping virtual machines to physical machines. In terms of energy efficiency, keeping machines idle is the main reason for high energy consumption. In fact, idle machines consume about 50–70% of the energy of active machines. As a result, VM placement has received a lot of interest because it allows cloud providers to control the use of resources as they can reduce the number of active machines and increase the energy efficiency of their infrastructure [7].

As a result, some research has focused on finding methods and technologies to reduce resource waste and improve data center energy efficiency. Because the request for deploying VM is unpredictable so that the virtual machine placement is considered an extremely complex allocation problem. Moreover, cloud providers often have large data centers and aim to serve workload fluctuation, it turns out that finding the best virtual machine placement strategy is an NP-hard problem. For example, if n is the total number of servers running and m is the number of virtual machines that must be deployed on the physical server, the possible number of virtual machines allocated to the available servers is n^m. Therefore, the solution space is too large, making virtual machine placement difficult to manage and requiring automation [8].

This research addresses virtual machine placement issues that consider multiple objectives. Consider reducing data center resource waste and power consumption by reducing the number of active physical machines. Many researchers have proposed many constructive and iterative strategies, some of them have proposed meta-heuristic optimization algorithms. However, monarch butterfly (MBO) optimization seems to outperform other optimization methods [9, 10]. Therefore, this study proposes a meta-heuristic method based on MBO and considers VM placement problem as a multi-objectives two-dimensional packaging optimization problem. Therefore, the proposed algorithm takes into account the multi-objective version of the VM placement problem and proposes a new VM placement method that takes into account both the energy efficiency of the data center and the reduction of waste of resources. Finally, the performance of the proposed algorithm is compared with different existing methods.

The remainder of the paper is organized as follows: Sect. 2 reviews and discusses the related work. Section 3 presents VM placement problem mathematical model where Sect. 3.1 introduces MBO algorithm details. Section 3.2 presents the proposed VM placement approach. Section 3.3 introduces the performance evaluation and the simulation results. Finally, the researcher concludes the paper with a summary and future work.

2 Related work

VM placement has an implication on application performance as well as cloud provider revenue. This leads to the problem of VM placement which aims to get optimal VM to PM allocation [11, 12]. Numerous techniques are proposed in the literature for VM placement based on different objectives, such as improving the energy efficiency of the physical server hosting virtual machines by reducing the waste of infrastructure resources [13], increasing the use of resources through consolidation [14] and distribution of workload virtual machines among physical machines to improve overall system performance [15, 16]. The VM placement techniques in the literature are classified into exact algorithms such as linear programming problem (LP) [17–21], dynamic programming [22], backward, branching and linking and meta-heuristics [6].

This research focuses on the meta-heuristic approach to address the problem of VM placement in cloud computing. The meta-heuristic strategy provides a high-level intelligent framework for solving VM placement in cloud computing. For example, ant colony optimization (ACO) is widely proposed to optimize VM allocation. In [23], they used ACO to find the best placement of the workload to improve the energy efficiency of the data center.

In addition, ACO has been applied to balance the use of computing resources [24]. In [25], the authors proposed that ACO adjusts the number of active servers based on the current workload. However, this technology consolidates virtual machines into a single resource and has high computational costs. In [6], the author introduced the ACO algorithm to find a set of non-primary solutions to improve the energy efficiency of the infrastructure and improve resource utilization. [26] Consider multiple resource constraints of physical machines and proposed ACO to improve network performance and increase network scalability.

The work in [27] solved the VM placement problem as a multi-objective optimization problem and applied ACO with positive feedback mechanism to update the pheromone for improving the convergence speed. In addition to using ACO, many researchers have also proposed particle swarm optimization (PSO) to solve the VM placement problem. For example, in [28], PSO is used to migrate VMs from overloaded hosts to underloaded hosts and to reduce power consumption. The author in [29] introduced PSO with a two-dimensional particle coding scheme to model VM placement problems to reduce energy consumption.

In addition, the author in [30] uses PSO to enhance VM allocation by reducing resource waste. The author in [31] proposed a hybrid genetic algorithm and PSO method to improve VM allocation, thereby reducing energy consumption and meeting service level agreements (SLA). The work in [32] introduced discrete PSO-based energy-aware VM mapping to minimize power consumption by predicting power consumption before assigning VMs to physical servers. The author in [33] proposed a heuristic PSO that can optimize VM placement to reduce resource waste. The work in [34] addressed the VM placement problem as a multi-objective optimization problem and applies an improved genetic algorithm to find the best VM placement. The improved algorithm introduces local search to customize genetic operators and elite strategies to find the best VM mapping strategy.

In [35], the authors proposed an improved genetic algorithm NS-GGA to model VM placement problems to reduce the number of active PMs, minimize network traffic and balance the use of multi-dimensional resources. The author in [36] introduced VM placement and traffic communication algorithm (VPTC) to reduce network power consumption and avoid congestion. The VPTC algorithm considers these goals when deciding to allocate a virtual machine using the GA. GA can also be used to optimize neural networks to predict workload to reduce energy consumption in cloud computing.

The problem of VM consolidation has been extensively studied. The author in [37] proposed a discrete differential evolution algorithm to consolidate virtual machines into fewer physical machines by migrating virtual machines in real time. The algorithm aims to improve resource utilization and energy efficiency. One of the disadvantages of this algorithm is that it does not consider other optimization issues, such as minimizing VM migration and resource utilization during VM consolidation.

Many researchers have explored the placement of virtual machines with multiple objectives. For example, the authors in [38] proposed reinforcement learning to minimize energy consumption and waste of resources. In addition, the authors applied the Chebyshev scalar function to simplify the weight selection process. Compared with the existing methods, the algorithm shows good performance.

One of the most important performance indicators in cloud computing is network latency, which is affected by the shared bandwidth between different user applications. Therefore, the authors in [39] considered optimizing bandwidth usage in the objective function of the VM placement problem. They proposed a multi-objective optimization solution based on genetic algorithm to save energy and optimize the total data transmission on the shared channel in the data center.

Due to the natural dynamics of cloud applications and fluctuations in workload, the authors in [40] proposed a prediction model to predict future resource requirements. They introduced an anti-correlated placement algorithm (PACPA) to predict future CPU usage and bandwidth resource requirements for hosted applications. Experimental results show that PACPA can improve migration scheduling time and overall location performance.

The work in [41] introduces an improved bin packing heuristic algorithm to improve VM placement in the OpenStack Neat framework. In addition, the authors introduced a new bin packaging rule called medium tuning to meet SLA requirements and avoid unnecessary VM migrations. The algorithm increased the number of SLAs and migrations by 78% and 46%, respectively.

In [42], the authors introduced a new bandwidth allocation strategy in the VM placement problem. Moreover, they proposed an improved Levy flight based on a whale optimization algorithm to minimize the number of active physical machines and optimize bandwidth usage. The authors analyzed the results obtained by Friedman's test, which showed that the algorithm can reduce the number of active servers.

Compared with other optimization algorithms, the MBO algorithm has shown a good performance. Therefore, this research has developed an MBO-based method that can allocate virtual machines to a minimum number of physical servers to reduce energy consumption and minimize the waste of physical machine resources.

3 Problem statement and formulation

The cloud environment contains a large number of server nodes, and it is assumed that the data center infrastructure is fully virtualized and all applications are now running on virtual machines. Assigning a VM to a PM can be considered a multi-dimensional vector packing problem, and the size of the container is resource capacity of the physical machine. This work considers virtual machines and server nodes to be characteristics of two-dimensional resources (CPU and memory). This research does not consider disk storage because network attached storage (NAS) is assumed to be used as the primary storage in the cluster. If a server node hosts n virtual machines, the CPU utilization of that server is estimated to be the sum of the CPU utilization of the virtual machines. This concept applies to memory resources. For example, a server node hosts two virtual machines. The first VM requires 15%, 30% of the server's CPU and server's memory, and the second VM requires 20%, 40%. Therefore, the server utilization is estimated to be 35%, 70%. In other words, each server hosts many virtual machines (VMs) and provides appropriate resources (such as CPU, RAM) for each machine to be able to run all processes. Each VM requires a different amount of resources because it may serve different workloads. The total used resources of the server can be calculated as the total resources consumed by the hosted virtual machines, as shown in Eqs. 1 and 2, and given by [43]

$$U_j^{\text{cpu}} = \sum_{i=1}^{m} x_{ij} v_i^{\text{cpu}} \tag{1}$$

$$U_j^{\text{mem}} = \sum_{i=1}^{m} x_{ij} v_i^{\text{mem}} \tag{2}$$

where U_j^{cpu} and U_j^{mem} are, respectively, the CPU utilization and memory utilization of server j, m is the number of active servers. x_{ij} is a binary variable that indicates whether the $\mathtt{VM_i}$ is hosted on server j and v_i^{cpu}, v_i^{mem} represent the CPU and memory requirements of $\mathtt{VM_i}$.

The resource utilization of each server is limited by some thresholds to avoid performance degradation and VM migration. For example, the thresholds for CPU and memory utilization are usually set to 80%, 80%. Now let 20%, 30% be the CPU and memory requirements for the third VM. However, the server can meet the CPU requirements of the third VM, the server cannot host this VM due to memory shortage. Also, due to the natural dynamics of application workload, the resource each VM needs may fluctuate over time. Therefore, if the server cannot satisfy the dynamic resource requirements of the hosted VM, the VM migrates to another server to avoid violation of the SLA. The example above demonstrates how the VM placement strategy has a major influence on resource waste and VM migration. In this context, this work considers two objectives. The first one is to minimize the waste of resources, and the second is to reduce energy consumption under the SLA constraint.

3.1 Resource wastage model

The resources available on each server have a great impact on the number of active servers that can host new virtual machines. The good virtual machine placement solution aims to balance resource usage across all dimensions to host virtual machines as much as possible using a minimum number of active servers. In other words, the key behind VM placement is to make effective use of resources to reduce resource waste. This work adopts Eq. 3 from [43] to calculate the potential cost of wasting resources on each active server.

$$W_j = \frac{\left| r_j^{\text{cpu}} - r_j^{\text{mem}} \right| + \epsilon}{U_j^{\text{cpu}} - U_j^{\text{mem}}} \tag{3}$$

W_j represents the amount of wasted resources from server j, and r_j^{cpu} and r_j^{mem} represent the remaining percentage of CPU resources and memory from server j. ϵ is a small positive real number whose value is set to 0.0001.

3.2 Power consumption model

Recently, researchers have demonstrated a linear relationship between *CPU* utilization and energy consumption [44]. In addition, the energy consumption of idle server represents 50–70% of the total energy consumption of the active server. The idle servers must be turned off in order to reduce energy consumption in the data center. Therefore, the energy consumption of the server can be expressed in terms of the CPU utilization as shown in Eq. 4 and given by [6]

$$P_j^T = \left(P_j^{\text{busy}} - P_j^{\text{idle}} \right) \times U_j^{\text{cpu}} + P_j^{\text{idle}} \tag{4}$$

P_j^T is the total power consumption of server j, and P_j^{idle} and P_j^{busy} represent the average power consumption of the server when idle and fully used, respectively.

3.3 VM placement as multi-objective optimization problem

This section describes VM placement as a multi-objective combinatorial optimization problem that aims to optimize both energy consumption and the overall resource wastage. Multi-objective evolutionary optimization algorithms use population-based method to find a feasible solution. These algorithms depend on the dominant principles in the selection process. The definition of the dominance concept can be expressed as the following multi-objective minimization problem with m parameters and n objectives.

$$\text{Minimize}\, \mathbf{f}(\mathbf{x}) = [f_1(x_1, \ldots, x_m), \ldots, f_n(x_1, \ldots, x_m)] \tag{5}$$

$\mathbf{x} = (x_1, \ldots, x_m) \in X$ $\mathbf{f} = (f_1, \ldots, f_n) \in Y$ where $\mathbf{x}$ is the solution vector, X is the parameter space, $\mathbf{f}$ is the objective vector and Y is the objective space. To model the problem of virtual machine placement, it is assumed that the data center consists of m virtual machines that must be assigned to a set of PMs. Let $\mathtt{VMs} = \{v_1, v_2, \ldots, v_m\}$ be a set of virtual machines, and each v_i represents a two-dimensional resource vector $v_i = \{v_i^{cpu}, v_i^{mem}\}$. In addition, assume that $\mathtt{PMs} = \{Pm_1, Pm_2, \ldots, Pm_n\}$ to act as a set of servers available in the data center. In addition, the CPU and memory utilization thresholds of server j are $T_j^{\text{cpu}}, T_j^{\text{mem}}$, respectively. It is assumed that the resources required by each VM cannot exceed the resources that the hosting server can provide. Suppose y is a binary variable that indicates whether server j is in use. Therefore, the main objectives of this work (minimizing resource wastage as well as energy consumption reduction) as given by [6] are described in Eqs. 6 and 7

$$\text{Minimize} \sum_{j=1}^{n} P_j = \sum_{j=1}^{n} y_j \times \left(P_j^{\text{busy}} - P_j^{\text{idle}}\right) \times U_j^{\text{cpu}} + P_j^{\text{idle}} \tag{6}$$

$$\text{Minimize} \sum_{j=1}^{n} W_j = \sum_{j=1}^{n} y_j \times \frac{\left|\left(T_j^{\text{cpu}} - U_j^{\text{cpu}}\right) - \left(T_j^{\text{mem}} - U_j^{\text{mem}}\right)\right| + \epsilon}{U_j^{\text{cpu}} + U_j^{\text{mem}}}$$

subject to

$$\sum_{i=1}^{n} x_{ij} = 1 \forall j \in J \tag{7}$$

$$U_j^{\text{cpu}} = \sum_{i=1}^{m} x_{ij} v_{ij}^{cpu} \le T_j^{\text{cpu}} \forall j \in J$$

$$U_j^{\text{mem}} = \sum_{i=1}^{m} x_{ij} v_{ij}^{\text{mem}} \le T_j^{\text{mem}} \forall j \in J$$

$$y_{ij}, xij \in \{1, 0\} \forall i \in I, j \in J$$

The first constraint in Eq. 7 indicates that each VM i is assigned to a single server j. The second and third constraints model the resource requirements of the hosted virtual machine. Finally, the fourth constraint defines the domain of the arbitrary variables. The binary variable $x_{i,j}$ indicates if VM i is hosted on server j and the binary variable y_j indicates whether server j is in use or not.

In this research, the first objective is to minimize the power consumption of allocating VM_i to server j. This objective can be described as the partial contribution of VM_i to power consumption for server j as given by [6] and described in Eq. 8.

$$Obj_{i,j,1} = \frac{1}{\epsilon + \sum_{j=1}^{m} P_j(x_0)} \tag{8}$$

where $P_j(x_0)$ is the normalized power consumption of the feasible solution x_0 of server j and its value is calculated as follows

$$P_j(x_0) = \frac{P_j^T}{P_j^{\text{max}}} \tag{9}$$

where P_j^{max} is the peak power consumption of server j. Similarly to the first objective function, the overall resource wastage results from allocating VM_i to server j can be calculated as given by [6] and described in Eq. 10.

$$Obj_{i,j,2} = \frac{1}{\epsilon + \sum_{j=1}^{m} W_j(x_0)} \tag{10}$$

where $W_j(x_0)$ is the normalized resource wastage of server j for solution x_0. The final multi-objective function combines the first objective and the second objective for each VM assignment as follows

$$Obj_{i,j} = Obj_{i,j,1} + Obj_{i,j,2}. \tag{11}$$

4 Monarch butterfly optimization

MBO is a new population-based meta-heuristic algorithm, introduced by Wang et al. It ideals and simplifies the migratory behavior of monarch butterflies between two different places when the seasons change. MBO uses the migration operator and the adjusting operator to update the newly generated butterflies. Like other swarm-based algorithms, MBO applies the iterative approach to update and spawn new individuals. This approach can be demonstrated as follows

1. *Initialization* MBO randomly generates a predefined number of butterflies that make up the population. Each butterfly represents a possible solution to the problem.
2. *Fitness evaluations* The fitness of each butterfly is calculated and sorted according to the objective function.
3. *Division* The resulting population is divided into two subpopulations: Subpopulation 1 is called Land 1, while subpopulation 2 is called Land 2. The sizes of subpopulation 1 and subpopulation 2 are defined by the predefined ρ ratio, which is called the ratio of migration.
4. *Migration* This process produces the first part of the new population. Replace existing butterflies on Land 1 with randomly selected individuals from Land 1 and Land 2. The size of the newly generated part is equal to the size of existing butterflies on Land 1. Each newly generated butterfly x^{t+1} is generated as follows: Suppose $x_{i,k}^{t+1}$ is the value of element k of butterfly i in generation $t+1$, then $x_{i,k}^{t+1}$ can be calculated as shown in Eq. 12.

$$x_{i,k}^{t+1} = \begin{cases} x_{r1,k}^{t} & r \le \rho \\ x_{r2,k}^{t} & r > \rho \end{cases} \tag{12}$$

where ρ is the migration period and equals to 5/12. $x_{r1,k}^t, x_{r2,k}^t$ are position values k at iteration t of randomly selected butterflies from Land 1 and Land 2, respectively. In addition, r is randomly generated value calculated from Eq. 13.

$$r = rand \times peri \tag{13}$$

where *rand* is a random value generated from a uniform distribution, while *peri* is a predefined constant value and is called the migration period and is equal to 1.2

5. *Adjustment* This operator is used to generate the second part of the new population. The size of the newly generated population equals the number of butterflies on Land 2. Each new butterfly can be generated based on the best individual butterfly or a randomly selected butterfly on Land 2. Suppose $x^{t+1}_{j,k}$ is the position value k of the butterfly j from Land 2, then $x^{t+1}_{j,k}$ is generated as shown in Eq. 14.

$$x^{t+1}_{j,k} = \begin{cases} x^{t}_{best,k} & rand \le \rho \\ x^{t}_{r3,k} & rand > \rho \end{cases} \tag{14}$$

In the above equation, $x^{t}_{best,k}$ and $x^{t}_{r3,k}$ are the position value k of the best global individual in Land 1 and Land 2, and k position value of the randomly selected butterfly from Land 2, respectively. The newly generated butterfly can be further updated as shown in Eq. 15 if the value of the randomly generated number is greater than the adjustment rate *BAR*

$$x^{t+1}_{j,k} = x^{t+1}_{j,k} + \alpha(dx_k - 0.5) \tag{15}$$

where α is the weighting factor that controls the influence of the adjustment process as shown in Eq. 16, where $S_{\max}$ is the maximum walking step that the individual can move. Furthermore, dx is the local walk of butterfly j performed by Levy flight that can be calculated as shown in Eq. 17.

$$\alpha = S_{\max}/t^2 \tag{16}$$

$$dx = \text{lev̀y}(x^t_j) \tag{17}$$

The two subpopulations of the migration and adjustment operators are combined to form the new population. The same process continues dividing, updating and combining the population until a satisfactory solution is found or the algorithm reaches a predefined maximum number of iterations. The pseudocode of the migration operator and the adjusting operator is shown in Algorithm 1 and Algorithm 2, respectively.

Algorithm 1 Migration operator

```
Begin
for i = 1 to Size(Land1) do
    for k = 1 to Dim (all element in $i$th Butterfly) do
        $r_1$=rand * period
        if $r_1 \le \rho$ then
            $r_2$=round(size(Land 1) * rand +0.5)
            Land1$(i,k)$=Land1$(r_2,k)$
        else
            $r_2$=round(size(Land2) * rand +0.5)
            Land1$(i,k)$=Land2$(r_2,k)$
        end if
    end for $k$
end for $i$
end
```

Algorithm 2 Adjusting operator

```
Begin
for j = 1 to Size(Land2) do
    $\alpha = maxwalkstep/(GenNo)^2$
    StepSize = ceil(exprnd(2* Maxgen))
    $dx = LevyFlight(StepSize, Dim)$
    for k = 1 to Dim (all element in $i$th Butterfly) do
        $r_1$=rand * period
        if $r_1 \le \rho$ then
            Land2(j,k)=Best(k)
        else
            $r_3 = round(size(Land2) * rand + 0.5)$
            Land2(j,k)=Land2($r_3$,k)
            if rand > BAR then
                Land2(j.k)=Land2(j,k)+scale * ($dx(j)$+0.5)
            end if
        end if
    end for k
end for j
end
```

5 MBO for solving VM placement problem

In our previous work [46], resource optimization and provisioning framework (ROP) is proposed to satisfy QoS goals. The MBO-VM algorithm proposed in this research is used as global resource optimizer in ROP to optimize the VM placement process. Thus, the MBO-VM algorithm runs on the controller node to reduce the power consumption and the cost of cooling process.

The subsequent sections present the design of the MBO for solving the VM placement problem in cloud data center. Firstly, the researcher introduces encoding schema to adapt

MBO for solving VM placement problem. Secondly, an efficient repair operator known as greedy optimization algorithm (GOA) is employed. Thirdly, this work introduces, evaluates and tests three different types of distribution methods in MBO's subpopulations.

5.1 Encoding scheme

Meta-heuristics have proven to be effective and high-performance and can solve many problems in science and engineering. However, one of the biggest challenges in applying any meta-heuristic is finding the right candidate encoding scheme. According to the VM placement problem in Sect. 3, if there are n servers running and m VMs, the search space size is n^m.

In an MBO implementation, a potential solution is represented as a set of virtual machines that must be assigned to a server. Each butterfly in the population is represented by a matrix of size m, where m is the dimension of the problem to be solved. In addition, each element of the matrix refers to the position of the element k of the butterfly j, and its value is in the range of [1, n]. For example, Fig. 1 shows one of the candidate solutions for m virtual machines and $n = 3$ servers.

In this example, VM4, VM7, VM9 are assigned to a server with an ID equal to P1, whereas VM3, VM5 are assigned to server P2. Another example, if there are five servers and ten virtual machines, one of the possible solutions is [1 3 2 1 4 2 1 5 2 4].

At the beginning of each iteration, a new population will be randomly generated, where each butterfly is represented by an array $x_i = \{x_{i,1}, x_{i,2}, x_{i,3}, \ldots, x_{i,m}\}$ and represents one of the potential solutions. For example, Eq. 18 demonstrates the generation of new butterfly i, where the solution j is equal to $\{1, 2, 3, \ldots, m\}$, a generated random number $\phi \in [0, 1]$, and the upper bound (ub) and lower bound (lb) are equals n and 1, respectively. In addition, according to the mapping function in Eq. 19, each value of $x_{i,j}$ is rounded to a closed integer value.

$$x_{i,j} = lb + (ub - lb) \times \phi \tag{18}$$

$$g(x_{i,j}) == \begin{cases} round(x_j) & x_{i,j} \leq n \\ n & x_{i,j} > n \end{cases} \tag{19}$$

5.2 Repair and optimization operator

In meta-heuristic algorithms, most of the time, some newly generated individuals violate the constraints of the problem. In the context of VM placement, some solutions violate the capacity of the target server. To overcome this challenge, there are two approaches. The first solution is to use the penalty method in the objective function [45]. The subsequent solution is to use a greedy method for individual optimization [47]. However, the penalty methods are possible, but not for large-scale problems. Greedy algorithms can be used to improve the performance of meta-heuristic algorithms [48]. Therefore, this work applies greedy repair and optimization operators to repair all infeasible solutions.

Algorithm 3 Repair

```
1  Begin
2  Input: $PM = \{Pm_1, \ldots, Pm_n\}$, $VMs = \{v_1, \ldots, v_m\}$
3          Population, $T^{cpu} = \{T_1^{cpu}, \ldots, T_n^{cpu}\}$
4          $T^{ram} = \{T_1^{ram}, \ldots, T_n^{ram}\}$
5  for i = 1 to PopulationSize do
6     for k = 1 to Dim (all element in $j$th Butterfly) do
7        $Pm = PM[x(i,k)]$
8        $Pm^{cpu} = Pm^{cpu} + VM_{[k]}^{cpu}$
9        $Pm^{ram} = Pm^{ram} + VM_{[k]}^{ram}$
10       check if the new VM violates
11       the server capacity constraints
12       if $(Pm^{cpu} > T_{[x(i,k)]}^{cpu}) or$
13       $(Pm^{ram} > T_{[x(i,k)]}^{ram})$ then
14          release VM from overloaded server
15          $Pm^{cpu} = Pm^{cpu} - VM_{[k]}^{cpu}$
16          $Pm^{ram} = Pm^{ram} - VM_{[k]}^{ram}$
17          set the k-value (the released VM)
18          of the current butterfly to zero
19          $x(i,k) = 0$
20       end if
21    end for k
22 end for i
23 Output: Population, PM
24 end
```

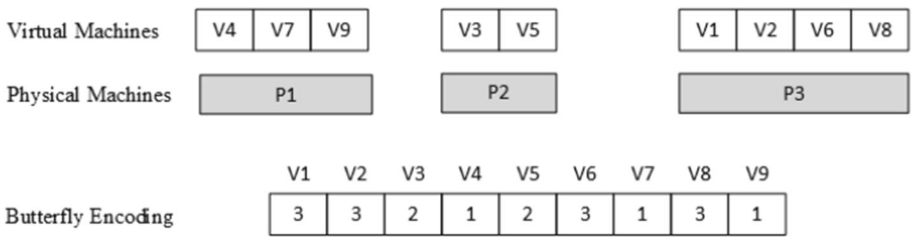

Fig. 1 An example of VM placement and its corresponding butterfly

Algorithm 4 Optimization Operator

```
1  Begin
2  Input: $PM = \{Pm_1, \ldots, Pm_n\}$
3  $VMs = \{v_1, \ldots, v_m\}$, Population
4  for i = 1 to PopulationSize do
5      for k = 1 to Dim do
6      check if current VM is not allocated to any PM
7          if $x(i,k) == 0$ then
8          $x(i,k) = Mapping(VM_k, PM, K)$
9          end if
10     end for k
11 end for i
12 Output: Population, PMPM
13 end
```

Algorithm 5 Mapping

```
1  Begin
2  Input: $PM = \{Pm_1, \ldots, Pm_n\}$, $VM_k$, K
3  IsMapped=false
4  for i = 1 to numberofPM do
3      $Pm = PM_{[i]}$
5      $Pm^{cpu} = Pm^{cpu} + VM^{cpu}_{[k]}$
6      $Pm^{ram} = Pm^{ram} + VM^{ram}_{[k]}$
7      check if this mapping does not
8      violate capacity constraints of PM
9      if $(Pm^{cpu} < T^{cpu}_{[i]})$&&
10     $(Pm^{ram} < T^{ram}_{[i]})$ then
11         IsMapped=true
12         break
13     else
14         release VM from overloaded server
15         $Pm^{cpu} = Pm^{cpu} - VM^{cpu}_{[k]}$
16         $Pm^{ram} = Pm^{ram} - VM^{ram}_{[k]}$
17     end if
18 end for i
19 if (!IsMapped) then
20     launch new Pm and add to PM list
21     allocated $VM_k$ to new PM
22 end if
23 Output: Pm
24 end
```

For abnormal individuals that violate the capacity limit mentioned in Eq. 7, the algorithm performs two operations: repair and optimization. On the one hand, Algorithm 3 shows the pseudocode of the repair operator. The algorithm estimates the final capacity of each server after mapping the virtual machines, as shown in Lines 6 to 9. Lines 12 to 18 detect overloaded servers, release some virtual machines from these servers and set the butterfly value k to zero, as shown in line 19. On the other hand, the optimization operator pseudocode is shown in Algorithm 4. The algorithm attempts to allocate the released virtual machine to a server with light load, as shown in Lines 4 to 11. However, if there are insufficient resources on the currently active servers to accommodate the released virtual machines, the algorithm starts a new server, as shown in Algorithm 5.

5.3 Main procedure of proposed MBO-VM algorithm

After the careful design of the encoding schema, and repair and optimization operator, it is time to present the proposed MBO-VM algorithm for VM placement. The pseudocode is described in Algorithm 6. The proposed MBO-VM algorithm first generates a random initial population of NP individuals and then divides the entire population into subpopulation 1 (NP1) and subpopulation 2 (NP2). The following evolutionary process repeats itself until a predefined certain criterion is satisfied. The migration operator uses the neighborhood mutation shown in Algorithm 1 to generate new monarch butterflies from subpopulation 1 and subpopulation 2. Then, the adjusting operator shown in Algorithm 2 uses the best butterfly in the two subpopulations and randomly selects individuals from the subpopulation 2 to generate a new subpopulation 2. In addition, the adjusting operator uses Levy flights to explore the search space more effectively and accelerates convergence. Then, the greedy algorithm shown in Algorithm 3 is applied to repair and optimize the infeasible solutions. Finally, the two subpopulations are regrouped to form the whole population. Figure 2 summarizes the main procedure of the proposed MBO-VM algorithm.

```
Algorithm 6 Proposed MBO-VM Placement Algorithm
    Begin
    Step 1: Initialization Set the generation counter $t$, adjusting operator rate $BAR$, migration period $Peri$,
        migration rate, max walk step $S_{max}$, migration period, and maximum generation $MaxGen$.
        Generate NP offsprings randomly $\{x_1, \ldots, x_{NP}\}$.
    Step 2: Fitness evaluation Calculate the fitness for each individual using Equation 11
    Step 3: While(stopping criterion)
        Divide the whole population into two subpopulations ($NP1$, and $NP2$)
        Find the global optimal individual $x_{best}$
        Generate new suppopulation1 using Algorithm 1.
        Generate new suppopulation2 using Algorithm 2.
        Repair and optimize the infeasible solutions using Algorithm 5.
        Update the best solutions $x_{best}$
        Increment generation counter
    Step 4: end while
    Step 5: Output the best results
```

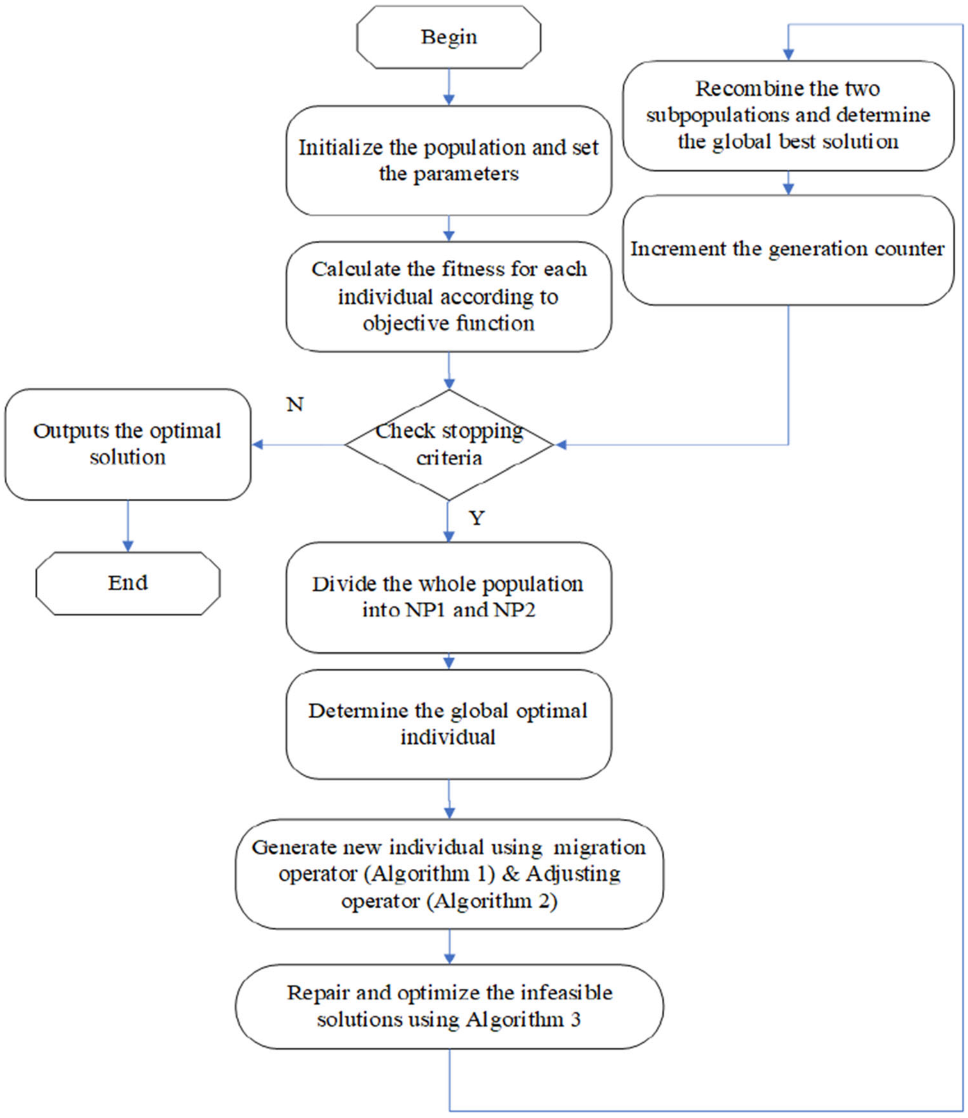
Begin
Initialize the population and set the parameters
Calculate the fitness for each individual according to objective function
Check stopping criteria
N
Y
Outputs the optimal solution
End
Divide the whole population into NP1 and NP2
Determine the global optimal individual
Generate new individual using migration operator (Algorithm 1) & Adjusting operator (Algorithm 2)
Repair and optimize the infeasible solutions using Algorithm 3
Recombine the two subpopulations and determine the global best solution
Increment the generation counter

Fig. 2 Flow chart of the MBO-VM algorithm

5.4 Complexity analysis of the proposed MBO-VM algorithm

This section estimates the time complexity of the proposed MBO-VM algorithm. Obviously, the time complexity is mainly determined by step 1–3. In step 1, the initialization of NP butterfly individual costs time $\mathcal{O}(NP \times n) = \mathcal{O}(n)$. In step 2, the time complexity for calculating the fitness of individuals using Quicksort algorithm costs $\mathcal{O}(nlogn)$. In step 3, the time complexity consists of computation times of migration operator, adjusting operator and greedy repair and optimization algorithms. The migration operator costs time $\mathcal{O}(NP1 \times n) = \mathcal{O}(n^2)$. The adjusting operator costs time $\mathcal{O}(NP2 \times n) = \mathcal{O}(n^2)$. Finally, the greedy repair and optimization algorithm costs $\mathcal{O}(n)$. Thus, the MBO-VM runs in time complexity $\mathcal{O}(n) + \mathcal{O}(nlogn) + \mathcal{O}(n^2) + \mathcal{O}(n^2) + \mathcal{O}(n) = \mathcal{O}(n^2)$.

5.5 MBO with different population methods

In the basic MBO, the population is divided into subpopulation 1 and subpopulation 2. Previous MBO studies have shown that individuals in subpopulation 1 have better fitness compared with individual in subpopulation 2 [49]. With VM placement problem, this work evaluated three different types of individual population strategies.

- *MBO Strategy 1* (*MBO-S1*) The subpopulation 1 and subpopulation 2 are randomly generated from the entire population. During the evaluation process at the end of each iteration, the newly generated subpopulation 1 and subpopulation 2 are not recombined. Therefore, each subgroup uses a different search operation to find potential solutions because there is little information exchange between subgroup 1 and subgroup 2. As a result, this strategy may perform worse and may not find a satisfactory solution.
- *MBO Strategy 2* (*MBO-S2*) During the initialization process, individuals with better fitness are assigned to subpopulation 1, and the remaining population belongs to subpopulation 2. Similar to MBO-S1, at the end of each iteration phase, the two subpopulations will not be recombined during the evaluation process.
- *MBO Strategy 3* (*MBO-S3*) This strategy is similar to the basic MBO, but the two subpopulations are recombined in a particular iteration and not every iteration. This method increases the performance of the evaluation process to a certain extent and therefore improves the ability to find the right possible solution.

6 Performance evaluation

The performance of our proposed MBO-VM algorithm to save energy and maximize resource usage has been compared with other VM placement algorithms in the literature.

6.1 Experimental setup

The CloudSim toolkit [50] was used to evaluate our proposed MBO-VM algorithm. CloudSim is the most widely used simulation tool among cloud researchers, and most cloud researchers use it extensively to develop VM placement algorithms. It provides a layered simulation framework that helps researchers to model, simulate, evaluate new applications, and cloud computing architectures. To evaluate the proposed MBO-VM algorithm, six different types of physical machine configurations were used to simulate the data center, as shown in Table 1. In addition, Table 2 shows all the different types of virtual machines used to evaluate the proposed VMP solution.

All experiments are performed using an HP laptop with Windows 10 operating system (with Core i5 and 8 GBs RAM), and three experiments are conducted to evaluate the effectiveness of the proposed MBO-VM solution in terms of energy saving and reducing resources wastage. The purpose of the first experiment is to verify the performance of MBO under different population strategies. The second experiment uses a synthetic workload that varied from light load to heavy load to evaluate the scalability of the proposed MBO-VM solution relative to other methods in the literature. The experiment consisted of 100–400 VMs and 100 PMs for two different types of servers.

The third experiment evaluates the performance of the proposed MBO-VM solution against real PlanetLab

Table 1 PM configurations in data center

Nos.	Machine model name	MIPS	Num cores	RAM (MB)	Type
1	HP Proliant ML 110 G4	1860	2	4096	Small
2	HP Proliant ML 110 G5	2660	2	4096	Small
3	HP Proliant ML 110 G3	3000	2	4096	Medium
4	IBM server x3250	3067	4	8192	Medium
5	IBM server x3550[Xeon-X5675]	3067	6	16384	Big
6	IBM server x3550 [Xeon-X5670]	2933	6	12288	Big

Table 2 VM configurations used in data center

Nos.	VM type	MIPS	Num cores	RAM (MB)	VM size (GBs)
1	Type 1 [Extra Big]	2500	1	1870	2.5
2	Type 2 [Big]	2000	1	1740	2.5
3	Type 3 [Small]	1000	1	1740	2.5
4	Type 4 [Extra Small]	500	1	613	2.5

Table 3 Energy consumption of different population strategies using PM Type 1

No. of VM	MBO-S1				MBO-S2				MBO-S3			
	Best	Worst	Mean	SD	Best	Worst	Mean	SD	Best	Worst	Mean	SD
100	23.44	24.93	23.63	0.80	22.31	24.23	23.95	0.70	20.15	20.85	20.52	0.18
200	45.21	46.86	46.32	0.90	44.76	45.23	45.10	0.76	43.22	43.83	43.50	0.12
250	57.34	58.63	58.42	0.96	56.46	57.37	56.76	0.82	54.78	55.12	54.84	0.14
300	70.21	73.47	72.48	1.65	68.72	70.54	69.21	1.21	64.72	64.98	64.84	0.13
400	92.34	95.32	94.23	1.77	91.86	93.98	92.56	1.43	89.55	90.98	89.65	0.08

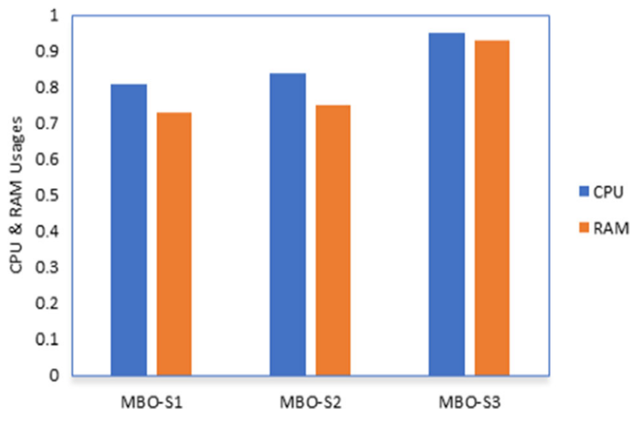

Fig. 3 The average CPU and RAM usage for different population strategies

workload traces [50]. This workload contains the CPU usage of 1033 virtual machines and is evaluated in a data center consisting of 400 PMs from six different types of servers.

6.2 Compare MBO performance with different population strategies in VM placement problem

This section uses the three different population strategies described above to evaluate the performance of the proposed MBO-VMP algorithm in terms of energy saving and resource waste reduction. Five different configurations are used to compare the performance of the proposed three population strategies under different load conditions (light to heavy load). The experiment uses two types of physical machines to simulate a 100 PM data center: HP ProLiant ML110 G4 and IBM x3250 server, with workloads ranging from 100 VM to 400 VMs.

Table 3 shows the performance comparison of MBO-S1, MBO-S2 and MBO-S3 in four different configurations. The table shows the best solution, worst solution and mean solution for the three different strategies along with standard deviation (SD). The best values obtained for each strategy are shown in bold. It is clear from Table 3 that MBO-S3 shows superior performance compared with the other two strategies. For example, the mean, best and worst values of all configurations obtained through MBO-3 are better than the values obtained through MBO-S1 and MBO-S2. In addition, the SD value of MBO-3 is much smaller than the SD values of the other two strategies. Generally, MBO-S3 is better than MBO-S1 and MBO-S2 in minimizing power consumption and has stable performance under different load conditions.

Figure 3 shows the average results of CPU and RAM usage when using MBO-S1, MBO-S2 and MBO-S3. Obviously, MBO-3 achieved the highest CPU and RAM usage, close to 100%. It is clear that, MBO-S3 balances CPU and RAM usage compared with MBO-S1 and MBO-S2. Therefore, MBO-S3 effectively finds a solution space and can obtain the best solution. This solution can minimize the waste of resources and balance the use of resources in different dimensions.

Experimental results show that the average performance of MBO-S3 is better than MBO-S1 and MBO-S2. Therefore, this research uses MBO-S3 (MBO-VMP) to solve the problem of virtual machine placement in cloud computing.

6.3 MBO-VM performance using synthetic workloads and variable virtual machine numbers

The purpose of this experiment is to compare the proposed MBO-VM with other literature methods such as ACO, PSO

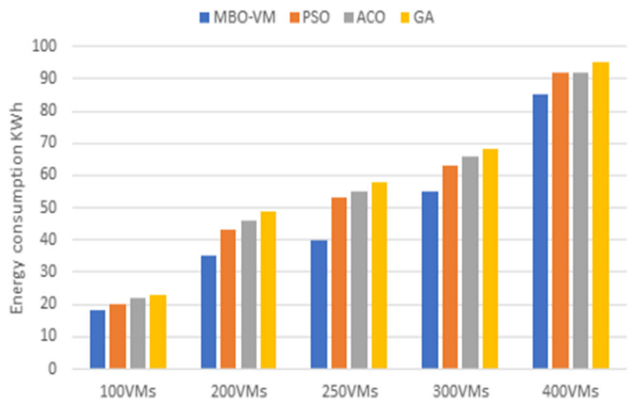

Fig. 4 Average power consumption of all active PMs of HP ProLiant ML110 G4 type

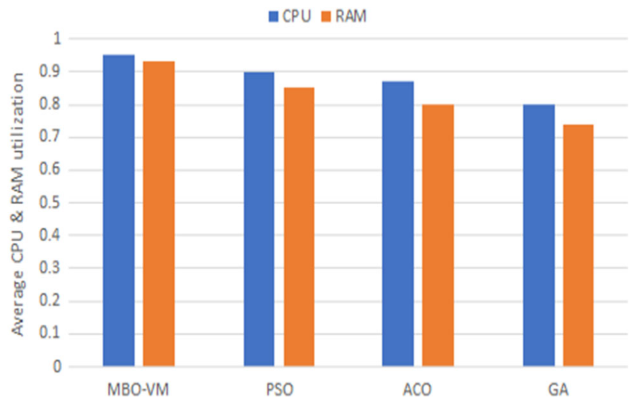

Fig. 5 Average CPU and RAM utilization of all active PMs of HP ProLiant ML110 G4 type

and GA. The configuration of the experiment is similar to the previous experiment.

Figure 4 shows the energy consumption results of the proposed MBO-VM and PSO, ACO and GA. The results show that the proposed MBO-VM can save about 10–2% of energy under light workload and 8–11% of energy under heavy load. For example, for 100 virtual machines, our proposed solution reduces energy consumption to 18 KWh, while the PSO method consumes 20 KWh. The results show that our proposed solution outperforms other methods in terms of energy consumption.

Figure 5 shows the resource utilization of an HP ProLiant ML110 G4. Obviously, the proposed MBO-VM can balance the utilization of CPU and RAM, thereby reducing the waste of resources and the number of active servers. On the one hand, when using the proposed solution, the CPU and RAM utilization results are approximately 0.95 and 0.93, respectively. On the other hand, the resource utilization when applying PSO is 0.90 for CPU and 0.85 for RAM. However, for GA, the resource usage of CPU and RAM is 0.84 and 0.74, respectively. Therefore, the results show that after using the proposed MBO-VMP, CPU resource usage increased by 13% and RAM usage increased by 25%.

6.4 MBO-VM performance using PlanetLab workload in a heterogeneous environment

In Experiment 2, all servers are homogeneous. Because in real environments, physical machines are usually heterogeneous. This experiment uses heterogeneous servers and real workloads to test the performance of the proposed MBO-VM. This workload consists of the resource utilization from 1033 VMs on PlanetLab servers.

Table 4 shows the running time of different methods to find the best allocation strategy for different VM placement with different problem size. Obviously, GA algorithm needs more time. This because in a heterogeneous environment with a large problem scale, the VM placement becomes more difficult. However, compared with other methods, the proposed MBO-VM| algorithm can still find the best VM placement strategy on different problem sizes.

As shown in Table 5, six types of physical machines are used to form three different configurations. For example, in configuration A1, the data center consists of 400 PM and two host types. In A2, the data center consists of 400 PM for four types of hosts. Finally, A3 is configured with six host types and has 400 PM

Figure 6 shows the energy consumption rate for *MBO-VM*, *ACO*, *GA* and *PSO* in three configurations with a PlanetLab workload. The results show that compared with other methods, the proposed *MBO-VM* method can reduce energy consumption by 7 $\sim$ 19%. For example, in *A1*

Table 4 Time (in seconds) to obtain optima with different problem size

No. of VM	MBO-VM Mean (time)	PSO Mean (time)	ACO Mean (time)	GA Mean (time)
100	0.5	0.8	1.4	1.9
200	0.8	1.3	2.1	2.3
250	1.2	2.0	2.5	2.6
300	2.0	2.6	2.8	3.3
400	2.1	3.5	3.4	3.5

Table 5 Configurations used in Experiment 3

Configuration name	Host type
A1	HP Proliant ML 110 G4
	IBM server x3250
A2	HP Proliant ML 110 G4
	IBM server x3250
	HP Proliant ML 110 G5
	IBM server x3550[Xeon-X5675]
A3	HP Proliant ML 110 G4
	IBM server x3250
	HP Proliant ML 110 G5
	IBM server x3550[Xeon-X5675]
	HP Proliant ML 110 G3
	IBM server x3550 [Xeon-X5670]

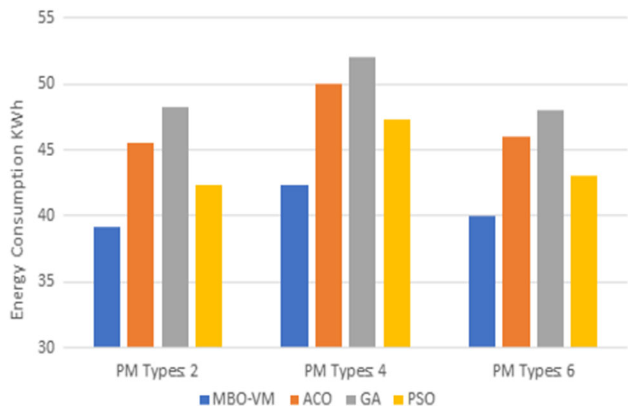

Fig. 6 Energy consumption for all configurations

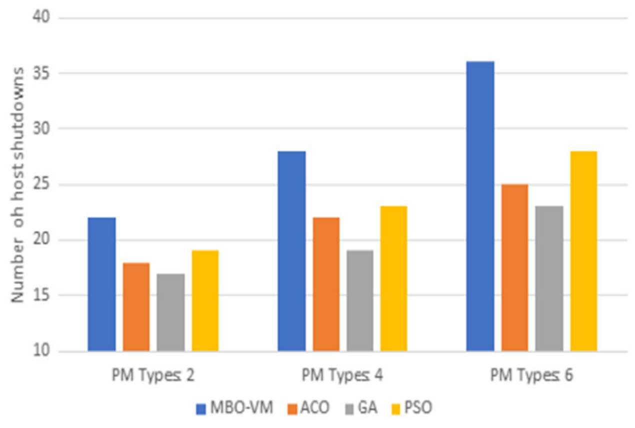

Fig. 7 Number of host shutdowns for all configurations

configuration, *MBO-VM* reduces energy consumption to 39.2 *Kwh*. On the other hand, the energy consumption of *ACO*, *GA* and *PSO* is 45.5, 48.3 and 42.3. Therefore, the results show that the proposed *MBO-VM* can save more energy for all configurations compared with other methods.

Figure 7 shows the number of host shutdowns in all configurations. It is noted that the total host shutdowns in case of the proposed MBO-VM are much higher compared with other solutions for all configurations. For example, in configuration A1, the proposed solution saves approximately 14–18% of the total number of physical machines. In the A3 configuration, the proposed MBO-VMl efficiently balances resource usage with increasing host capacity and shuts down more physical machines. In other words, as the host capacity increases, the proposed MBO-VM increases the number of server shutdowns from 14–18% to 22–31% of the total hosts. As a result, each running physical machine can accommodate more virtual machines and therefore reduces the amount of PM required to serve the workload.

7 Conclusion

With the development of large-scale cloud computing environments, energy consumption has a major impact on the total cost of cloud system. Therefore, this paper proposes MBO-VM method for dynamically assigning VMs to available PMs based on current workload and PM characteristics. The main goal of the MBO-VM is to reduce energy consumption and minimize waste of resources (packaging efficiency). The CloudSim toolkit is used to evaluate the performance of the proposed MBO-VM with real workloads and synthetic workloads. Simulation results show that the proposed MBO-VM makes the data center more energy efficient and balances resource usage compared with other existing VM placement methods. Future research is planned to use the proposed solution to solve other cloud issues, such as dynamic virtual machine migration and task scheduling issues. In addition, it can be applied to other emerging applications, such as image segmentation and medical diagnostics.

Compliance with Ethical Standards

Conflict of interest M. Ghetas declares that he has no conflict of interest.

References

1. Chen Z-G, Du K-J, Zhan Z-H, Zhang J (2015) Deadline constrained cloud computing resources scheduling for cost optimization based on dynamic objective genetic algorithm. In: 2015 IEEE congress on evolutionary computation (CEC), IEEE, pp 708–714

2. Li H-H, Fu Y-W, Zhan Z-H, Li J-J (2015) Renumber strategy enhanced particle swarm optimization for cloud computing resource scheduling. In: 2015 IEEE congress on evolutionary computation (CEC), IEEE, pp 870–876
3. Chen Z-G, Zhan Z-H, Li H-H, Du K-J, Zhong J-H, Foo YW, Li Y, Zhang J (2015) Deadline constrained cloud computing resources scheduling through an ant colony system approach. In: 2015 International conference on cloud computing research and innovation (ICCCRI), IEEE, pp 112–119
4. Li H-H, Chen Z-G, Zhan Z-H, Du K-J, Zhang J (2015) Renumber coevolutionary multiswarm particle swarm optimization for multi-objective workflow scheduling on cloud computing environment. In: Proceedings of the companion publication of the 2015 annual conference on genetic and evolutionary computation, pp 1419–1420
5. Masdari M, Nabavi SS, Ahmadi V (2016) An overview of virtual machine placement schemes in cloud computing. J Netw Comput Appl 66:106–127
6. Gao Y, Guan H, Qi Z, Hou Y, Liu L (2013) A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. J Comput Syst Sci 79(8):1230–1242
7. Braiki K, Youssef H (2018) Multi-objective virtual machine placement algorithm based on particle swarm optimization. In: 2018 14th International wireless communications and mobile computing conference (IWCMC), IEEE, pp 279–284
8. Adamuthe AC, Pandharpatte RM, Thampi GT (2013) Multiobjective virtual machine placement in cloud environment. In: 2013 International conference on cloud and ubiquitous computing and emerging technologies, IEEE, pp 8–13
9. Ghetas M, Yong CH, Sumari P (2015) Harmony-based monarch butterfly optimization algorithm. In: 2015 IEEE International conference on control system, computing and engineering (ICCSCE), IEEE, pp 156–161
10. Ghetas M, Chan HY (2018) Integrating mutation scheme into monarch butterfly algorithm for global numerical optimization. Neural Comput Appl. https://doi.org/10.1007/s00500-020-05381-x
11. Vogels W (2008) Beyond server consolidation. Queue 6(1):20–26
12. Cardosa M, Singh A, Pucha H, Chandra A (2012) Exploiting spatio-temporal tradeoffs for energy-aware mapreduce in the cloud. IEEE Trans Comput 61(12):1737–1751
13. Greenberg A, Hamilton J, Maltz DA, Patel P (2008) The cost of a cloud: research problems in data center networks. ACM, New York
14. Xiao Z, Chen Q, Luo H (2012) Automatic scaling of internet applications for cloud computing services. IEEE Trans Comput 63(5):1111–1123
15. Sahu Y, Pateriya R, Gupta RK (2013) Cloud server optimization with load balancing and green computing techniques using dynamic compare and balance algorithm. In: 2013 5th International conference and computational intelligence and communication networks. IEEE, pp 527–531
16. Amokrane A, Zhani MF, Langar R, Boutaba R, Pujolle G (2013) Greenhead: virtual data center embedding across distributed infrastructures. IEEE Trans Cloud Comput 1(1):36–49
17. Lawey AQ, El-Gorashi TE, Elmirghani JM (2014) Distributed energy efficient clouds over core networks. J Lightw Technol 32(7):1261–1281
18. Speitkamp B, Bichler M (2010) A mathematical programming approach for server consolidation problems in virtualized data centers. IEEE Trans Serv Comput 3(4):266–278
19. Chaisiri S, Lee B-S, Niyato D (2009) Optimal virtual machine placement across multiple cloud providers. In: 2009 IEEE Asia-Pacific services computing conference (APSCC), IEEE, pp 103–110
20. Alicherry M, Lakshman T (2013) Optimizing data access latencies in cloud systems by intelligent virtual machine placement. In: 2013 Proceedings IEEE INFOCOM, IEEE, pp 647–655
21. Dang HT, Hermenier F (2013) Higher SLA satisfaction in datacenters with continuous VM placement constraints. In: Proceedings of the 9th workshop on hot topics in dependable systems, pp 1–6
22. Goudarzi H, Pedram M (2012) Energy-efficient virtual machine replication and placement in a cloud computing system. In: 2012 IEEE Fifth international conference on cloud computing, IEEE, pp 750–757
23. Feller E, Rilling L, Morin C (2011) Energy-aware ant colony based workload placement in clouds. In: 2011 IEEE/ACM 12th international conference on grid computing, IEEE, pp 26–33
24. Ferdaus MH, Murshed M, Calheiros RN, Buyya R (2014) Virtual machine consolidation in cloud data centers using ACO metaheuristic. In: European conference on parallel processing, Springer, pp 306–317
25. Green MI (2010) Cloud computing and its contribution to climate change. In: Greenpeace international, vol 83
26. Dong J-K, Wang H, Li Y, Cheng S (2014) Virtual machine placement optimizing to improve network performance in cloud data centers. J China Univ Posts Telecommun 21(3):62–70
27. Ma F, Liu F, Liu Z (2012) Multi-objective optimization for initial virtual machine placement in cloud data center. J Inf Comput Sci 9(16):5029–5038
28. Dashti SE, Rahmani AM (2016) Dynamic VMs placement for energy efficiency by PSO in cloud computing. J Exp Theor Artif Intell 28(1–2):97–112
29. Wang S, Liu Z, Zheng Z, Sun Q, Yang F (2013) Particle swarm optimization for energy-aware virtual machine placement optimization in virtualized data centers. In: 2013 International conference on parallel and distributed systems, IEEE, pp 102–109
30. Kumar D, Raza Z (2015) A PSO based VM resource scheduling model for cloud computing. In: 2015 IEEE international conference on computational intelligence and communication technology, IEEE, pp 213–219
31. Sharma NK, Reddy GRM (2016) Multi-objective energy efficient virtual machines allocation at the cloud data center. IEEE Trans Serv Comput 12(1):158–171
32. Reddy VD, Gangadharan G, Rao GSV (2019) Energy-aware virtual machine allocation and selection in cloud data centers. Soft Comput 23(6):1917–1932
33. Abdessamia F, Tai Y, Zhang WZ, Shafiq M (2017) An improved particle swarm optimization for energy-efficiency virtual machine placement. In: 2017 International conference on cloud computing research and innovation (ICCCRI), IEEE, pp 7–13
34. Wang S, Gu H, Wu G (2013) A new approach to multi-objective virtual machine placement in virtualized data center. In: 2013 IEEE eighth international conference on networking, architecture and storage, IEEE, pp 331–335
35. Liu C, Shen C, Li S, Wang S (2014) A new evolutionary multi-objective algorithm to virtual machine placement in virtualized data center. In: 2014 IEEE 5th International conference on software engineering and service science, IEEE, pp 272–275
36. Yang T, Lee YC, Zomaya AY (2014) Energy-efficient data center networks planning with virtual machine placement and traffic configuration. In: 2014 IEEE 6th international conference on cloud computing technology and science, IEEE, pp 284–291
37. Li Z, Yu X, Yu L, Guo S, Chang V (2020) Energy-efficient and quality-aware VM consolidation method. Future Gener Comput Syst 102:789–809
38. Qin Y, Wang H, Yi S, Li X, Zhai L (2020) Virtual machine placement based on multi-objective reinforcement learning. Appl Intell. https://doi.org/10.1007/s10489-020-01633-3

39. Farzai S, Shirvani MH, Rabbani M (2020) Multi-objective communication-aware optimization for virtual machine placement in cloud datacenters. Sustain Comput Inform Syst. https://doi.org/10.1016/j.suscom.2020.100374
40. Shaw R, Howley E, Barrett E (2019) An energy efficient anti-correlated virtual machine placement algorithm using resource usage predictions. Simul Model Pract Theory 93:322–342
41. Moges FF, Abebe SL (2019) Energy-aware VM placement algorithms for the OpenStack Neat consolidation framework. J Cloud Comput 8(1):2
42. Abdel-Basset M, Abdle-Fatah L, Sangaiah AK (2019) An improved Lévy based whale optimization algorithm for bandwidth-efficient virtual machine placement in cloud computing environment. Cluster Comput 22(4):8319–8334
43. Gao Y, Guan H, Qi Z, Wang B (2012) An ant colony system algorithm for the problem of server consolidation in virtualized data centers. J Comput Inf Syst 8(16):6631–6640
44. Deepika T, Prakash P (2020) Power consumption prediction in cloud data center using machine learning. Int J Electr Comput Eng (IJECE) 10(2):1524–1532
45. Joines JA, Houck CR (1994) On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's. In: Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence, IEEE, pp 579–584
46. Ghetas M, Yong CH (2018) Resource management framework for multi-tier service using case-based reasoning and optimization algorithm. Arab J Sci Eng 43(2):707–721
47. Simon D (2013) Evolutionary optimization algorithms. Wiley, New York
48. Feng Y, Wang G-G, Li W, Li N (2018) Multi-strategy monarch butterfly optimization algorithm for discounted 0–1 knapsack problem. Neural Comput Appl 30(10):3019–3036
49. Feng Y, Wang G-G, Deb S, Lu M, Zhao X-J (2017) Solving 0–1 knapsack problem by a novel binary monarch butterfly optimization. Neural Comput Appl 28(7):1619–1634
50. 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