Hindawi
Applied Computational Intelligence and Soft Computing
Volume 2021, Article ID 8892734, 14 pages
https://doi.org/10.1155/2021/8892734

Research Article

Toward Enhancing the Energy Efficiency and Minimizing the SLA Violations in Cloud Data Centers

E. I. Elsedimy[1] **and Fahad Algarni**[2]

[1]*Department of System and Information Technology, Faculty of Management Technology and Information System, Port Said University, Port Fuad 42526, Egypt*
[2]*Faculty of Computing and Information Technology, University of Bisha, Bisha 61922, Saudi Arabia*

Correspondence should be addressed to E. I. Elsedimy; sayed_elsedimy@yahoo.com

Received 17 August 2020; Revised 7 December 2020; Accepted 22 December 2020; Published 13 January 2021

Academic Editor: Miin-Shen Yang

Copyright © 2021 E. I. Elsedimy and Fahad Algarni. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Recently, the problem of Virtual Machine Placement (VMP) has received enormous attention from the research community due to its direct effect on the energy efficiency, resource utilization, and performance of the cloud data center. VMP is considered as a multidimensional bin packing problem, which is a type of NP-hard problem. The challenge in VMP is how to optimally place multiple independent virtual machines into a few physical servers to maximize a cloud provider's revenue while meeting the Service Level Agreements (SLAs). In this paper, an effective multiobjective algorithm based on Particle Swarm Optimization (PSO) technique for the VMP problem, referred to as VMPMOPSO, is proposed. The proposed VMPMOPSO utilizes the crowding entropy method to optimize the VMP and to improve the diversity among the obtained solutions as well as accelerate the convergence speed toward the optimal solution. VMPMOPSO was compared with a simple single-objective algorithm, called First-Fit-Decreasing (FFD), and two multiobjective ant colony and genetic algorithms. Two simulation experiments were conducted to verify the effectiveness and efficiency of the proposed VMPMOPSO. The first experiment shows that the proposed algorithm has better performance than the algorithms we compared it to in terms of power consumption, SLA violation, and resource wastage. The second indicates that the Pareto optimal solutions obtained by applying VMPMOPSO have a good distribution and a better convergence than the comparative algorithms.

1. Introduction

Cloud computing provides a promising approach through which pools of services are delivered to users over the Internet [1]. It can deliver infrastructures, platforms, and applications as hosted services to large numbers of users in the pay-as-you-go model [2]. Cloud users can purchase as more or as less computing resources as they need, since the resources of the cloud appear to be unlimited to them. To meet the high-performance expectations of cloud users in a cost-effective manner, cloud providers must consider several key challenges, such as large power consumption, high resource wastage, and high Service Level Agreement (SLA) violation [3]. In this context, energy management in cloud data centers has been a crucial issue since it has a direct effect on operating costs [3]. Statistics show that the infrastructure and power costs contribute about 75%, while the information technology contributes just 25% of the total cost of operating a cloud data center [4]. This high increase in power consumption is not only due to the badness of computing resources and the energy inefficiency of hardware but also due to the inefficient usage of the available resources. The gathered data from more than 5000 physical servers over six months have shown that although the majority of these servers are not idle most of the time, the utilization rarely reaches 100% [5]. Moreover, the active servers usually operate at 10–50% of their capacity, which leads to great waste of resources and power. Therefore, it is extremely important to reduce the large power consumption caused by idle servers to maximize a provider's revenue while satisfying the Quality of Service (QoS) delivered to cloud consumers [6].

One of the ways to address the power inefficiency and ineffective resource utilization of cloud data centers is to employ the capability of the virtualization [7] technologies to create multiple virtual machines (VMs) on a single server and turn off the idle server. With the support of virtualization, the resources of a single physical machine such as the amounts of CPUs, main memory, storage, and network bandwidth are partitioned into multiple execution environments for several VMs. Virtual Machine Placement (VMP) [8, 9] is an important issue in cloud environment virtualization, especially in the infrastructure as a service (IaaS) [10] model. VMP maps a set of VMs into a suitable set of physical servers. However, calculating all possible VM-server mappings in the cloud data center environment and selecting the best one is not feasible, since the complexity will grow quickly with the number of VMs and servers. Furthermore, different VM placement solutions may leave different numbers of physical resources and consume different amounts of power [11]. Therefore, how to optimally place virtual machines into the cloud data centers to minimize power consumption and resource wastage while ensuring the QoS delivered to cloud users is still a challenge for researchers.

Up to now, several techniques have been applied to address the problem of VMP in virtualized data center's infrastructure, such as Linear Programming (LP) [12], random greedy algorithm [13], heuristic bin packing [14], simulated annealing optimization [15], constraint programming [16], Genetic Algorithm (GA) [17], and a classical Ant Colony Optimization (ACO) algorithm [18]. Another approach to optimize VM placement is achieved based on a generalization of the Knapsack Problem [19]. The VM section algorithm attempts to automatically adjust its behavior depending on the workload of the submitted process. Although these approaches may found the solution for the VMP problem, they are easy to drop into local extreme solutions. Also, they focus on optimizing a single target such as power consumption, SLA, load balancing, or resource utilization.

In this work, we propose a new algorithm for the VMP problem, called VMPMOPSO. The proposed algorithm combines the principles of the PSO technique and multiobjective optimization. It can be divided into four parts: (1) a suitable encoding scheme for the VMP problem is designed to map all possible solutions to the particle space; (2) the fitness values of all particles are evaluated; (3) the Pareto dominance rules are utilized to establish the preferences among the generated solutions; after that, the nondominated solutions are put into archive set; (4) finally, the crowding entropy method is used to promote the diversity among nondominated solutions. Compared to two existing multiobjective VMP algorithms presented in [20, 21] and a single-objective algorithm proposed in [22], the proposed algorithm has better performance in terms of power consumption, resource wastage, and SLA violation. Furthermore, the obtained solutions by the VMPMOPSO algorithm have better distribution and convergence than the existing algorithms.

The remainder of this paper is organized as follows. In Section 2, the recent related works in VMP approaches are discussed. The proposed management framework for VMP in the cloud data center's infrastructure is described in Section 3. After that, Section 4 presents the problem statement and formulates the VMP problem. In Section 5, we provide background knowledge of the proposed VMPMOPSO algorithm. The simulation results and evaluations are given in Section 6. Finally, the conclusions and future work for this paper are presented in Section 7.

2. Related Work

In recent years, the problem of VMP has received significant attention, and a lot of studies have been conducted to overcome the limitations of conventional VMP approaches [23–27]. Babu and Samuel [28] propose a bin packing-based approach to optimize the VMP in the cloud data center environment. In this approach, each physical machine is considered as bins and the VMs as the targets to be inserted in the free bin. The goal is to minimize energy consumption by placing VMs into a few hosts. In [29], Verma et al. introduce a modified version of the First-Fit-Decreasing (FFD) algorithm in order to map VMs into the server that has enough resources to host them. Their objective is to minimize resource wastage by reducing the fragment leak that results from placing more than one VM in a single host. However, the authors did not discuss the violations in SLA due to VM allocation. Recently, Beloglazov and Buyya [30] present two novel algorithms, which can participate with each other in developing a dynamic VM consolidation approach. They firstly propose a Modified Best-Fit-Decreasing (MBFD) algorithm, which regards the problem of VMP as a bin backing problem. This algorithm can be divided into two parts: (1) VMs are sorted in decreasing order according to their current CPU utilization; (2) VMs are mapped into the servers that have enough resources and provide the minimum energy consumption due to this mapping. Finally, for eliminating servers' overloads, they apply a Minimum Migration (MM) algorithm, which aims to control VMs' migration among servers by applying the upper and lower threshold of their CPU utilization. The results show that the proposed algorithms outperform the other existing policies. However, the proposed MBFD algorithm often is basically the single point search, which concentrates only on the local solution and ignores the global optimum placement solution in a large-scale data center.

Xu and Fortes [11] proposed an improved GA based on fuzzy multiobjective evaluation in order to effectively deal with a large solutions search space of a VMP problem. Their objective is to reduce power consumption, efficiently utilize multidimensional resources, and avoid hotspots. The authors also design a two-level control system for managing resources automatically in virtualized data centers. At the virtual-container level, in order to improve the SLA rate, a local controller is utilized to determine the number of resources demanded by applications and requests additional resources to avoid the degradation in performance. At the

resource-pool level, the global controller is applied to manage VM placement and resource allocation, in which the physical resources are allocated as slices based on the demands of VM requests. Although the proposed approach achieves better results when compared to the other existing algorithms, it has more time complexity results from solving the multidimensional resources allocation based on the multidimensional matrix.

Liu et al. [31] formulated the VMP problem as a multiobjective optimization problem and solved it by using an improved evolutionary multiobjective optimization algorithm named NS-GGA. This algorithm incorporates the fast nondominated sorting of NSGA-II into the grouping GAs. The analysis of the results shows that the proposed NS-GGA outperforms the existing algorithms. However, it focuses only on solving the VMP problem to obtain the Pareto optimal front while issues related to how to improve the diversity among the obtained solutions of the VMP problem have less attention in this work.

Gao et al. [21] regard the problem of VMP in the virtualized data center as a multiobjective optimization problem and solve it by using an improved ACO system algorithm. The goal is to find a set of Pareto optimal solutions that improve resource utilization and energy efficiency together. However, how to guarantee users' SLA has not been discussed in the paper.

Zheng et al. [32] presented a novel approach for the VMP problem called VMPMBBO. The proposed approach applied a biogeography-based optimization to ensure convergence to the optimum solution in the large solutions' search space of the VMP optimization problem. Their objective is to improve cloud data center efficiency by minimizing the total power consumption and resource wastage simultaneously. However, the influence of VMP operation on SLA violation has not been considered in their paper.

In [33], Kumar and Raza adapted the PSO technique-based VM scheduling strategy in order to minimize the total resource wastage and psychical server utilizations. The proposed work achieves better results compared to traditional strategies such as Best-Fit, First-Fit, and Worst-Fit placement. However, this method can be trapped in local optima.

Recently, Dashti and Rahman in [34] proposed a modified PSO to reallocate migrated VMs in the overloaded host. However, they considered only the server utilization and the migration cost to get the optimal solution for the VM placement process. Tripathi et al. [35] formulate the VM placement process as a multiobjective optimization problem with better energy consumption and resource usage. However, they ignore the SLA violations of the physical server.

As discussed above, many approaches have been proposed by the researchers for addressing the VMP problem but none of them have considered all the issues of such problem which ensure the QoS delivered to the users and at the same time maximize the cloud provider's revenue. Therefore, the development of a novel algorithm for the VMP problem is required in order to reduce the operating cost to the cloud provider and at the same time provide good services to the consumers.

3. System Architecture and Problem Formulation

This section presents the system architecture for VM management in the cloud environment and formulates the VM placement as a multiple-objective optimization problem.

3.1. System Architecture. On the cloud computing IaaS platform, the computing resources are provided to consumers as a service and by needs, and consumers can buy as much or as little computing power as they demand. Cloud users can access computing resources via any electronic device connected to the Internet, such as a laptop, PC, PDA, and mobile phone. Today, many IaaS providers such as Google Compute Engine [36] and Amazon AWS [37] provide users with an exclusive infrastructure in the form of VM instances. They offer several types of VM instances with varying configurations and different prices. As there are a large number of users who request VM instances simultaneously, so how to optimally place these VMs on the cloud data centers to maximize the cloud provider's profit while satisfying users' requirements is a significant problem to solve. To address this issue, we present our proposed system architecture for VM management in the cloud environment as shown in Figure 1.

On cloud data centers, first, cloud users request VMs from a cloud IaaS provider using a VM description template. The configuration of each VM is defined in terms of physical resource metrics such as CPUs cores, disk storage, memory capacity, and network bandwidth. Second, the VM scheduler accepts and handles the incoming requests from cloud users for renting the VM instances. At the same time, two actions were performed automatically: provisioning of virtual resources and monitoring of these resources. Therefore, in this paper, we employ an optimization policy coupled with our proposed algorithm to optimize the placement of VMs into different servers taken into account the conflicting goals of cloud providers and consumers. The objective of the proposed algorithm is to find a set of Pareto optimal solutions that optimize multiobjectives such as SLA violations, power consumption, and resource wastage. To improve the diversity among the obtained solutions and accelerate the convergence speed toward the optimal solution, the proposed algorithm utilizes a crowding entropy mechanism to estimate the crowding distance around each obtained solution accurately.

3.2. Problem Formulation. The cloud IaaS providers periodically receive a large number of computing resource requests, such as CPU, main memory, storage, and network bandwidth, from different users. These resources are packed as VMs and then be placed on different servers at the cloud provider infrastructure based on specific constraints, such as meeting the SLA requirements, improving the energy efficiency, and reducing the wastage of resources. Here, both the VM's resources and the server's resources are represented by a multidimensional vector. We use two dimensions to

represent a VM and a server-CPU and memory. The resources of each server can be expressed using $R_k = \{R_k^{\text{cpu}}, R_k^{\text{mem}}\}, 1 \le k \le m$, where m represents the total number of servers, R_k^{cpu} represents the capacities of CPU, and R_k^{mem} represents the size of main memory. Similarly, the resources requested by each VM can be represented by the vector $V_i = \{V_i^{\text{cpu}}, V_i^{\text{mem}}\}, 1 \le i \le n$, where n is the total number of VMs requests. We use a binary variable x_k to refer to whether server k is used ($x_k = 1$) or not ($x_k = 0$). Additionally, by assuming that a single server is able to host multiple independent VMs, we define the decision variable y_{ik} which represents whether VM i is mapped to server k ($y_{ik} = 1$) or not ($y_{ik} = 0$).

3.2.1. SLA Violation Modeling. In the cloud computing environment, placing the VMs into a suitable host that provides the least violation in SLA is extremely important for both cloud providers and consumers. Beloglazov and Buyya [38, 39] define the SLA violation due to VM allocation as a fraction of the difference between the total demands of processing units of CPU resources by all virtual machines and the actually allocated processing units of CPU relatively to the total requested of processing units of CPU over the lifetime of the virtual machines. However, these studies only consider the performance degradation due to the lack of CPU resources, rather than other resources such as storage, main memory, and network bandwidth. Here, we quantify the SLA violation taking into account a two-dimensional vector of resources (CPU and memory), rather than a CPU resource only, as shown in the following:

$$\text{SLA violation} = \begin{cases} \text{SALV}_k^{\text{cpu}} = \dfrac{\sum_{i=1}^{n}\left(D_{ik}^{\text{cpu}} - A_{ik}^{\text{cpu}}\right)}{\sum_{i=1}^{n} D_{ik}^{\text{cpu}}}, \\ \text{SLAV}_k^{\text{mem}} = \dfrac{\sum_{i=1}^{n}\left(D_{ik}^{\text{mem}} - A_{ik}^{\text{mem}}\right)}{\sum_{i=1}^{n} D_{ik}^{\text{mem}}}, \\ \text{SLAV}_k = \dfrac{\text{SALV}_k^{\text{cpu}} + \text{SLAV}_k^{\text{mem}}}{2}, \end{cases} \tag{1}$$

where D_{ik}^{cpu} and D_{ik}^{mem} are demands of CPU and main memory resources by VM i carried on physical machine k, respectively, $1 \le i \le n$, and n represents the total number of VMs. Similarly, A_{ik}^{cpu} and A_{ik}^{mem} are the amounts of CPU and main memory resources allocated to VM i.

Based on the above definition of SLA violation, SLA violation (k) is introduced to quantify the SLA violations caused by physical machine k, as shown in

$$\text{SLA violation}(k) = x_k \times \left(\frac{\left(\left(\sum_{i=1}^{n} y_{ik} \times \left(D_{ik}^{\text{cpu}} - A_{ik}^{\text{cpu}}\right)\right) / \left(\sum_{i=1}^{n} D_{ik}^{\text{cpu}}\right)\right) + \left(\left(\sum_{i=1}^{n} y_{ik} \times \left(D_{ik}^{\text{mem}} - A_{ik}^{\text{mem}}\right)\right) / \left(\sum_{i=1}^{n} D_{ik}^{\text{mem}}\right)\right)}{2} \right). \tag{2}$$

3.2.2. Resource Wastage Modeling. The residual resources available on each physical server may vary greatly with different VMP solutions. Therefore, it is necessary to develop a high-effective VMP approach to increase resource utilization of the physical server and keep the remaining resources on each server balanced along different dimensions. Here, we define the resource wastage W_k as the average value between different dimension residual resources (CPU and memory) and the whole physical machine resources, as seen in the following:

$$\text{resource wastage} = \begin{cases} W_k^{\text{cpu}} = \dfrac{k^{T\text{cpu}}}{k^{\text{cpu}}}, \\ W_k^{\text{mem}} = \dfrac{k^{T\text{mem}}}{k^{\text{mem}}}, \\ W_k = \dfrac{W_k^{\text{cpu}} + W_k^{\text{mem}}}{2}. \end{cases} \tag{3}$$

Based on the above consideration, resource wastage (k) is introduced to calculate the total wasted resources of the k-th server, as seen in

$$\text{resource wastage}(k) = x_k \times \left(\sum_{i=1}^{n} y_{ik} \times \frac{\left(k^{T\text{cpu}} / K^{\text{cpu}}\right) + \left(k^{T\text{mem}} / K^{\text{mem}}\right)}{2} \right). \tag{4}$$

3.2.3. Power Consumption Modeling. Recent studies [40] show that the total power consumption of the physical server is linearly proportional to CPU utilization. These studies indicate that an idle server consumes approximately 70% of the energy consumed when the CPU is fully utilized. Therefore, in order to save power, idle servers should be turned off. Finally, as seen in equation (5), power consumption (k) is presented to compute the total power consumption by k-th server.

$$\text{power consumption}(k) = x_k \times \left(\left(P_k^{\text{busy}} - P_k^{\text{idel}}\right) \times \sum_{i=1}^{n} \left(y_{ik} \times D_{ik}^{\text{cpu}}\right) + P_k^{\text{idel}} \right), \tag{5}$$

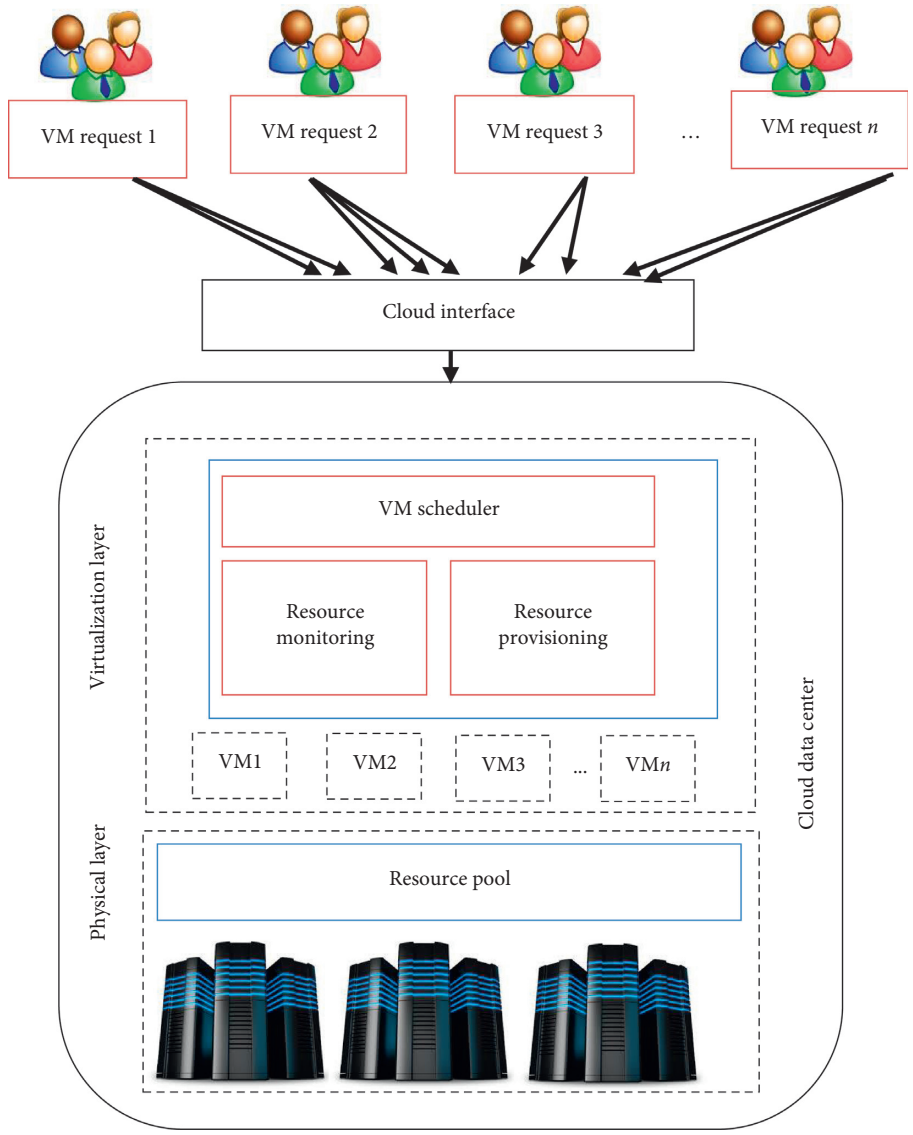

Figure 1: The system architecture for VM management in the cloud environment.

where P_k^{idel} and P_k^{busy} refer to the power consumption by server k at idle and full load, respectively. D_{ik}^{cpu} represents the number of CPU demands by VM i carried on server k. In our simulation experiments, P_k^{idel} and P_k^{busy} are set to 162 and 215 Watt, respectively.

3.2.4. Optimization Formulation. The VMP problem is described as follows. Assume that there are n of VMs which are to be placed on m of servers in the cloud data center, and none of these VMs require that computing resources exceed the capacity of a single server. The goal is to simultaneously minimize SLA violations, wastage of resources, and power consumption. Therefore, we formulate the VMP problem as follows:

$$\text{minimize: } \sum_{k=1}^{m} \text{SLA violation}(k), \tag{6}$$

$$\text{minimize: } \sum_{k=1}^{m} \text{resource wastage}(k), \tag{7}$$

$$\text{minimize: } \sum_{k=1}^{m} \text{power consumption}(k), \tag{8}$$

subject to

$$\sum_{k=1}^{m}\sum_{i=1}^{n} y_{ik} = 1, \quad \forall i = 1,\ldots,n; k = 1,\ldots,m, \tag{9}$$

$$\sum_{k=1}^{m}\sum_{i=1}^{n} y_{ik}\cdot V_{ik}^{\text{cpu}} \le \sum_{k=1}^{m} x_k \cdot R_K^{\text{cpu}}, \tag{10}$$

$$\sum_{k=1}^{m}\sum_{i=1}^{n} y_{ik}\cdot V_{ik}^{\text{ram}} \le \sum_{k=1}^{m} x_k \cdot R_K^{\text{ram}}. \tag{11}$$

The objectives in equations (6) and (7), respectively, aim to minimize the SLA violation and resource wastage by all the physical servers, and the objective in equation (8) aims to reduce the total power consumption. The constraint in equation (9) ensures that each VM is mapped into one, and only one, host. Constraints in equations (10) and (11) ensure that the requested resources by all VMs do not exceed the full capacity of the available server's resources.

4. An Introduction to the Multiobjective Improved PSO

In this section, we firstly provide an overview of PSO algorithm principles and then present background knowledge of multiobjective optimization. Finally, we give a brief description of a Multiobjective Particle Swarm Optimization (MOPSO) algorithm coupled with the crowding entropy mechanism to optimize the VMP.

4.1. Particle Swarm Optimization Algorithm. The *Particle Swarm Optimization (PSO)* is a popular computational technique [41] inspired by the social behavior of fish schooling and bird flocking to seek foods. Recent studies [42, 43] show that PSO performs better in distributed and grid computing environments compared with GA. The PSO also can achieve better convergence on the search than GA [42]. Furthermore, Reyes-Sieraa and Coello [44] note that the PSO uses simple operations with few parameters to produce very good results at a low computational cost.

In PSO, a population of all possible solutions, called particles, was firstly initialized. Then, these particles are allowed to fly with an adaptable velocity searching for the optimum solution in the solution search space. Each particle can store the best location obtained so far (locally optimal), as well as the best position tracked by the entire swarm (global optimal). Here, we use the quadruple $\langle S_i, V_i, P_i, G_i\rangle$ to represent each particle in the swarm, where $S_i = (S_{i1}, S_{i2}, \ldots, S_{id}, \ldots)$ and $V_i = (V_{i1}, V_{i2}, \ldots, V_{id}, \ldots)$ are the position and velocity vectors of the ith particle, respectively.$P_i = (P_{i1}, P_{i2}, \ldots, P_{id}, \ldots)$ denotes the local best position obtained by ith particle and $G_i = (G_{i1}, G_{i2}, \ldots, G_{id}, \ldots)$ represents the global optimum position searched by the entire swarm, where d represents the dimension of particles.

At each iteration t, the PSO algorithm updates particle's velocity and position-based equations (12) and (13), respectively.

$$\begin{aligned} V_{id}(t+1) = {} & \phi\cdot V_{id}(t) + \psi_1\cdot\vartheta_1\cdot\left[P_{id}(t) - S_{id}(t)\right] \\ & + \psi_2\cdot\vartheta_2\cdot\left[P_{gd}(t) - S_{id}(t)\right], \end{aligned} \tag{12}$$

$$S_{id}(t+1) = S_{id}(t) + V_{id}(t+1), \tag{13}$$

where ψ_1 and ψ_2 are the acceleration coefficients. The parameters ϑ_1 and ϑ_2 are positive random numbers uniformly distributed between 0 and 1. ϕ is the inertia weight, $0 \le \phi \le 1$, utilized to control the unlimited growth of the particle's velocity. The inertia weight is defined as follows:

$$\phi = \phi_{\max} - \frac{\phi_{\max} - \phi_{\min}}{t_{\max}} \times t, \tag{14}$$

where $\phi_{\max}$ and $\phi_{\min}$ denote the maximum and minimum inertia's values, respectively, t denotes the current iteration value, and $t_{\max}$ denotes the maximum number of iterations, where the higher the value of $t_{\max}$ is, the smaller the weight ϕ will be.

4.2. Definition of Multiobjective Optimization. Most real-world problems have more than one of the objectives which are usually in conflict with each other, and optimizing these objectives simultaneously is very difficult. Multiobjective Optimization (MOO) addresses this problem. Many multiobjective evolutionary algorithms have been developed to solve MOO problems such as NSGA-II [45], SEPA [46], and MOPSO [47] algorithms. The experimental results in [48] show that the MOPSO algorithm performs better than traditional multiobjective algorithms for MOO problems. Also, it can achieve less computing time and high searching ability than PAES and NSGA-II algorithms. Since the first effort is introduced in [49] by Parsopoulos and Vrahatis to employ the principles of the traditional PSO algorithm in MOO problems, many authors have presented different attempts to improve the performance of MOPSO, for example, Durillo et al. [50], Nebro et al. [51], and Moubayed et al. [52]. The majority of these efforts focused on improving convergence and diversity among the generated solutions. The work in [53] incorporates a nondominated sorting mechanism in the MOPSO algorithm, for improving the global searching scale. His mechanism started well, but with the large size of search space, controlling the number of the best solution is difficult and the mechanism may be tended to drop into local optima. To avoid the MOPSO fall into a local solution, Li [54] suggested using a maximin strategy coupled with a random inertia weight. In this strategy, the

maximin fitness function is utilized to sort solutions and determine Pareto dominance without the need for clustering or niching procedure. Although the proposed strategy improves the diversity of the swarm, its power in global searching is not satisfied.

The target of MOO is to find those values of an n-dimensional decision variable vector $(x_1, \ldots, x_n)^T$ for which the values of an m-dimensional objective vector $f(x_1, \ldots, x_m)^T$ are minimized (or maximized). The best solutions for the MOO problem are called Pareto optimal solutions, which result from the preference among several objectives. MOO problem can be described in the following manner:

$$\begin{cases} \min & y = F(x) = [f_1(x), f_2(x), \ldots, f_m(x)]^T, \\ \text{s.t.} & g_i(x) \le 0,\ i = 1, 2, \ldots, Q, \\ & h_j(x) = 0,\quad j = 1, 2, \ldots, P, \end{cases} \tag{15}$$

where $x = (x_1, \ldots, x_n)^T \in X \subset R^n$ is known as a decision vector, X represents the set of feasible solutions, $y = (y_1, \ldots, y_m)^T \in Y$ is known as an objective vector, and Y is an objective space.$g_i(x)$ refers to the ith inequality constraint while $h_j(x)$ is the jth equality constraint. As discussed above, we search for Pareto optimum. More formally, we have the following.

Definition 1 (dominance). Given two decision vectors $\overrightarrow{u} = (u_1, \ldots, u_m) \in X$ and $\overrightarrow{v} = (v_1, \ldots, v_m) \in X$, we say that $\overrightarrow{u}$ dominates $\overrightarrow{v}$, denoted by $\overrightarrow{u} \prec \overrightarrow{v}$, if and only if (if) $\forall j \in \{1, 2, \ldots, m\}$: $f_j(\overrightarrow{u}) \le f_j(\overrightarrow{v})$ and $\exists i \in \{1, 2, \ldots, m\}$: $f_i(\overrightarrow{u}) \le f_i(\overrightarrow{v})$.

Definition 2 (Pareto optimality). A vector $\overrightarrow{u}$ is Pareto optimal solution with respect to X if, $\nexists \overrightarrow{v} \in X$: $\overrightarrow{v} \succ \overrightarrow{u}$.

Definition 3 (Pareto optimal set). P^* is known as the Pareto optimal set that contains all Pareto optimal solutions and defined as

$$P^* = \{\overrightarrow{v} \in tX | n \nexists q \overrightarrow{v} h \in {}_X x\colon 7\overrightarrow{v}C \succ; \overrightarrow{u}\}. \tag{16}$$

4.3. Crowding Distance-Based External Archive. To improve the diversity among the obtained solution by the MOPSO algorithm and enhance the searching capability, a good crowding distance method is required to measure the crowding degree around each generated solution in the external archive accurately. Several methods have been applied to estimate the crowding degree, such as an adaptive hypercube used in MOPSO [47], the density estimation strategy used with SPEA2 in [55], and a crowding distance method proposed in [45]. These methods have effectively contributed to the MOPSO algorithm, but the diversity preservation for the obtained Pareto optimal set and the global optimum update strategy are very complex, leading to high time complexity. Here, the crowding entropy method, proposed by Wang et al. in [56], is incorporated in MOPSO to find the perfect solutions with good diversity. This method coupled the entropy distribution with the crowding distance strategy in order to estimate the density around each particle more accurately. It can be defined as follows:

$$\mathrm{CE}_i = \sum_{j=1}^{m} \frac{\left(\beta_{ij} H_{ij}\right)}{\left(f_j^{\max} - f_j^{\min}\right)}, \tag{17}$$

where

$$\begin{aligned} H_{ij} &= -\left[el_{ij} \log_2\left(el_{ij}\right) + eu_{ij} \log_2\left(eu_{ij}\right)\right], \\ el_{ij} &= \frac{\delta l_{ij}}{\beta_{ij}}, \\ eu_{ij} &= \frac{\delta u_{ij}}{\beta_{ij}}, \\ \beta_{ij} &= \delta l_{ij} + \delta u_{ij}, \end{aligned} \tag{18}$$

where $f_j^{\max}$ and $f_j^{\min}$ represent the maximum and minimum values of the jth objective function, respectively, and m is the total number of objectives. δl_{ij} and δu_{ij} represent the distances of the ith particle to its lower and upper closest neighbor particles along the jth objective function, respectively.

Finally, to extend the crowding entropy method to MOPSO, the most important issue is the modification of the preferred method. Therefore, the preference rules of MOPSO can be defined in the following manner:

$$\{\overrightarrow{u}, \overrightarrow{u} \prec \overrightarrow{v}, \overrightarrow{v}, \overrightarrow{v} \prec \overrightarrow{u}, LC(\overrightarrow{u}, \overrightarrow{v}), \overrightarrow{v} \overline{\prec} \overrightarrow{u} \wedge \overrightarrow{v} \overline{\prec} \overrightarrow{u}, \tag{19}$$

where $LC(\overrightarrow{u}, \overrightarrow{v})$ represents the less crowded distance between $\overrightarrow{u}$ and $\overrightarrow{v}$ decision vectors. According to the dominance relation between two solution vectors $\overrightarrow{u}$ and $\overrightarrow{v}$, there are at most three cases: (1) $\overrightarrow{u}$ dominates $\overrightarrow{v}$; (2) $\overrightarrow{v}$ dominates $\overrightarrow{u}$; (3) $\overrightarrow{u}$ and $\overrightarrow{v}$ are nondominated with each other $(\overrightarrow{v} \overline{\prec} \overrightarrow{u} \wedge \overrightarrow{v} \overline{\prec} \overrightarrow{u})$.

5. The Proposed VMPMOPSO Approach

In this section, a Multiobjective Particle Swarm Optimization (MOPSO) algorithm, extended with crowding entropy method, referred to as VMPMOPSO, has been proposed for the problem of VMP. The goal of the VMPMOPSO algorithm is to find the optimal placement solution that effectively minimizes the violation in SLA and resource wastage while saving energy, as well.

The pseudocode of the proposed VMPMOPSO algorithm is listed in Algorithm 1. This algorithm starts its work with the initialization of some parameters, such as the particle's position and velocity, the swarm size, and the maximum iterations. Then, each particle is evaluated based on fitness functions 6, 7, and 8, respectively. The obtained nondominated solutions during the evaluation process are stored in the archive set. By continuing to add the new nondominated solutions into the archive set, the external archive is formed. When there are repeated solutions that

have the same crowding entropy value, the proposed algorithm deletes them and keeps one. After that, the crowding entropy is applied to improve the diversity among the obtained solutions. The solution with the highest crowding entropy value has a better chance to be chosen as the global best. After selecting the global best, the velocity and location of each particle are updated from the iteration t to $t+1$ using equations (12) and (13), respectively. Finally, VMPMOPSO checks the termination condition to ensure if the loop excessed the maximum iteration or not. If the loop excessed the maximum iteration, stop; and return the Pareto set that stores the optimal solutions of the VMP problem; otherwise, continue until the end of a loop.

5.1. Particle Definition. In order to apply the proposed VMPMOPSO algorithm successfully for the VMP problem, it is necessary to develop a high-effective encoding scheme to map the solutions of the VMP problem to the particle space. It is a critical step because the best mapping of the problem solutions into the particle leads to more efficiency and better performance of the VMPMOPSO algorithm. Here, we define the position matrix of particles $S_i(t)$ that represent all possible solutions of the VMP problem as follows:

$$S_i(t) = \begin{bmatrix} y_{11}(t) & y_{12}(t) & \dots & y_{1k}(t) & \dots & y_{1n}(t) \\ y_{21}(t) & y_{22}(t) & \dots & y_{2k}(t) & \dots & y_{2n}(t) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ y_{i1}(t) & y_{i2}(t) & \dots & y_{ik}(t) & \dots & y_{in}(t) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ y_{m1}(t) & y_{m2}(t) & \dots & y_{mk}(t) & \dots & y_{mn}(t) \end{bmatrix}, \quad (20)$$

where i represents the ith possible solutions and t denotes the iteration number. Also, there are n of VMs that are to be placed on m physical servers. We use a decision variable y_{ik} to represent whether VM i is allocated to server k or not. If VM i is allocated to server k, $y_{ik} = 1$; if not, $y_{ik} = 0$.

5.2. Update Strategy for Global Optimum. In VMPMOPSO, the crowding entropy mechanism incorporates into MOPSO specifically on the global optimum selection. The goal is to improve convergence to true Pareto optimal and control the capacity of the archive by deleting the redundant solutions in the archive set. Based on the crowding entropy, the crowding distance values of all nondominated solutions in the archive were calculated and ranked in a descending order. After that, the solutions with the highest crowding values have a better chance to be selected as the global optimum. For example, given two solution vectors $\vec{u}, \vec{v} \in E$, E is an archive set, $|E| = N$, and the solution vector $\vec{u}$ is said to the global optimum along jth objective function if the following conditions satisfied:

(1) If a solution vector $\vec{u}$ has crowding entropy value that dominates a solution vector $\vec{v}$ along the jth objective function, denoted as $CE_{\vec{u}}(j) \prec CE_{\vec{v}}(j)$, then the solution $\vec{u}$ is more crowded than solution $\vec{v}$

(2) If the crowding entropy value of a solution vector $\vec{u}$ dominates all solutions in archive E along the jth objective function, denoted as $CE_{\vec{u}}(j) \prec \sum_{i=1}^{N} CE_i(j)$, then the solution $\vec{u}$ is selected as the global optimum

6. Results and Discussion

In this section, we have conducted two simulation experiments to verify the effectiveness and efficiency of our proposed VMPMOPSO algorithm. We compare the performance of the proposed algorithm with three competing algorithms including a traditional single-objective algorithm FDD [52] and two multiobjective algorithms MGGA [19] and VMPACS [20] in terms of energy consumption, SLA violation, and resource wastage. We use CloudSim toolkit [57] to simulate and evaluate our proposed algorithm which was coded in java language and run on Intel Core i3 with 1.90 GHz CPU and 4 GB RAM. For our experiments, we use one cloud data center with 200 physical machines. We randomly generated 200 VMs instances. Each instance requests CPU and memory resources. We present the linear correlations of CPU and memory utilizations into the VM instances and use the method in [22] listed in Algorithm 2 to generate VM deployment requests.where both D_i^{cpu} and D_i^{mem} form the VM deployment request. ε_h, ε_n, and μ are random numbers verified randomly in the interval [0, 1]. $\overline{D^{\text{CPU}}}$ and $\overline{D^{\text{mem}}}$ represent the reference values of CPU and main memory demands, respectively.

To control the generated VM deployment requests under different parameters, we assume two cases. In the first case, the range of generated CPU and memory resources is in the interval [0, 50), and the references values $\overline{D^{\text{CPU}}}$ and $\overline{D^{\text{mem}}}$ are set to 25%, in which the probability value that controls the correlation between different reference values, denoted as P, is varying and set to 0.0, 0.25, 0.50, 0.75, and 1.0. Also, the average correlation coefficients of CPU and memory demands are set to −0.754, −0.348, −0.072, 0.371, and 0.755 per deployment request. These five correlation coefficient statuses represent strong-negative, weak-negative, no, weak-positive, and strong-positive. In the second case, the range of generated CPU and memory demands is in the interval [0, 90%), and the reference values $\overline{D^{\text{CPU}}}$ and $\overline{D^{\text{mem}}}$ are set to 45%, in which the average correlation coefficient of CPU and main memory demands, referred to as the term "corr.", is −0.755, −0.374, −0.052, 0.398, and 0.751.

The parameter values of the optimization method in our VMPMOPSO algorithm are randomly verified as follows. The population size is limited to 300 particles, and the maximum number of iterations $t_{\max}$ is 500. The inertia weight ϕ was determined randomly in the interval [0, 0.8], the acceleration constants ψ_1, ψ_2 were verified randomly in the interval [1.5, 2.5], and the random numbers ϑ_1, ϑ_2 were verified randomly in the interval [0, 1].

6.1. Experiment I. We verified the performance of our proposed VMPMOPSO algorithm by comparing it with three algorithms, namely, FDD [52], MGGA [19], and VMPACS [20]. Figure 2 shows the total power consumption,

Input:
n, number of the VMs;
m, number of the physical servers;
R, population size;
D, dimension of particles
$E^{\max}$, maximum size of archive E;
$t_{\max}$, maximum number of iteration;
$b^{\min}$, $b^{\max}$, minimum and maximum bounds of random number.
Output: E: the archive set (Pareto set) of the optimal VM placement solutions.

Set $t = 0$, $S_{id} = R$ and $(b^{\min}, b^{\max})$, $V_{id} = 0$, $A = \varnothing$, $i = (1, 2, \ldots, R)$, $d = (1, 2, \ldots, D)$,
Repeat
  for $i = 1$ to R **do**
    Generate a random position $S_i = (S_{i1}, S_{i2}, \ldots, S_{id}, \ldots,)$ for the ith particle;
    Evaluate the fitness value of S_i for all i using formulas (6), (7), and (8);
      / * External archive updating * /
    if $S_i \succ \forall E[\lambda]$ **then**
      Update E: $E \leftarrow E \cap \overline{E}[\theta] \cup S_i$, θ is the do minated solution;
      else if $(S_i \not\succ \forall E) \wedge (S_i \not\prec \forall E)$ **then**
      Update E: $E \leftarrow E \cup S_i$;
    end if
  if $|E| > E^{\max}$ **then**
    for $j = 1$ to $|E|$ **do**
      Calculate the crowding entropy CE of each solution in E according to formula (17);
      Sort archive E in descending CE values;
      Update E: $E \leftarrow E \cap \overline{E}[\delta]$, δ is the removed solution (redundant solutions);
        Select the global best position P_{gd} randomly from the top of E
    end for
  end if
    Personal best position P_{id} update;
  end for
for $i = 1$ to R **do**
    for $d = 1$ to D **do**
      Update the velocity V_{id} using formula (12);
      Compute the new position S_{id} using formula (13)
    end for
  end for
$t = t + 1$;//increment steps
Until $(t > t_{\max})$
Output archive E (the Pareto set)

Algorithm 1: Pseudocode of the proposed VMPMOPSO.

for $i = 1 : 200$ **do**
  $D_i^{\text{cpu}} = 2 \times \overline{D^{\text{cpu}}} \times \varepsilon_h$
  $D_i^{\text{mem}} = 2 \times \overline{D^{\text{mem}}} \times \varepsilon_n$
  if $((\mu < \text{P})$ && $(D_i^{\text{cpu}} \geq \overline{D^{\text{cpu}}}) || (\mu \geq P)$ && $(D_i^{\text{cpu}} < \overline{D^{\text{cpu}}}))$ **then**
    $D_i^{\text{ram}} = D_i^{\text{mem}} + \overline{D^{\text{mem}}}$
  end if
end for

Algorithm 2: VM deployment request generator.

wastage resource, and SLA violation of the four algorithms (FDD, MGGA, VMPACCS, and VMPMPSO) when the reference value is set to $\overline{D^{\text{CPU}}} = \overline{D^{\text{mem}}} = 25\%$, and the correlation coefficient is 0.754, −0.348, −0.072, 0.371, and 0.755, respectively. As shown in Figure 2(a), the VMPMOPSO algorithm can reduce power consumption significantly. It is easy to find that the VMPMOPSO algorithm has the least power consumption compared to FDD, MGGA, and VMPACCS. This is because the proposed VMPMOPSO based on crowding entropy can effectively improve the diversity among the obtained solutions of the VM placement problem and explore more feasible solutions in unknown regions and, therefore, increase the probability of finding the most optimal placement solution that minimizes the total

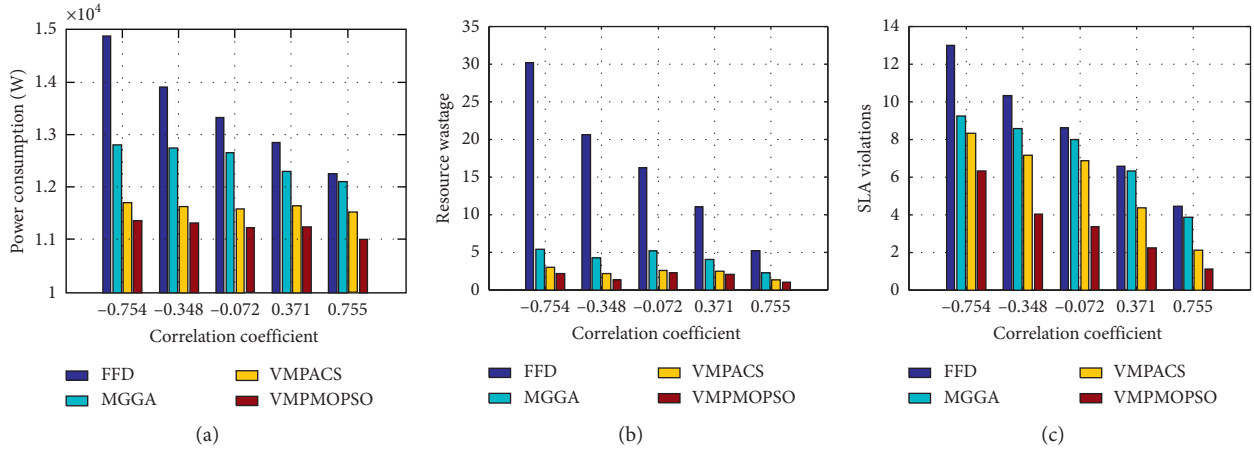

Figure 2: (a) Power consumption, (b) resource wastage, and (c) SLA violation of different algorithms in the case of $\overline{D^{\mathrm{CPU}}}$ and $\overline{D^{\mathrm{RAM}}} = 25\%$.

power consumption. Similarly, we compare the resource wastage of FDD, MGGA, VMPACS, and VMPMOPSO algorithms, as shown in Figure 2(b). We can observe that the VMPMOPSO algorithm is superior to FFD, MGGA, and VMPACS algorithms in all scenarios. VMPMOPSO can effectively find the optimal placement solution that leaves the least amount of resource wastage. Finally, the third parameter to evaluate is SLA violation which is defined as the degradation of performance in servers after placing VM. For the four algorithms (FDD, MGGA, VMPACS, and VMPMOPSO) with different correlation coefficients (−0.754, −0.348, −0.072, 0.371, and 0.755) and when $\overline{D^{\mathrm{CPU}}} = \overline{D^{\mathrm{mem}}} = 25\%$, the SAL violation is shown in Figure 3(c). It is easy to find that VMPMOPSO leads to the least SLA violation compared with FDD, MGGA, and VMPACS, respectively. The reason is that the VMPMOPSO algorithm considers CPU utilization and memory usage of the physical server before making the placement decision.

Figure 3 reports the results of the second scenario when the reference value is $\overline{D^{\mathrm{CPU}}} = \overline{D^{\mathrm{mem}}} = 45\%$ and Corr. is set to −0.755, −0.374, −0.052, 0.398, and 0.751, respectively. As shown in Figures 3(a)–3(c), once again our proposed VMPMOPSO algorithm is compared to FFD, MGGA, and VMPACS algorithms in terms of power consumption, resource wastage, and SLA violation. It is shown that, in this scenario also, our proposed VMPMOPSO algorithm offers better results.

To demonstrate the effectiveness of the VMPMOPSO algorithm, we conducted another simulation experiment to compare MGGA, VMPACS, and our proposed VMPMOPSO algorithm. Figures 4 and 5 show the convergence curves of the three algorithms. Each point on the curve refers to an optimal solution for the problem of VMP in the large-scale data center. In addition, the set of solutions that are not dominated by any other point combined together in a set of nondominated solutions is called the Pareto set. Here, three objective functions are defined as follows: f_1 represents SLA violation, f_2 represents power consumption, and f_3 represents resource wastage. The distribution of the Pareto optimal solutions of MGGA, VMPACS, and VMPMOPSO after running 100 generations is shown in Figure 4. Similarly, the distribution of the Pareto optimal solutions of MGGA, VMPACS, and VMPMOPSO after running 500 generations is shown in Figure 5. As shown in Figures 4 and 5, the Pareto solution set given by the VMPMOPSO algorithm has better distribution and convergence than that from MGGA and VMPACS. It can be seen that the VMPMOPSO algorithm can quickly move toward the Pareto front, and the results are closer to the Pareto optimal solution than MGGA and VMPACS. Additionally, we observe that the obtained solutions by VMPMOPSO do not concentrate on a small region but distribute over the large search space of the target problem that leads to more diversity on the search and increased the chance to converge the Pareto optimal solution with high speed.

6.2. Experiment II. In this section, we carried out another experiment to evaluate the performance of the VMPMOPSO algorithm with two multiobjective algorithms: MGGA and VMPACS. We calculated two metrics, Generational Distance (GD) [58] and Spacing (SP) [59], for each algorithm. GD and SP can be defined as follows:

$$\mathrm{GD} = \frac{1}{n}\sqrt{\sum_{i=1}^{n} d_i^2}, \tag{21}$$

where d_i is the Euclidean distances of the ith solution to the true Pareto front and n is the number of nondominated solutions. The smaller the value of GD, the better the convergence toward the Pareto front.

$$\mathrm{SP} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(\overline{d} - d_i\right)^2}, \tag{22}$$

where $d_i = \min_j\left\{\sum_{k=1}^{m} f_k^i - f_k^j\right\}$, $(i, j = 1, 2, \ldots, n)$, f is the objective function, $\overline{d}$ is the average of all d_i, and m is the number of objectives. The value of SP close to 0 indicates that the solutions that have been found by the algorithm are a good solution set. Table 1 shows the GD and SP obtained by VMPMOPSO and VMPACS algorithms. In this table, the

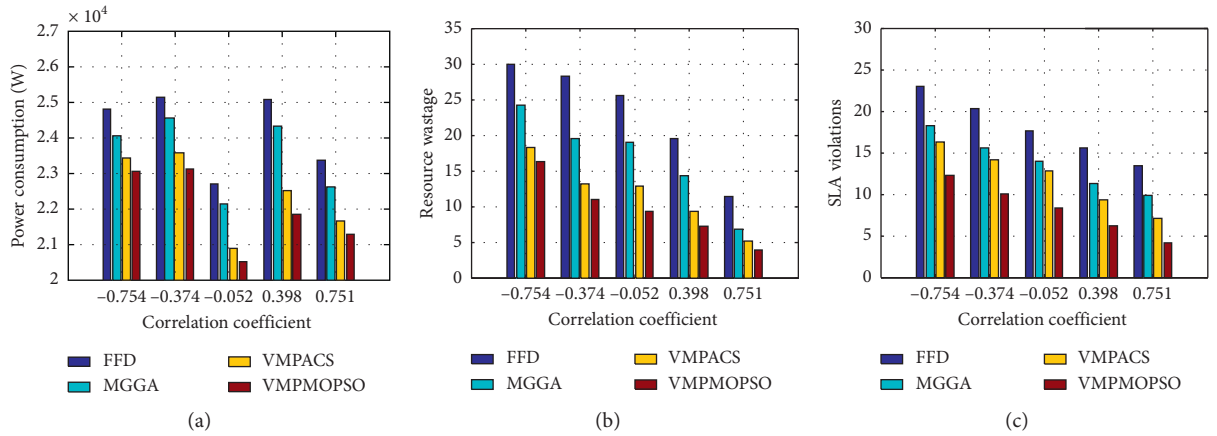

Figure 3: (a) Power consumption, (b) resource wastage, and (c) SLA violation of different algorithms in the case of $\overline{D^{\mathrm{CPU}}}$ and $\overline{D^{\mathrm{RAM}}} = 45\%$.

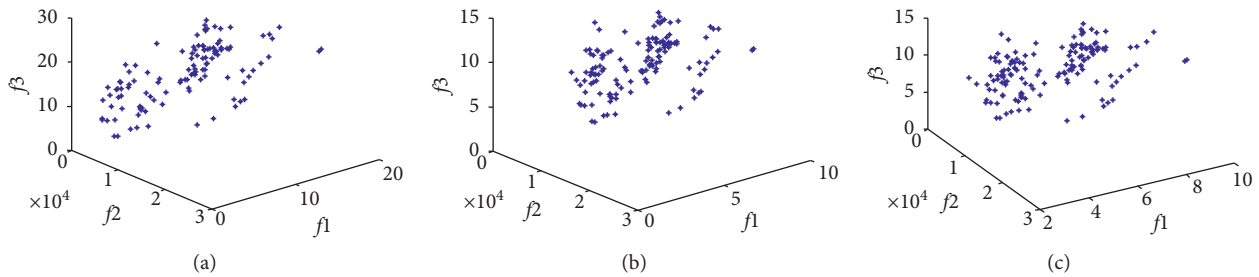

Figure 4: The distribution of Pareto optimal solution set of (a) MGGA, (b) VMPACS, and (c) VMPMOPSO after 100 generations.

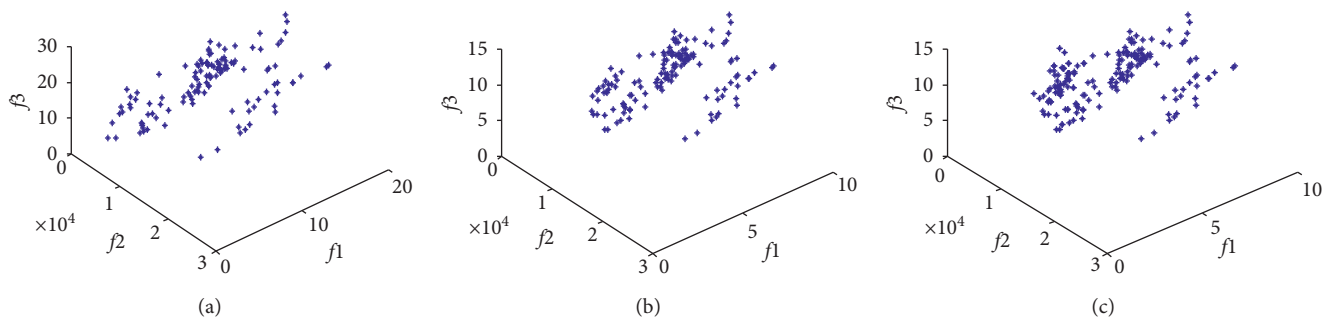

Figure 5: The distribution of Pareto optimal solution set of (a) MGGA, (b) VMPACS, and (c) VMPMOPSO after 500 generations.

Table 1: GD and SP performance comparison of VMPMPSO, VMPACS, and MGGA.

Reference value	Corr.	GD			SP		
		MGGA	VMPACS	VMPMPSO	MGGA	VMPACS	VMPMPSO
$\overline{D^{\mathrm{CPU}}} = \overline{D^{\mathrm{mem}}} = 25\%$	−0.754	0.07321	0.04513	0.02172	0.5391	0.2265	0.1731
	−0.348	0.05864	0.03520	0.01834	0.3873	0.1670	0.1213
	−0.072	0.05393	0.03274	0.01025	0.3235	0.1383	0.1134
	0.371	0.03275	0.02145	0.00831	0.2836	0.1152	0.1087
	0.750	0.02381	0.01956	0.00547	0.1678	0.0778	0.0542
$\overline{D^{\mathrm{CPU}}} = \overline{D^{\mathrm{mem}}} = 45\%$	−0.755	0.03540	0.02545	0.01325	0.1953	0.1082	0.1024
	−0.374	0.04171	0.02638	0.01473	0.2284	0.1163	0.1133
	−0.052	0.04838	0.03156	0.01891	0.2845	0.1361	0.1258
	0.392	0.02576	0.01672	0.00739	0.1570	0.0674	0.0532
	0.751	0.01839	0.00878	0.00324	0.1079	0.0495	0.0270

column "Corr." shows the correlation coefficients for the CPU utilization and memory usage.

As revealed in Table 1, we can conclude that the average GD values of VMPMOPSO are less than those of the other two algorithms (MGGA and VMPACS). These results were due to the fact that the obtained solutions by the VMPMOPSO algorithm are closer to the true Pareto front than the other solutions given by MGGA and VMPACS. Regarding the SP metric, it is clear that the VMPMOPSO algorithm performs the best results compared to the MGGA and VMPACS algorithms. In other words, the distribution of the optimal solution set of VMPPSO is more uniform than the other optimal solutions set of MGGA and VMPACS algorithms.

7. Conclusions

In this paper, we propose a novel VMP algorithm called VMPMOPSO. VMPMOPSO treats the problem of VMP as a multiobjective optimization problem and uses MOPSO to optimally solve the VMP problem. The proposed approach optimizes multiobjectives such as SLA violations, power consumption, and resource wastage. We have conducted two simulation experiments to evaluate the effectiveness and efficiency of the VMPMOPSO algorithm. VMPMOPSO was compared with a single-objective FDD algorithm and two existing multiobjective algorithms, MGGA and VMPACS. The experimental results show that VMPMOPSO is more effective and efficient than FDD, MGGA, and VMPACS algorithms. Furthermore, VMPMOPSO with the help of the crowding entropy method maintains good population diversity and can quickly move toward the Pareto front. In general, the VMPMOPSO algorithm can find the optimal VMP solution that offers the best compromises among different objectives and not only helps providers to maximize the profit by reducing energy consumption but also helps them to keep the SLA at the desired level. Future work will focus on using VMPMOPSO to handle some real-world multiobjective problems.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, 2009.

[2] T. Liu, T. Lu, W. Wang et al., "SDMS-O: a service deployment management system for optimization in clouds while guaranteeing users' QoS requirements," *Future Generation Computer Systems*, vol. 28, no. 7, pp. 1100–1109, 2012.

[3] B. Jennings and R. Stadler, "Resource management in clouds: survey and research challenges," *Journal of Network and Systems Management*, vol. 23, no. 3, pp. 567–619, 2015.

[4] L. Christian and P. E. Belady, "In the data center, power and cooling costs more than the it equipment it supports," *Electronics-Cooling*, vol. 13, no. 1, 2007.

[5] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, 2007.

[6] S. K. Garg, S. Versteeg, and R. Buyya, "A framework for ranking of cloud computing services," *Future Generation Computer Systems*, vol. 29, no. 4, pp. 1012–1023, 2013.

[7] P. Barham, B. Dragovic, K. Fraser et al., "Xen and the art of virtualization," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 164–177, 2003.

[8] M. Cardosa, M. Korupolu, and A. Singh, "Shares and utilities based power consolidation in virtualized server environments," in *Proceedings of IFIP/IEEE Integrated Network Management (IM'09)*, pp. 327–334, Long Island, NY, USA, June 2009.

[9] R. Ranjana and J. Raja, "A survey on power aware virtual machine placement strategies in a cloud data center," in *Proceedings of IEEE International Conference on Green Computing, Communication and Conservation of Energy (ICGCE'13)*, pp. 747–752, Chennai, India, December 2013.

[10] T. Dillon, C. Wu, and E. Chang, "Cloud computing: issues and challenges," in *Proceedings of 24th IEEE International Conference on Advanced Information Networking and Applications (AINA'10)*, pp. 27–33, Perth, Australia, April 2010.

[11] J. Xu and J. Fortes, "A Multi-objective virtual machine placement in virtualized data center environments," in *Proceedings of Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*, pp. 179–188, IEEE, Hangzhou, China, December 2010.

[12] S. Chaisiri, B. Lee, and D. Niyato, "Optimal virtual machine placement across multiple cloud providers," in *Proceedings of IEEE Asia-Pacific Services Computing Conference*, pp. 103–110, Singapore, December 2009.

[13] K. Mills, J. Filliben, and C. Dabrowski, "Comparing VM placement algorithms for on-demand clouds," in *Proceedings of 3rd IEEE International Conference on Cloud Computing Technology and Science (CloudCom'11)*, pp. 91–98, Athens, Greece, August 2011.

[14] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing SLA violations," in *Proceedings of 10th IEEE Symposium on Integrated Management (IM)*, pp. 119–128, Munich, Germany, May 2007.

[15] X. Liao, H. Jin, and H. Liu, "Towards a green cluster through dynamic remapping of virtual machines," *Future Generation Computer Systems*, vol. 28, no. 2, pp. 469–477, 2012.

[16] H. N. Van, F. D. Tran, and J.-M. Menaud, "Performance and power management for cloud infrastructures," in *Proceedings of 2010 IEEE 3rd International Conference on Cloud Computing (CLOUD)*, pp. 329–336, IEEE, Miami, FL, USA, July 2010.

[17] S. Agrawal, S. K. Bose, and S. Sundarrajan, "Grouping genetic algorithm for solving the server consolidation problem with conflicts," in *Proceedings of 1st ACM/SIGEVO Summit on Genetic and Evolutionary Computation (GEC'09)*, pp. 1–8, Shanghai China, June 2009.

[18] E. Feller, L. Rilling, and C. Morin, "Energy-aware ant colony based workload placement in clouds," in *Proceedings of 2011 IEEE/ACM 12th International Conference on Grid Computing*, pp. 26–33, IEEE, Newport Beach, CA, USA, May 2011.

[19] A. Al-Dulaimy, W. Itani, R. Zantout, and A. Zekri, "Type-Aware virtual machine management for energy efficient cloud data centers," *Sustainable Computing: Informatics and Systems*, vol. 19, 2018.

[20] J. Xu and J. Fortes, "A multi-objective approach to virtual machine management in datacenters," in *Proceedings of 8th ACM International Conference on Autonomic Computing (ICAC'11)*, pp. 225–234, Karlsruhe Germany, June 2011.

[21] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," *Journal of Computer and System Sciences*, vol. 79, no. 8, pp. 1230–1242, 2013.

[22] Y. Ajiro and A. Tanaka, "Improving packing algorithms for server consolidation," in *Proceedings of International Conference for the Computer Measurement Group*, pp. 399–406, San Diego, CA, USA, December 2007.

[23] T. H. Duong-Ba, T. Nguyen, B. Bose, and T. T. Tran, "A Dynamic virtual machine placement and migration scheme for data centers," *IEEE Transactions on Services Computing*, vol. 1, 2018.

[24] B. Speitkamp and M. Bichler, "A mathematical programming approach for server consolidation problems in virtualized data centers," *IEEE Transactions on Services Computing*, vol. 3, no. 4, pp. 266–278, 2010.

[25] A. Fatima, N. Javaid, T. Sultana et al., "Virtual machine placement via bin packing in cloud data centers," *Electronics*, vol. 7, no. 12, p. 389, 2018.

[26] X.-F. Liu, Z.-H. Zhan, J. D. Deng, Y. Li, T. Gu, and J. Zhang, "An energy efficient ant colony system for virtual machine placement in cloud computing," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 113–128, 2018.

[27] V. Reddy, G. Gangadharan, and G. Rao, "Energy-aware virtual machine allocation and selection in Cloud data centers," *Soft Computing*, vol. 23, 2019.

[28] K. Babu and P. Samuel, "Virtual machine placement for improved quality in IaaS cloud," in *Proceedings of Advances in Computing and Communications (ICACC)*, Kochi, India, August 2014.

[29] A. Verma, P. Ahuja, and A. Neoqi, "pMapper: power and migration cost aware application placement in virtualized systems," in *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, pp. 243–264, Leuven, Belgium, December 2008.

[30] A. Beloglazov and R. Buyya, "Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 7, pp. 1366–1379, 2013.

[31] C. Liu, C. Shen, and S. Li, "A new evolutionary multi-objective algorithm to virtual machine placement," in *Proceedings of Virtualized Data Center in Software Engineering and Service Science (ICSESS)*, Tianjin, China, December 2014.

[32] Q. Zheng, R. Li, X. Li et al., "Virtual machine consolidated placement based on multi-objective biogeography-based optimization," *Future Generation Computer System*, vol. 54, pp. 96–122, 2016.

[33] D. Kumar and Z. Raza, "A PSO based VM Resource scheduling model for Cloud computing," in *Proceedings of IEEE International Conference on Computational Intelligence & Communication Technology*, Ghaziabad, India, December 2015.

[34] S. E. Dashti and A. M. Rahmani, "Dynamic VMs placement for energy efficiency by PSO in Cloud computing," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 28, no. 1-2, pp. 97–112, 2015.

[35] A. Tripathi, I. Pathak, and D. Vidyarthi, "Energy efficient VM placement for effective Resource utilization using modified binary PSO," *The Computer Journal*, vol. 61, 2017.

[36] "Google app-engine," https://cloud.google.com/products/app-engine/.

[37] "Amazon elastic compute cloud (EC2)," http://aws.amazon.com/ec2/.

[38] A. Beloglazov and R. Buyya, "Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers," in *Proceedings of 8th International Workshop on Middleware for Grids, Clouds and E-Science, ACM*, Bangalore, India, December 2010.

[39] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[40] X. Fan, W. Weber, and L. Barroso, "Power provisioning for a warehouse-sized computer," in *Proceedings of 34th Annual International Symposium on Computer Architecture*, pp. 13–23, New York, NY, USA, June 2007.

[41] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of Sixth International Symposium on Micro Machine and Human Science (MHS'95)*, pp. 39–43, Nagoya, Japan, January 1995.

[42] A. Salman, I. Ahmad, and S. Al-Madani, "Particle swarm optimization for task assignment problem," *Microprocessors and Microsystems*, vol. 26, no. 8, pp. 363–371, 2002.

[43] L. Z. Hang, Y. H. Chen, R. Y. Sun, S. Jing, and B. Yang, "A task scheduling algorithm based on PSO for grid computing," *International Journal of Computational Intelligence Research*, vol. 4, no. 1, pp. 37–43, 2008.

[44] M. Reyes-Sierra and C. A. C. Colello, "Multi-Objective particle swarm optimizers: a survey of the state-of-the-art," *International Journal of Computational Intelligence Research*, vol. 2, no. 3, pp. 287–308, 2006.

[45] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: nsga-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[46] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[47] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 256–279, 2004.

[48] C. A. Coello Coello and M. S. Lechuga, "MOPSO: a proposal for multiple objective particle swarm optimization," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC'02)*, pp. 1051–1056, Honolulu, HI, USA, May 2002.

[49] K. E. Parsopoulos and M. N. Vrahatis, "Particle swarm optimization method in multi-objective problems," in *Proceedings of ACM 2002 Symposium on Applied Computing (SAC'2002)*, pp. 603–607, Madrid, Spain, March 2002.

[50] J. J. Durillo, J. García-Nieto, A. J. Nebro, C. A. C. Coello, F. Luna, and E. Alba, "Multi-objective particle swarm optimizers: an experimental comparison," *Lecture Notes in Computer Science*, vol. 1, pp. 495–509, 2009.

[51] A. J. Nebro, J. J. Durillo, J. Garcia-Nieto, C. A. Coello Coello, F. Luna, and E. Alba, "SMPSO: a new PSO-based

metaheuristic for multi-objective optimization," in *Proceedings of IEEE Symposium on Computational Intelligence in Multi-Criteria Decision Making*, pp. 66–73, Nashville, TN, USA, March 2009.

[52] N. AI. Moubayed, A. Pertovski, and J. McCall, "D^2MOPSO: MOPSO based on decomposition and dominance with archiving using crowding distance in objective and solution spaces," *Evolutionary Computation*, vol. 22, no. 1, pp. 47–77, 2014.

[53] X. D. Li, "A non-dominated sorting particle swarm optimizer for multi-objective optimization," *Lecture Notes in Computer Science*, pp. 37–48, Springer, Berlin, Germany, 2003.

[54] X. D. Li, "Better spread and convergence: particle swarm multi-objective optimization using the maximin fitness function," *Lecture Notes in Computer Science*, pp. 17–28, Springer, Berlin, Germany, 2004.

[55] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: improving the strength pareto evolutionary algorithm," *Computer Engineering and Networks Laboratory TIK*, ETH Zurich, Zürich, Switzerland, 2001.

[56] Y.-N. Wang, L.-H. Wu, and X.-F. Yuan, "Multi-objective self-adaptive differential evolution with elitist archive and crowding entropy-based diversity measure," *Soft Computing*, vol. 14, no. 3, pp. 193–209, 2010.

[57] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[58] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.

[59] J. Schott, "Fault tolerant design using single and multicriteria genetic algorithm optimization," Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1995.